Autonomous Manipulation Learning for Similar Deformable Objects via Only One Demonstration

Yu Ren[†1,2,3] Ronghan Chen[†1,2,3] Yang Cong[1,2*]
[1]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences
[2]Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences, Beijing, 100049, China.
renyu@sia.cn, chenronghan@sia.cn congyang81@gmail.com

Abstract

In comparison with most methods focusing on 3D rigid object recognition and manipulation, deformable objects are more common in our real life but attract less attention. Generally, most existing methods for deformable object manipulation suffer two issues, 1) ***Massive demonstration****: repeating thousands of robot-object demonstrations for model training of one specific instance; 2)* ***Poor generalization****: inevitably re-training for transferring the learned skill to a similar/new instance from the same category. Therefore, we propose a category-level deformable 3D object manipulation framework, which could manipulate deformable 3D objects with only one demonstration and generalize the learned skills to new similar instances without re-training. Specifically, our proposed framework consists of two modules. The Nocs State Transform (NST) module transfers the observed point clouds of the target to a pre-defined unified pose state (i.e., Nocs state), which is the foundation for the category-level manipulation learning; the Neural Spatial Encoding (NSE) module generalizes the learned skill to novel instances by encoding the category-level spatial information to pursue the expected grasping point without re-training. The relative motion path is then planned to achieve autonomous manipulation. Both the simulated results via our* $\mathbf{Cap_{40}}$ *dataset and real robotic experiments justify the effectiveness of our framework.*

1. Introduction

Autonomous 3D object recognition and manipulation [1, 11, 12, 36] is crucial for robots and has broad applications for our human lives, e.g., the bin-picking for industrial robot, housework for service robot. Recently, most

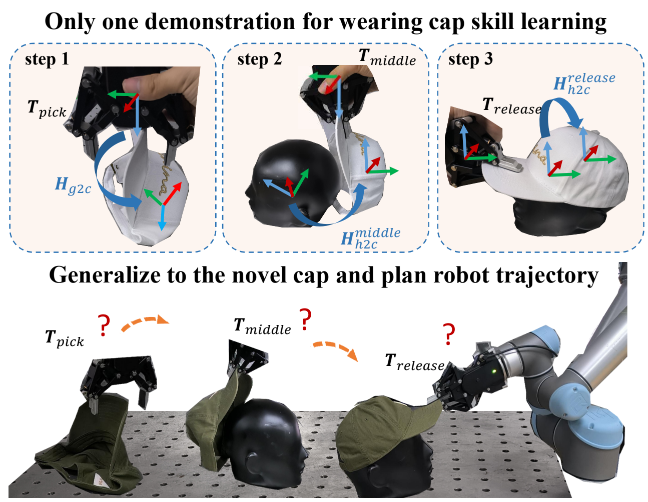


Figure 1. The demonstration of our proposed framework for wearing the cap: 1) the manipulation skill is learned via only one demonstration; 2) the learned manipulatin skill could be generalized to other novel caps without re-training.

state-of-the-arts focus on 3D rigid object recognition and manipulation [9]; in contrast, less attention is concerned on non-rigid/deformable objects, which are actually more common in our real lives, such as clothes, animals, vegetables or even human ourselves. This is partially because the motion space representation of rigid objects is relatively simple and could be represented by a 6-DOF linear vector; nevertheless, the deformations of non-rigid/deformable objects are difficult to match and get a uniform linear representation. Recently, the data-driven methods [2, 32, 39] achieve significant progress for non-rigid/deformation object manipulation, which can estimate the states of the deformable objects and predict the appropriate manipulations simultaneously. However, these methods suffer two issues: 1) **Massive demonstration**: thousands of repetitions of robot-object demonstrations are needed to train the model

*This work is supported by the National Nature Science Foundation of China under Grant 62127807 and 62225310. The corresponding author is Prof. Yang Cong.
†These authors contributed equally to this work

to manipulate one specific instance; 2) **Poor generalization**: re-training is inevitably needed to transfer the learned skill from the known instance to a similar/new instance from the same category. Let us take the task of wearing a cap as an example, the training phase repeats thousands of times wearing procedures in the simulated environment or in the real world; however, to handle a novel cap, we need to recollect data of the new cap and re-train the model to adapt to the shape and deformation of the new cap. Therefore, these complex and time-consuming procedures limit their practical applications.

In this paper, we aim at solving a more challenging task—learning to manipulate freely *deformed unseen* objects of the same category, from only one demonstration. To learn from few demonstration, prior works learn dense features [13], and estimate the grasp pose on target objects by feature matching. However, such features are not robust to large deformation, and cannot generalize to novel objects.

To tackle such problem, we develop two new components: 1) the **Nocs State Transfer (NST)** module transfers the target objects under arbitrary deformation to a pre-defined canonical state (*i.e*., **Nocs state**), thus effectively eliminating the disturbance caused by deformation in feature matching; 2) the **Neural Spatial Encoding (NSE)** module learns to encode the Nocs coordinates obtained from the NST into category-level features via self-reconstruction and contrastive losses. By encoding and constrating between similar geometric structures, the NSE features can generalize well, thus further enabling effective manipulation pose transfer on novel objects. Our framework needs to be pre-trained only once for the whole object category, *i.e*., without re-training for some specific new instance. For different manipulation tasks on the objects within the same category, only one demonstration is needed. Then, the robot could plan the related manipulation path accordingly.

The main contributions are presented as follows:

- We present a novel framework which learns to manipulate similar non-rigid/deformable objects via only one robot demonstration. To the best of our knowledge, this is the earliest exploration about generalization learning of deformable object manipulation.
- Our framework can generalize the learned skills from known instances to other novel/similar instances without tedious data collection or model re-training, which expands its application possibilities in the real world.
- We contribute a simulated caps dataset containing 4000 annotated frames of 40 deformable caps; moreover, a real robotic system is also designed to serve people wearing caps automatically. Both the simulated results and real-world experiments justify the effectiveness of our proposed framework and system.

2. Related Work

2.1. Deformable Object Manipulation

Various approaches are proposed for deformable object manipulation in decades of robotic research [46]. Conventional methods [17, 23, 43] leverage the high-fidelity physics-based model to estimate and simulate the state of the target object. [23] leverages a minimal-energy curve to plan the execution path for ropes. [17] conduct dressing task using the physics simulation of humans and clothes. [43] dresses a person using the optical Flow-based method and state regression. Nowadays, data-driven methods achieve promising progress in deformable object manipulation. [35] trains a network to combine cloth manipulations with shape changes of the target to perform folding tasks. ACID [32] learns implicit representations of the states of the target object to make plan manipulation. [31] uses a goal-conditioned transporter network to tackle manipulation of cables, fabrics, and bags spanning. However, the above methods are limited in real-world applications since they repeat robot-object interactions in a time-consuming way and cannot generalize to novel instances directly.

2.2. Learning from Demonstration

Demonstration learning is a powerful and intuitive method to teach robots to perform complex tasks [15, 28, 33]. Observing expert demonstrations, it learns to generate appropriate motion steps from input observations [24, 29, 41]. Nonetheless, these methods cannot handle novel instances from the same category without re-training. Recently, several works [33, 38] explore demonstration learning on category-level rigid object manipulation. [38] learns to perform industrial tasks interacting with novel instances from a video demonstration. [33] proposes an implicit neural field to generalize learned skills at the category level. However, these methods cannot be applied to deformable objects. Compared with manipulation on rigid objects whose pose can be fully specified as a low-dimensional vector [12, 14, 44], deformable objects have infinite continuous configuration (*i.e.,* pose) spaces, severely self-occlusion. These characteristics make the skill generalization of the category-level deformable objects hard to achieve.

2.3. Implicit Neural Representations

Implicit neural representation [22] describes the surface or volume of the 3D object as mapping functions and makes great progresses in representing 3D geometric shapes [4, 25, 45, 48]. It has been well-extended to various 3D tasks like scene reconstruction [7, 34, 37], scene understanding [47], and view Synthesis [19, 26]. Most saliently, several works combine the implicit neural representation with object robot manipulation [18, 32, 33] and achieve promising performance. However, these methods are either designed

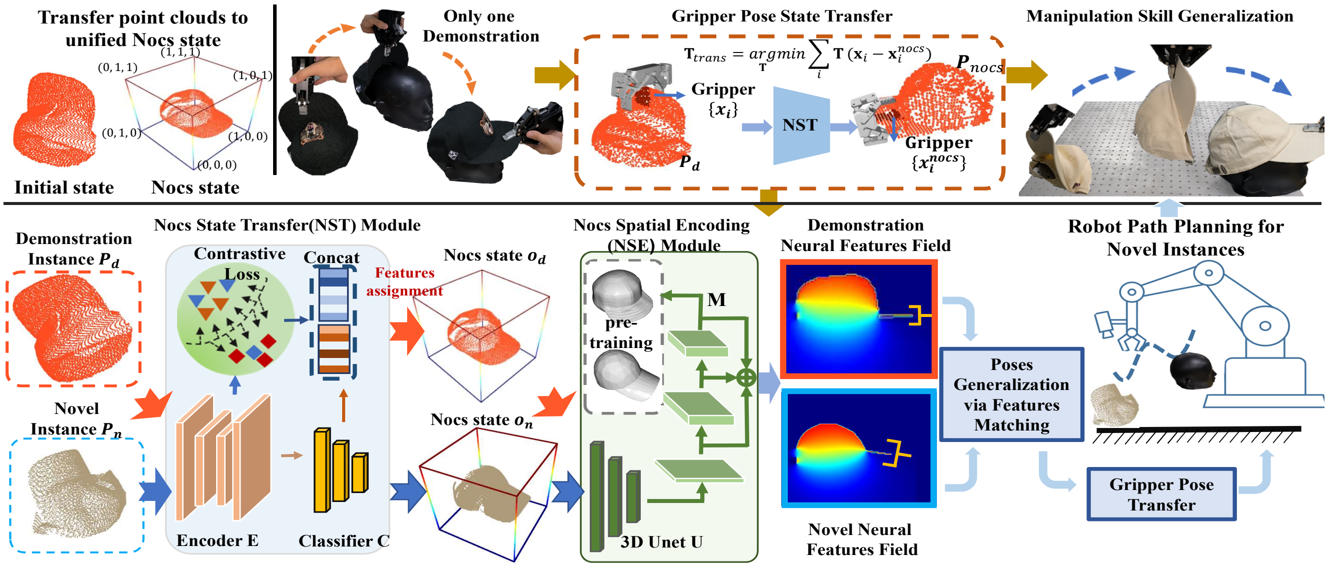


Figure 2. Overall workflow of our developed framework, which is made up of two significant modules, *i.e.,* Nocs State Transfer (NST) and Neural Spatial Encoding (NSE). Specifically, the NST module transfers the observed point cloud of the target object from the deformable pose state to the unified Nocs state; then the NSE module encodes the query coordinates around the transferred point cloud to the neural spatial features. During the demonstration phase, the NSE module intends to learn the manipulation skill (spatial relationships between the end-gripper, the target, and the environment) from robot-object demonstration. During the test phase, the learned skill is generalized to novel objects. The total framework needs to be pre-trained only one time for each category.

for rigid objects or cannot handle novel instances. Therefore, learning category-level skills for deformable objects via one demonstration is still an unsolved problem.

3. Method

3.1. Preliminary

Normalized Canonical state. Normalized Canonical state (Nocs state) is a pre-defined pose state of the deformable object [7]. As shown in Fig. 2 (top left), we define the Normalized Canonical State for the caps by first translating them to the center of a unit cube (Nocs cube), and then simulating they were worn on a head. Finally, we scale the cap until its longest bounding box edge matches the unit cube to get the Nocs state of the caps.

Here, we further provide the necessity for introducing the Nocs state. Compared with the rigid transformations which can be described by a low-dimensional vector (*i.e.*, Rotation and translation), non-rigid deformation has near infinite degree-of-freedom, *i.e.*, pose states [6]. As a result, it is impossible for a deep model to learn whether such new geometrics are suitable for manipulation. On the contrary, once the object point clouds are transferred to a unified state, the relative poses between objects and the environment are fixed and describable, which provides the foundation for skills generalization.

3.2. Overall Framework

Our framework aims to learn manipulation skills from few demonstrations and generalize skills to novel instances according to their deformed state and geometry diversity.

Our overall framework is shown in Fig. 2, which consists of Nocs State Transform (NST) module and Neural Spatial Encoding (NSE) module. Given the demonstration $\mathcal{D} = \{\mathbf{P}|\{\mathbf{T}\}\}$, where $\mathbf{P}$ is the point cloud of a deformable target object and $\{\mathbf{T}\}$ are key observed gripper poses for skill execution, the NST is designed to transfer the point cloud $\mathbf{P}$ together with related gripper poses $\{\mathbf{T}\}$ from arbitrary deformable states to the unified Nocs state. The transferred Nocs state gripper pose $\mathbf{H}$ determines a concrete relative pose between the gripper and the target free from the disturbance of infinite deformable states. The NSE aims to construct a neural spatial encoding function to encode the $\{\mathbf{H}\}$ into the category-level geometric features. With the cooperation of the NST and NSE, our framework learns the $\mathbf{H}$ between gripper and object under arbitrary deformable state, generalizes the $\mathbf{H}$ to novel instance o_n depending on common geometry features, and generate gripper poses $\{\mathbf{T}^{o_n}\}$ for the o_n on its deformable states from generalized $\mathbf{H}$. Finally, the skill could be executed on o_n with the trajectory calculated from generated gripper poses $\{\mathbf{T}^{o_n}\}$.

3.3. Nocs State Transfer (NST) Module

Implementation. The details about how NST transfers the observed point clouds $\mathbf{P}$ from the deformable pose state to the Nocs state $\mathbf{P}_{nocs}$ are described as follows:

Step 1: Given $\mathbf{P} \in \mathbb{R}^{N\times3}$ (N denotes the point number), we adopt the block $\mathbf{E}$ of pointnet++ [27] to encode points in

$\mathbf{P}$ into the per-point features $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^{N} = \mathbf{E}\left(\mathbf{P}; \theta_{\mathbf{E}}\right) \in \mathbb{R}^{N \times C}$, where C denotes the feature channel, and $\theta_{\mathbf{E}}$ is the weights of the encoder $\mathbf{E}$.

Step 2: The classifier $\mathbf{C}$ is employed to predict the Nocs state coordinate of each point in $\mathbf{P}$ based on $\{\mathbf{f}_i\}_{i=1}^{N}$. Specifically, we divide the Nocs cube in Fig. 2 into a 64^3 volume, and predict the id in x, y and z axes for the voxel that each point occupies $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) = \mathbf{C}\left(\mathbf{F}; \theta_{\mathbf{C}}\right)$.

Step 3: For each point, we select the voxel with maximum probabilities of three axes as the predicted voxel, and denote its centra coordinate as the corresponding point in the Nocs state. All corresponding points constitute the point clouds $\mathbf{P}_{nocs}$.

Training. In pre-training task, we train the encoder $\mathbf{E}$ and classifier $\mathbf{C}$ using the simulated data only once for each category. The object function is presented as follows:

$$\mathcal{L}_{nocs} = -\left(\mathbf{c}_x \sum \log \mathbf{p}_x + \mathbf{c}_y \sum \log \mathbf{p}_y + \mathbf{c}_z \sum \log \mathbf{p}_z\right), \tag{1}$$

where $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)$ is the output of the $\mathbf{C}$; $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ is the ground-truth Nocs coordinate for each point.

Besides, we introduce an auxiliary loss, i.e., the contrastive loss [42], to enforce the points with the same Nocs coordinates to learn similar features and vice versa. This further improves the generalization ability of category-level features in NSE. Specifically, the contrastive loss $\mathcal{L}_{const}$ is defined as follows:

$$\mathcal{L}_{const} = -\sum_{(i,j)\in\mathcal{P}} \log \frac{\exp\left(\mathbf{f}_i \cdot \mathbf{f}_j / \tau\right)}{\sum_{(\cdot,k)\in\mathcal{P}} \exp\left(\mathbf{f}_i \cdot \mathbf{f}_k\right)}, \tag{2}$$

where $\mathcal{P}$ is the set of point pairs that have the same Nocs coordinates, and $\mathbf{f}$ is the feature of the corresponding point. For a matched point pair $(i,j) \in \mathcal{P}$, we regard $\mathbf{f}_i$ as the query feature, $\mathbf{f}_j$ as the positive key feature, and $\mathbf{f}_k$ $(k \neq j)$ as the negative key feature.

3.4. Neural Spatial Encoding (NSE) Module

Given the per-point features $\mathbf{F}$ and the transferred point cloud $\mathbf{P}_{nocs}$, NSE maps arbitrary query coordinates $\mathbf{x}$ in Nocs state space to a neural spatial feature $\mathbf{f_s}$. These features encode the coordinates with category-level geometry information and make them easy to generalize to novel instances according to common geometry features.

Implementation. NSE constructs a feature volume covering the Nocs cube first and then generates features for given query coordinate by interpolating from feature volume. The details are decsribed as follows:

Step 1 : With the output of the NST module, NSE generates new point-level features for each point in $\mathbf{P}$ by concatenating its original point coordinate, predicted Nocs coordinate, predicted probability and per point features $\mathbf{f}_i \in \mathbf{F}$.

Step 2: NSE divides the nocs cube into a 32^3 volume, and assigns the features generated from step 1 to the volume according to predicted nocs coordinates. The features that are assigned to the same location of the volume will be aggregated by a max pooling operation. For those location that have no corresponding point in $\mathbf{P}$, a zero feature is initialized as the placeholder. The assigned result is denoted as $\mathbf{V} \in \mathbb{R}^{32\times32\times32\times(C+3+3+3)}$.

Step 3: NSE feeds $\mathbf{V}$ into the block $\mathbf{U}$ of 3D UNet [8] to generate dense features $\mathbf{D} = \mathbf{U}(\mathbf{V}; \theta_{\mathbf{U}})$. This embeds the features with more context information.

Step 4: NSE interpolates $\mathbf{D}$ at a given query coordinate $\mathbf{x}$, as $\mathbf{D}\left(\mathbf{x}\right) = Interpolate(\mathbf{x}|\mathbf{D})$. Then, the $\mathbf{D}\left(\mathbf{x}\right)$ is concatenated with coordinate $\mathbf{x}$, and forwarded to a MLP $\mathbf{M}$. Finally, NSE generates the spatial neural feature for the query coordinate $\mathbf{M}$ by:

$$\mathbf{f}_s = \bigoplus_{i=1}^{L} \mathbf{M}^i(\mathbf{D}\left(\mathbf{x}; \theta_{\mathbf{D}}\right) \bigoplus \mathbf{x}; \theta_{\mathbf{M}}), \tag{3}$$

where $\mathbf{M}^i$ denotes the features generated from the i-th layer of $\mathbf{M}$, $\theta_{\mathbf{M}}$ is the weight parameters of the $\mathbf{M}$. Finally, we can get an implict encoding functoin by rewriting Eq.(3) as $\mathbf{f}_s = \Phi\left(\mathbf{x}|\mathbf{P}\right)$. Φ describes the ability of NSE that encodes query coordinates in Nocs space to features given observed point cloud $\mathbf{P}$.

Training. We train $\mathbf{U}$ and $\mathbf{M}$ by shape completion [7]. The key insight is that accurate completion guides the NSE to encode salient geometric features for each point. Specifically, given a coordinate in Nocs cube, the $\mathbf{U}$ and $\mathbf{M}$ are leveraged to predict the generalized winding number [3, 16] $w(\mathbf{x})$ for $\mathbf{x}$:

$$\mathbf{w}(\mathbf{x}) = \mathbf{M}(\mathbf{D}\left(\mathbf{x}; \theta_{\mathbf{D}}\right) \bigoplus \mathbf{x}; \theta_{\mathbf{M}}). \tag{4}$$

Then the mesh model of the object under the Nocs state can be reconstructed by the marching cubes method [21]. The concept of generalized winding number is proposed by [16] to describe the magnitude of a query coordinate surrounded by an object surface. Intuitively, it is determined by the relative pose between the query coordinate and the object as well as the geometry structure of the object. Once the $\mathbf{U}$ and $\mathbf{M}$ can predict the winding number in the Nocs cube accurately, it is reasonable to regard they are embedded with geometric information, which enables precise manipulation pose determination.

The ground truth of the $w(\mathbf{x})$ can be determined by integrating the solid angle [16] over all Nocs points at $\mathbf{x}$. Similarly, we use the data in a simulated environment to train the parameters of $\mathbf{U}, \mathbf{M}$. Given a mesh model of a target object under the Nocs state, we sample a 128^3 point grid in the Nocs cube. During training, we use the predicted winding number from $\mathbf{M}$ and ground truth to calculate a $L1$ loss as:

$$\mathcal{L}_{implicit} = \left|\mathbf{M}^i\left(\mathbf{D}\left(\mathbf{x}\right)\right) - w(\mathbf{x})\right|, \tag{5}$$

where $w(\mathbf{x})$ is the ground-truth winding number at point $\mathbf{x}$.

3.5. Manipulation skill generalization

This section introduces how we learn and generalize manipulation skills from demonstration to novel objects leveraging NST and NSE.

Demonstration phase. Learning manipulation skills from demonstration $\mathcal{D} = \{\mathbf{P}|\{\mathbf{T}\}\}$, where $\mathbf{P}$ is the point cloud of a deformable target object and $\{\mathbf{T}\}$ are key observed gripper poses, can be divided into three steps.:

Step 1: Computing the Nocs state grippr pose $\mathbf{H}$ for $\mathbf{T}$ shown in demonstration $\mathcal{D}$. Specifically, we first transfer observed point cloud $\mathbf{P}$ to Nocs state $\mathbf{P}_{nocs}$ using NST. To obtain the Nocs state gripper pose, we need to transform the demonstrated gripper pose $\mathbf{T}$ to the Nocs space. We denote such transformation as $\mathbf{T}_{trans}$. Since $\mathbf{T}_{trans}$ cannot be directly solved, we regard the transformation of a small set of object points enclosed by the gripper as $\mathbf{T}_{trans}$. As shown in Fig. 2 (top middle), we denote the sampled point set in the observation space as $\{\mathbf{x}_i\}$, and denote their Nocs coordinates as $\{\mathbf{x}_i^{nocs}\}$. Then the transformation $\mathbf{T}_{trans}$ can be derived by solving a Procrustes problem:

$$\mathbf{T}_{trans} = \arg\min_{\mathbf{T}} \sum_i \mathbf{T}(\mathbf{x}_i - \mathbf{x}_i^{nocs}). \tag{6}$$

Finally, we can get the $\mathbf{H}$ by transferring $\mathbf{T}$ to Nocs state: $\mathbf{H} = \mathbf{T}_{trans}\mathbf{T}$.

Step 2: Encoding Nocs state gripper pose $\mathbf{H}$ with category-level geometry features to make it able to adapt the shape of novel instances. In detail, we sample N points from the model of the gripper to form a query set $\mathbf{G} = \{\mathbf{x}_i\}_{i=1}^{N}$. We then move the query set $\mathbf{G}$ to Nocs state using $\mathbf{H}$ as $\mathbf{H}(\mathbf{G}) = \mathbf{H}\mathbf{G}$. Finally, We feed $\mathbf{P}$ into NSE and encode $\mathbf{H}$ by generating the implicit features of all points in $\mathbf{H}(\mathbf{G})$ using Eq. (3) as:

$$\mathcal{Z}(\mathbf{H}|\mathbf{P}) = \Phi(\mathbf{H}(\mathbf{G})|\mathbf{P}). \tag{7}$$

Generalization phase. Given the observed point cloud $\mathbf{P}_{o_n}$ of a novel cap o_n and $\mathcal{Z}$ learned from Eq. (7), the NST and NSE modules could generalize the manipulation skill to o_n in three steps.:

Step 1: Transferring $\mathbf{P}_{o_n}$ to $\mathbf{P}_{nocs}^{o_n}$ (Nocs state) using the NST module.

Step 2: Generalizing the spatial relationships $\mathbf{H}$ to novel object o_n. Specifically, we randomly initialize a gripper pose $\mathbf{H}$ in Nocs state and represent it as:

$$\mathcal{Z}^{o_n} = \Phi(\mathbf{H}(\mathbf{G})|\mathbf{P}_{o_n}) \tag{8}$$

using the same operations in step2 of the demonstration phase, where $\mathbf{G}$ is the query set used in Eq.(7). We then try to minimize $\mathcal{L}_{\mathcal{Z}} = |\mathcal{Z}^{o_n} - \mathcal{Z}|$ in order to optimize $\mathbf{H}$ iteratively and get generalized $\mathbf{H}$ for novel objects o_n as:

$$\mathbf{H}^{o_n} = \arg\min_{\mathbf{H}} |\Phi(\mathbf{H}(\mathbf{G})|\mathbf{P}_{o_n}) - \mathcal{Z}|. \tag{9}$$

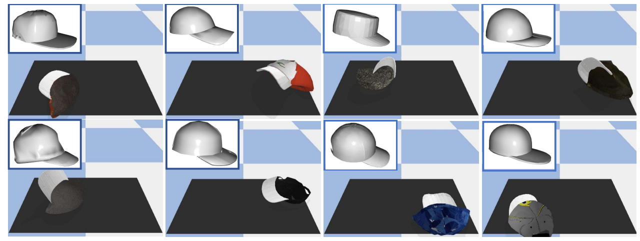


Figure 3. Some instances contained in $\mathbf{Cap_{40}}$. We totally capture 4000 scenarios for pre-training our proposed framework.

Step 3: Calculating the gripper poses for novel object o_n. We use the points set from $\mathbf{P}_{nocs}^{o_n}$ between gripper fingers at $\mathbf{H}^{o_n}$ in Nocs state and its corresponding set in original coordinates to obtain the transformation $\mathbf{T}_{itrans}$ and then we can get gripper pose for o_n as $\mathbf{T}^{o_n} = \mathbf{T}_{itrans}\mathbf{H}^{o_n}$.

Finally, the robot can plan the motion path by calculating the trajectory related to different gripper poses, and move along the trajectory to execute skill on the novel instance.

3.6. Application to cap wearing

As shown in Fig. 1, wearing cap can be divided into three steps: I) grasping the cap under arbitrary pose states, II) moving the cap to the middle pose, and III) moving the cap to the release pose. During the execution of the task, the poses of the robot gripper and the head are known. Observing a demonstration $\mathcal{D} = \{\mathbf{P}_{o_d}, \mathbf{T}_{pick}, \mathbf{T}_{middle}, \mathbf{T}_{release}\}$, where $\mathbf{P}_{o_d} \in \mathbb{R}^{N\times 3}$ is the observed point clouds of the demonstrated cap o_d; $\mathbf{T}_{pick}, \mathbf{T}_{middle}, \mathbf{T}_{release} \in \mathbb{R}^{4\times 4}$ are recorded gripper poses during task performing, the proposed framework automatically estimates $\mathbf{H}^{pick}$, $\mathbf{H}^{middle}$, and $\mathbf{H}^{release}$ that denotes the Nocs state gripper poses between gripper and cap in step I, step II and III respectively. In the test phase, given the point clouds of the novel cap o_n, the framework generalizes the $\mathbf{H}^{pick}$, $\mathbf{H}^{middle}$, $\mathbf{H}^{release}$ to the novel instance o_n and generate gripper poses and path to execute wearing cap task.

3.7. Implementation detail

We provide the overall pre-training formulation of the proposed framework as follows:

$$\begin{aligned} &\min_{\theta_{\mathbf{E}},\theta_{\mathbf{C}}} \ \mathcal{L}_{nocs} + \mathcal{L}_{const}, \\ &\min_{\theta_{\mathbf{U}},\theta_{\mathbf{D}}} \ \mathcal{L}_{implicit}, \end{aligned} \tag{10}$$

where the $\mathcal{L}_{nocs}$ and $\mathcal{L}_{const}$ are adopted to enable Encoder $\mathbf{E}$ and Classifier $\mathbf{C}$ to predict Nocs coordinates for points under initial state correctly, and the $\mathcal{L}_{implicit}$ helps 3D block $\mathbf{U}$ and MLP $\mathbf{M}$ in NSE module to extract spatial information from object point clouds under Nocs state.

We use an NVIDIA TITAN GPU for model pre-training and the code of our framework is implemented based on Pytorch [20]. ADAM optimizer is adopted as the main parameters optimizer. The MLPs $\mathbf{C}$ and $\mathbf{M}$ are composed of

Method	$GSR\,(\uparrow)$	$WSR_{\tau=0.1d}\,(\uparrow)$	$ADP_{\tau=0.1d}\,(\downarrow)$	$WSR_{\tau=0.2d}\,(\uparrow)$	$ADP_{\tau=0.2d}\,(\downarrow)$
DON [13]	55.6	18.0	0.061	25.12	0.125
NDF [33]	87.5	25.0	0.057	27.31	0.113
Garmennet [7]	90.3	43.6	0.031	65.16	0.043
Ours-w/o $\mathcal{L}_{const}$	98.2	53.4	0.033	83.17	0.048
Ours-w/o NSE	98.0	51.0	0.034	82.14	0.047
Ours	**98.5**	**55.0**	**0.030**	**86.55**	**0.041**

Table 1. Performance comparison of our framework with other methods, such as DON [13], NDF [33], and Garmentnet [7]. The Grasp Success Rate (GSR), Wearing Success Rate (WSR), and Average Distance of Points are considered for evaluation. The tuning parameter d denotes the diameter of each target object. Obviously, ours with various variations outperforms othe methods.

2 and 3 fully connected layers respectively. For **E**, a pointnet++ [27] feature encoder is adopted. The 3D Unet **U** [8] has three 3D conv-layers (128, 64, 128).

4. Simulated data generation

A simulated cap dataset $\mathbf{Cap_{40}}$ containing 40 instances is built for framework pre-training and task simulation(Fig. 3). We collect the original mesh data from sketchfab and process them with simplification, reconstruction and hole filling using Blender [40]. Finally, the modified meshes are loaded to the Pybullet [10] for simulation with appropriate material parameters.

In order to generate data for pre-training, we randomly initialize a pose above the table for a cap, and then let it freely fall to the table by gravity. We capture the RGBD image from the virtual camera when the cap deforms, and extract the point cloud of the cap as one frame. We repeat this procedure iteratively to capture 4000 frames in total. For each cap, we put it on a simulated human head in the Blender as its Nocs state. We leverage the vertices corresponding relationships to generate the ground-truth Nocs coordinates and apply [16] to generate the ground-truth generalized winding number of the Nocs mesh.

5. Experiments

Based on the proposed framework, we develop a robotic system that can conduct cap wearing tasks for humans. In this section, we verify the developed system in both simulated and real world environments. The code, dataset and videos are available on the webpage[1].

5.1. Setup

Simulated experiments: We execute our simulated experiments in pybullet [10] environment. Our simulated system includes a robotic arm, a parallel finger gripper and a virtual camera. In the experiments, an arbitrary cap is placed on the table under a random initialized pose. We then extract the point clouds of the caps from the captured

[1] https://renyu2016.github.io/DLCDO.github.io/

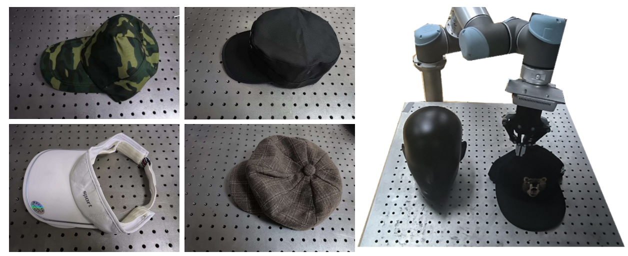


Figure 4. The real robotic testing platform includes a Ur5 robotic arm as the main body, an AG-95 parallel finger gripper, and a Realsense 435i depth camera. We evaluate the success rate with different random initialization.

RGBD images and feed the point clouds to the framework. After that, the end effector poses of three steps are obtained leveraging framework and the robot can wear the caps on the head step by step. We run 600 attempts on 20 novel caps (*i.e.* 30 attempts for each cap) that are unseen in pre-training and demonstration.

Real robotic experiments: As shown in Fig. 4, our robotic system includes an Ur5 robotic arm as the main body, an AG-95 parallel finger gripper as the manipulator and a Realsense 435i as the visual sensor. An additional human head model is placed on a table as the target for the arm wearing caps on it. The execution steps are same as simulated experiments. We run 100 attempts on 10 novel caps that do not appear in pre-training and demonstration. As shown in Fig. 4, the caps have distinct geometric structures, thus are capable of evaluating the generalization ability of the proposed method.

5.2. Baselines

As far as we know, there is no prior work that focuses on learning category-level deformable object manipulation from few demonstrations. To justify the effectiveness of our method, we compare with several baselines:

- Garmentnets [7] is a non-rigid registration method. We apply it to directly transfer the demonstrated grasp point to novel instance by estimating its correspondence on the novel instance.
- NDF [33] learns to manipulate rigid objects from a few

Methods	Grasp Success Rate	Wear Success Rate
DON [13]	0.31	0.11
NDF [33]	0.64	0.12
Garmennet [7]	0.72	0.44
Ours-w/o $\mathcal{L}_{const}$	0.85	0.76
Ours-w/o NSE	0.84	0.72
Ours	**0.85**	**0.78**

Table 2. The result of wearing cap task. 100 attempts on 10 caps.

Method	$\tau = 0.04d \uparrow$	$\tau = 0.1d \uparrow$
NST-w/o $\mathcal{L}_{const}$	57.0	95.6
NST	**62.0**	**96.8**

Table 3. The accuracy of the predicted Nocs coordinates Acc_{nocs} predicted by NST module. d denotes the diameter of the target object.

demonstrations. For deformable caps, we feed them to the framework and regard different poses as different instances. We then complete the task by following the procedure of the NDF.

• DON [13] learns pixel-level 2D descriptors of the object from images. We train DON on the simulated dataset, and generalize grasp points to novel objects by feature matching. The robot can then move to the poses to finish the task.

5.3. Evaluation metrics

Three evaluation metrics are adopted:

The **Grasp Success Rate** (GSR) is defined as:

$$GSR = \frac{GS_{num}}{T_{num}}, \tag{11}$$

where GS_{num} is the successful times of the robot grasp caps in the test phase; and T_{num} is the total test number.

The **Wearing Success Rate** (WSR) is designed as :

$$\begin{gathered} WSR = \frac{\sum \mathbb{I}\left(\frac{1}{m_v}\sum|\mathbf{v}_r - \mathbf{v}_t| < \tau\right)}{T_{num}}, \\ \mathbb{I}(t) = \begin{cases} 1, & if\ t\ is\ True, \\ 0, & if\ t\ is\ False, \end{cases} \end{gathered}, \tag{12}$$

where m_v is the total of vertexes on the cap mesh, $\mathbf{v}_r$ is the real initialization coordinates of the vertex, and $\mathbf{v}_t$ is the expected target coordinates of the vertex by simulating the cap that is correctly worn on the head. τ is the tuning parameter.

The **Average Distance of Points** (ADP) is defined as:

$$ADP = \frac{\sum \mathbb{I}\left(\frac{1}{m_v}\sum|\mathbf{v}_r - \mathbf{v}_t| < \tau\right)}{T_{num}} \sum \frac{1}{m_v} \sum |\mathbf{v}_r - \mathbf{v}_t| . \tag{13}$$

Metrics	CD↓ [5]	EMD↓ [30]
Ours	0.0485	0.0312

Table 4. We evaluate the Nocs mesh reconstruction performance of the NSE module with the Chamfer Distance (CD) [5] and Earth Mover‘s Distance (EMD) [30] as evaluation metrics, respectively.

5.4. Performance comparison on cap wearing

Analysis of results: The performance of our proposed framework and other baselines in simulated environment and real world are shown in Table 1 and Table 2 respectively. From the presented result, we can obtain the following observations: 1) our proposed method and two other variations outperforms all other baselines in terms of all metrics by $3\% \sim 20\%$, which denotes that our proposed framework could learn manipulation via only one demonstration and generalize the learned skill to novel instances well. 2) Although some methods achieve a relative higher performance in terms of the GSR, they show a low performance on WSR. This is because compared with grasping, placing cap on the head is more difficult and requires more precise manipulation. 3) Our proposed framework outperforms other methods in terms of APD, which means that our framework can not only wear the cap on the head successfully, but also in the desired pose.

Analysis of other baselines: In experiments, 1) We observe that DON and NDF often fail when the grasped area deforms significantly. This is because the features struggle to match under large deformation. 2) Similar to our method, Garmentnets deform each instance to its own canonical space to transfer grasp pose. However, it can only apply the same pose to manipulate different caps, ignoring the intra-class shape diversity, leading to the drop of WSR. On the contrary, we develop NSE module to handle shape diversity by learning class-general features to infer specialized grasp pose for each instance.

	Grasp Success Rate	Place Success Rate
Simulated env	0.95	0.93
Real World	0.89	0.87

Table 5. Results of cap hanging task. 100 attempts on 10 caps.

5.5. Evaluation on Soft part manipulation

We leverage cap wearing task to show that our framework could learn to manipulate similar deformable objects via only one demonstration. Here, we further design a hanging task with caps where the robot grasps the soft part of the target, as shown in Fig. 7 5) 6). The hanging task shows that 1) our framework could manipulate the soft part of the deformable objects and 2) our framework could generalize well to different manipulation task. From Tab. 5, we can observe that our method also achieves good performance with only one demonstration, which verifies the soft part manipulation ability and the generalizaiton ability.

5.6. Effectiveness Analysis

Nocs state prediction. We provide the accuracy of the Nocs coordinates predicted by NST module in Tab. 3. The accuracy of the predicted nocs coordinates Acc_{nocs} for each instance is calculated as:

$$Acc_{nocs} = \frac{1}{n_p} \sum \left| \mathbf{x}_{pre} - \mathbf{x}_{gth} \right|, \tag{14}$$

where n_p is the point number in an observed point cloud $\mathbf{P}$; $\mathbf{x}_{pre}$ and $\mathbf{x}_{gth}$ are the predicted result and the ground-truth Nocs coordinates respectively. We also provide several visualization results of the Nocs coordinates predicted by NTS module in Fig. 5. From Tab. 3 and Fig. 5, we find the NST module can predict the Nocs coordinates of the most points in $\mathbf{P}$ correctly.

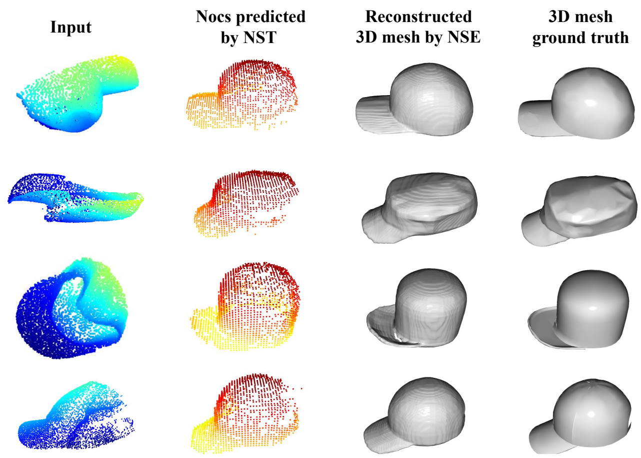
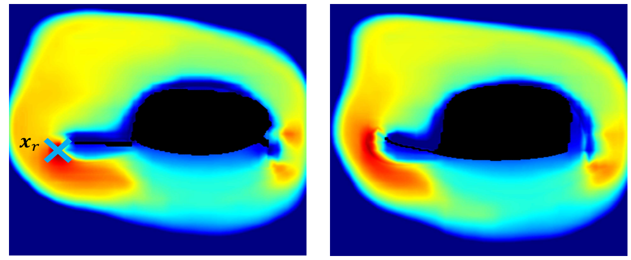


Figure 5. Visualization of the input point clouds, the Nocs coordinates predicted by the NST, the Nocs meshes reconstructed by the NSE, and the ground-truth Nocs mesh. Intuitively, NST can accurately align the partial inputs with various large deformations into Nocs, and NSE can further reconstruct high-quality mesh models under Nocs state.

Shape completion. As stated in Sec. 3.4, the NSE module learns to extract the neural spatial features via shape completion. Here, we provide the Nocs mesh reconstruction result in Fig. 5 and Tab. 4. The presented results show that NSE can exactly reconstruct the Nocs mesh model of the deformable object, which means NSE indeed has the ability to complete the whole shape from partial point clouds.

Spatial relationship encoding. Our method generalizes the demonstrated pose to novel objects via feature matching. To accurately estimate the manipulation poses, the features learned by NSE should be able to generalize to the same parts on novel objects, while rejecting wrong grasp points at different parts. To demonstrate such ability, we construct two energy fields, and visualize them in Fig. 6a, and Fig. 6b. Specifically, E_1 measures the similarity between the feature of the grasping point x_r and other points in the Nocs cube of the demonstrated object $\mathbf{P}_d$: $E_1(\mathbf{x}, \mathbf{x}_r) = ||\Phi(\mathbf{x}|\mathbf{P}_d) - \Phi(\mathbf{x_r}|\mathbf{P}_d)||$. E_2 measures the similarity between the feature of x_r and the features in the Nocs cube of a novel object $\mathbf{P}_n$: $E_2(\mathbf{x}, \mathbf{x}_r) = ||\Phi(\mathbf{x}|\mathbf{P}_n) - \Phi(\mathbf{x_r}|\mathbf{P}_d)||$. The visualization results show that the feature at x_r can be accurately transferred to the novel objects.



(a) The visualization result of E_1 (b) The visualization result of E_2

Figure 6. The visualization results of $E1$ and $E2$ show that the NSE module has the ability to describe the spatial information of coordinates in Nocs state.

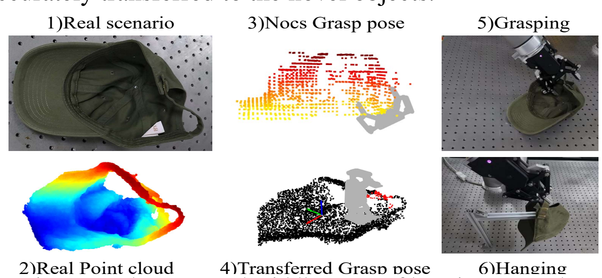


Figure 7. The steps and middle states of hanging cap task.

5.7. Ablation study

We conduct ablation studies by ablating $\mathcal{L}_{const}$ (Ours-w/o $\mathcal{L}_{const}$) and the NSE module (Ours-w/o NSE). For the variant without the NSE module, we leverage the similarity of the features to search the key points of different manipulation steps in the novel object and transfer them to the original point cloud to get the gripper poses. From the results presented in Tab. 1, we can observe that the performances decrease about $2\% \sim 4\%$ after removing any components of our method, which justifies the effectiveness of each strategy and module.

6. Conclusions

In this paper, we propose a deformable 3D object manipulation framework with the NST and NSE modules, which can learn to manipulate similar non-rigid/deformable objects via only one robot demonstration and achieve learned skills generalization from known instances to novel similar instances smoothly without re-training. Based on our proposed framework, a new simulated dataset $\mathbf{Cap_{40}}$ is collected and annotated, and a real robotic system is built to achieve cap wearing automatically as well. Both simulated results and real-world experiments justify the effectiveness of our framework and robotic system. Actually, the cap wearing is just a simple case for deformable 3D objects manipulation, our idea could be extended to more general and complex cases in future.

References

[1] David B. Adrian, Andras Gabor Kupcsik, Markus Spies, and Heiko Neumann. Efficient and robust training of dense object nets for multi-object robot manipulation. In *2022 International Conference on Robotics and Automation, ICRA 2022*, pages 1562–1568. IEEE, 2022. 1

[2] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. In *International Conference on Intelligent Robots and Systems, IROS 2022*. 1

[3] Gavin Barill, Neil G. Dickson, Ryan M. Schmidt, David I. W. Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Trans. Graph.*, 37(4):43, 2018. 4

[4] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs:divergence guided shape implicit neural representation for unoriented point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19301–19310. IEEE, 2022. 2

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 7

[6] Ronghan Chen, Yang Cong, and Jiahua Dong. Unsupervised dense deformation embedding network for template-free shape correspondence. In *International Conference on Computer Vision, ICCV 2021*, pages 8341–8350. IEEE. 3

[7] Cheng Chi and Shuran Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3324–3333, 2021. 2, 3, 4, 6, 7

[8] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 4, 6

[9] Yang Cong, Ronghan Chen, Bingtao Ma, Hongsen Liu, Dongdong Hou, and Chenguang Yang. A comprehensive study of 3-d vision-based robot manipulation. *IEEE Trans. Cybern.*, 53(3):1682–1698, 2023. 1

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021. 6

[11] Jiahua Dong, Yang Cong, Gan Sun, Lixu Wang, Lingjuan Lyu, Jun Li, and Ender Konukoglu. Inor-net: Incremental 3-d object recognition network for point cloud representation. *IEEE Transactions on Neural Networks and Learning Systems*, 1(1):1–13, 2023. 1

[12] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4497–4506. IEEE, 2021. 1, 2

[13] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *CoRL*, volume 87 of *Proceedings of Machine Learning Research*, pages 373–385. PMLR, 2018. 2, 6, 7

[14] Kartik Gupta, Darius Burschka, and Arnav Bhavsar. Effectiveness of grasp attributes and motion-constraints for fine-grained recognition of object manipulation actions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016*, pages 1232–1239. IEEE Computer Society, 2016. 2

[15] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8565–8574. Computer Vision Foundation / IEEE, 2019. 2

[16] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.*, 32(4):33:1–33:12, 2013. 4, 6

[17] Ariel Kapusta, Zackory Erickson, Henry M Clever, Wenhao Yu, C Karen Liu, Greg Turk, and Charles C Kemp. Personalized collaborative plans for robot-assisted dressing via optimization and simulation. *Autonomous Robots*, 43(8):2183–2207, 2019. 2

[18] Ninad Khargonkar, Neil Song, Zesheng Xu, Balakrishnan Prabhakaran, and Yu Xiang. Neuralgrasps: Learning implicit representations for grasps of multiple robotic hands. *arXiv preprint arXiv:2207.02959*, 2022. 2

[19] Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 1919–1928. IEEE, 2022. 2

[20] Mona Lisa and Hew Bot. My Research Software, 12 2017. 5

[21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 4

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[23] Mark Moll and Lydia E Kavraki. Path planning for deformable linear objects. *IEEE Transactions on Robotics*, 22(4):625–636, 2006. 2

[24] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. 2

[25] David R. Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. Deepcurrents: Learning implicit representations of shapes with boundaries. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 18644–18654. IEEE, 2022. 2

[26] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body:

Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 9054–9063. IEEE, 2021. 2

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3, 6

[28] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition,CVPR 2022*, pages 5163–5173. IEEE, 2022. 2

[29] Corban G. Rivera, David A. Handelman, Christopher R. Ratto, David Patrone, and Bart L. Paulhamus. Visual goal-directed meta-imitation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022*, pages 3766–3772. IEEE, 2022. 2

[30] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. 7

[31] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575, 2021. 2

[32] Bokui Shen, Zhenyu Jiang, Christopher Choy, Leonidas J Guibas, Silvio Savarese, Anima Anandkumar, and Yuke Zhu. Acid: Action-conditional implicit visual dynamics for deformable object manipulation. *arXiv preprint arXiv:2203.06856*, 2022. 1, 2

[33] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400, 2022. 2, 6, 7

[34] Christiane Sommer, Lu Sang, David Schubert, and Daniel Cremers. Gradient-sdf: A semi-implicit surface representation for 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 6270–6279. IEEE, 2022. 2

[35] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. Emd net: An encode–manipulate–decode network for cloth manipulation. *IEEE Robotics and Automation Letters*, 3(3):1771–1778, 2018. 2

[36] Junke Wang, Zuxuan Wu, Jingjing Chen, Xintong Han, Abhinav Shrivastava, Ser-Nam Lim, and Yu-Gang Jiang. Objectformer for image manipulation detection and localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2354–2363. IEEE, 2022. 1

[37] Yifan Wang, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 374–383. Computer Vision Foundation / IEEE, 2021. 2

[38] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *arXiv preprint arXiv:2201.12716*, 2022. 2

[39] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning, CoRL, 2021*, volume 164, pages 192–202. 1

[40] Wikipedia contributors. Blender (software) — Wikipedia, the free encyclopedia, 2022. [Online; accessed 9-November-2022]. 6

[41] Alan Wu, A. J. Piergiovanni, and Michael S. Ryoo. Action-conditioned convolutional future regression models for robot imitation learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2035–2037. Computer Vision Foundation / IEEE Computer Society, 2018. 2

[42] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference*, volume 12348, pages 574–591. 4

[43] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, Kei Okada, and Masayuki Inaba. Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions. In *2014 IEEE/SICE International Symposium on System Integration*, pages 564–570, 2014. 2

[44] Francisco Yandún, Abhisesh Silwal, and George Kantor. Visual 3d reconstruction and dynamic simulation of fruit trees for robotic manipulation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020*, pages 238–247. Computer Vision Foundation / IEEE, 2020. 2

[45] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. GIFS: neural implicit function for general shape representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12819–12829. IEEE, 2022. 2

[46] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Sci. Robotics*, 6(54):8803, 2021. 2

[47] Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3d scene understanding from a single image with implicit representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 8833–8842. Computer Vision Foundation / IEEE, 2021. 2

[48] Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. Deep implicit templates for 3d shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1429–1439. IEEE, 2021.