Autonomous Manipulation Learning for Similar Deformable Objects via Only One Demonstration

Yu Ren† Ronghan Chen† Yang Cong‡

†State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences
‡Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences

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 Cap40 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 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

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 manipulation skill could be generalized to other novel caps without re-training.
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 of the same category, from only one demonstration. To deformed unseen task—learning to manipulate freely, these complex and time-consuming procedures limit their practical applications.

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 contrasting 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
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 geometries 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 \( \mathcal{D} = \{ \mathbf{P} \mid \{ \mathbf{T} \} \} \), 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 \( \mathbf{P}_{\text{nocs}} \) are described as follows:

**Step 1:** Given \( \mathbf{P} \in \mathbb{R}^{N \times 3} \) (\( N \) denotes the point number), we adopt the block \( \mathbf{E} \) of pointnet++ [27] to encode points in
P into the per-point features $F = \{f_i\}_{i=1}^N = E(P; \theta_E) \in \mathbb{R}^{N \times C}$, where $C$ denotes the feature channel, and $\theta_E$ is the weights of the encoder $E$.

**Step 2:** The classifier $C$ is employed to predict the Nocs state coordinate of each point in $P$ based on $\{f_i\}_{i=1}^N$. 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 $(p_x, p_y, p_z) = C(F; \theta_C)$.

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

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

$$\mathcal{L}_{nocs} = - \left( c_x \sum \log p_x + c_y \sum \log p_y + c_z \sum \log p_z \right), \tag{1}$$

where $(p_x, p_y, p_z)$ is the output of the $C$; $(c_x, c_y, c_z)$ 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:

$$\mathcal{L}_{const} = - \sum_{(i,j) \in P} \log \frac{\exp(f_i \cdot f_j / \tau)}{\sum_{(i,k) \in P} \exp(f_i \cdot f_k)}, \tag{2}$$

where $P$ is the set of point pairs that have the same Nocs coordinates, and $f$ is the feature of the corresponding point. For a matched point pair $(i, j) \in P$, we regard $f_i$ as the query feature, $f_j$ as the positive key feature, and $f_k (k \neq j)$ as the negative key feature.

### 3.4. Neural Spatial Encoding (NSE) Module

Given the per-point features $F$ and the transferred point cloud $P_{nocs}$, NSE maps arbitrary query coordinates $x$ in Nocs state space to a neural spatial feature $f_x$. 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 described as follows:

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

**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 $P$, a zero feature is initialized as the placeholder. The assigned result is denoted as $V \in \mathbb{R}^{32 \times 32 \times 32 \times (C + 3 + 3)}$.

**Step 3:** NSE feeds $V$ into the block $U$ of 3D U-Net [8] to generate dense features $D = U(V; \theta_U)$. This embeds the features with more context information.

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

$$f_x = \sum_{i=1}^M M_i(D(x; \theta_D) \oplus x; \theta_M), \tag{3}$$

where $M_i$ denotes the features generated from the $i$-th layer of $M$, $\theta_M$ is the weight parameters of the $M$. Finally, we can get an implicit encoding function by rewriting Eq.(3) as $f_x = \Phi(x|P)$. $\Phi$ describes the ability of NSE that encodes query coordinates in Nocs space to features given observed point cloud $P$.

**Training.** We train $U$ and $M$ 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 $U$ and $M$ are leveraged to predict the generalized winding number [3, 16] $w(x)$ for $x$:

$$w(x) = M(D(x; \theta_D) \oplus x; \theta_M). \tag{4}$$

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 $U$ and $M$ 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 $w(x)$ can be determined by integrating the solid angle [16] over all Nocs points at $x$. Similarly, we use the data in a simulated environment to train the parameters of $U$, $M$. 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 $M$ and ground truth to calculate a $L1$ loss as:

$$\mathcal{L}_{implicit} = \| M^i(D(x)) - w(x) \|_1, \tag{5}$$
where \( w(x) \) is the ground-truth winding number at point \( x \).

### 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 \( D = \{P|T]\} \), where \( P \) is the point cloud of a deformable target object and \( |T| \) are key observed gripper poses, can be divided into three steps:

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

\[
T_{\text{trans}} = \arg \min_T \sum_i T(x_i - x_i^{\text{nocs}}). \tag{6}
\]

Finally, we can get the \( H \) by transferring \( T \) to Nocs state:

\[
H = T_{\text{trans}} T.
\]

**Step 2:** Encoding Nocs state gripper pose \( H \) 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 \( G = \{x_i\}_{i=1}^N \). We then move the query set \( G \) to Nocs state using \( H \) as \( H(G) = HG \). Finally, We feed \( P \) into NSE and encode \( H \) by generating the implicit features of all points in \( H(G) \) using Eq. (3) as:

\[
Z(H|P) = \Phi(H|G|P). \tag{7}
\]

**Generalization phase.** Given the observed point cloud \( P_o \) of a novel cap \( o_n \), and \( Z \) learned from Eq. (7), the NST and NSE modules could generalize the manipulation skill to \( o_n \) in three steps:

**Step 1:** Transferring \( P_o \) to \( P^{\text{nocs}}_o \) (Nocs state) using the NST module.

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

\[
Z^{o_n} = \Phi(H|G|P_o) \tag{8}
\]

using the same operations in step 2 of the demonstration phase, where \( G \) is the query set used in Eq.(7). We then try to minimize \( \mathcal{L}_Z = |Z^{o_n} - Z| \) in order to optimize \( H \) iteratively and get generalized \( H \) for novel objects \( o_n \) as:

\[
H^{o_n} = \arg \min_H |\Phi(H|G|P_o) - Z|. \tag{9}
\]

### 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 \( D = \{P_{qd}, T_{pick}, T_{middle}, T_{release}\} \), where \( P_{qd}, T_{pick}, T_{middle}, T_{release} \in \mathbb{R}^{4 \times 4} \) are recorded gripper poses during task performing, the proposed framework automatically estimates \( H^{pick}, H^{middle}, H^{release} \) 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 \( H^{pick}, H^{middle}, H^{release} \) 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:

\[
\min_{\theta_E, \theta_C} \mathcal{L}_{\text{nocs}} + \mathcal{L}_{\text{const}},
\]

\[
\min_{\theta_U, \theta_D} \mathcal{L}_{\text{implicit}}, \tag{10}
\]

where the \( \mathcal{L}_{\text{nocs}} \) and \( \mathcal{L}_{\text{const}} \) are adopted to enable Encoder \( E \) and Classifier \( C \) to predict Nocs coordinates for points under initial state correctly, and the \( \mathcal{L}_{\text{implicit}} \) helps 3D block \( U \) and MLP \( M \) 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 \( C \) and \( M \) are composed of

![Figure 3. Some instances contained in Cap40. We totally capture 4000 scenarios for pre-training our proposed framework.](image)
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 other methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>$GSR$ (↑)</th>
<th>$WSR_{τ=0.1d}$ (↑)</th>
<th>$ADP_{τ=0.1d}$ (↓)</th>
<th>$WSR_{τ=0.2d}$ (↑)</th>
<th>$ADP_{τ=0.2d}$ (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DON [13]</td>
<td>55.6</td>
<td>18.0</td>
<td>0.061</td>
<td>25.12</td>
<td>0.125</td>
</tr>
<tr>
<td>NDF [33]</td>
<td>87.5</td>
<td>25.0</td>
<td>0.057</td>
<td>27.31</td>
<td>0.113</td>
</tr>
<tr>
<td>Garmentnet [7]</td>
<td>90.3</td>
<td>43.6</td>
<td>0.031</td>
<td>65.16</td>
<td>0.043</td>
</tr>
<tr>
<td>Ours-w/o $L_{const}$</td>
<td>98.2</td>
<td>53.4</td>
<td>0.033</td>
<td>83.17</td>
<td>0.048</td>
</tr>
<tr>
<td>Ours-w/o NSE</td>
<td>98.0</td>
<td>51.0</td>
<td>0.034</td>
<td>82.14</td>
<td>0.047</td>
</tr>
<tr>
<td>Ours</td>
<td>98.5</td>
<td>55.0</td>
<td>0.030</td>
<td>86.55</td>
<td>0.041</td>
</tr>
</tbody>
</table>

4. Simulated data generation

A simulated cap dataset 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.

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 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, and a Realsense 435i depth camera. We evaluate the success rate with different random initialization.

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
where \( m \) the real initialization coordinates of the vertex, and \( v \) is the predicted by NST module. 

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Grasp Success Rate</th>
<th>Wear Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DON [13]</td>
<td>0.31</td>
<td>0.11</td>
</tr>
<tr>
<td>NDF [33]</td>
<td>0.64</td>
<td>0.12</td>
</tr>
<tr>
<td>Garmennet [7]</td>
<td>0.72</td>
<td>0.44</td>
</tr>
<tr>
<td>Ours-w/o ( L_{const} )</td>
<td>0.85</td>
<td>0.76</td>
</tr>
<tr>
<td>Ours-w/o NSE</td>
<td>0.84</td>
<td>0.72</td>
</tr>
<tr>
<td>Ours</td>
<td>0.85</td>
<td>0.78</td>
</tr>
</tbody>
</table>

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

\[
\text{Grasp Success Rate (GSR)} = \frac{GSR_{\text{num}}}{T_{\text{num}}},
\]

where \( GSR_{\text{num}} \) is the successful times of the robot grasp caps in the test phase; and \( T_{\text{num}} \) is the total test number.

The \textbf{Wearing Success Rate} (\( WSR \)) is designed as:

\[
WSR = \frac{\sum_{v_t} \left( \frac{1}{m_v} \sum |v_r - v_t| < \tau \right)}{T_{\text{num}}}.
\]

where \( m_v \) is the total of vertexes on the cap mesh, \( v_r \) is the real initialization coordinates of the vertex, and \( v_t \) is the expected target coordinates of the vertex by simulating the cap that is correctly worn on the head. \( \tau \) is the tuning parameter.

The \textbf{Average Distance of Points} (\( ADP \)) is defined as:

\[
ADP = \frac{\sum_{v_t} \left( \frac{1}{m_v} \sum |v_r - v_t| < \tau \right)}{T_{\text{num}}} \sum_{m_v} \frac{1}{m_v} \sum |v_r - v_t|.
\]

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 \( ADP \), 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.

### 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 $\text{Acc}_{\text{nocs}}$ for each instance is calculated as:

$$\text{Acc}_{\text{nocs}} = 1 - \frac{1}{n_p} \sum |x_{\text{pre}} - x_{\text{gth}}|,$$

where $n_p$ is the point number in an observed point cloud $P$; $x_{\text{pre}}$ and $x_{\text{gth}}$ 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 $P$ 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.](image1)

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 $P_d$: $E_1(x, x_r) = ||\Phi(x|P_d) - \Phi(x_r|P_d)||$. $E_2$ measures the similarity between the feature of $x_r$ and the features in the Nocs cube of a novel object $P_n$: $E_2(x, x_r) = ||\Phi(x|P_n) - \Phi(x_r|P_d)||$. The visualization results show that the feature at $x_r$ can be accurately transferred to the novel objects.

![Figure 6. The visualization results of $E_1$ and $E_2$ show that the NSE module has the ability to describe the spatial information of coordinates in Nocs state.](image2)

5.7. Ablation study

We conduct ablation studies by ablating $\mathcal{L}_{\text{const}}$ (Ours-w/o $\mathcal{L}_{\text{const}}$) 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% ~ 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 Cap40 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


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


