ABSTRACT

We introduce a novel approach for manipulating articulated objects with the ability to generalize across object translations, scaling, and robot kinematics. In traditional imitation learning scenarios, the learned policy fails if camera perspectives or the relative positions of objects change during inference, and is specific to certain robot configurations. To tackle these challenges, we propose an object-centric representation for both observations of object motions and robot actions, which decouples vision-based decision making from robot execution. The center here refers to a canonical frame of the articulated object that remains invariant regardless of the contact point. We further use object-centric imitation learning to make stable predictions under varying camera views, object placements, and robot kinematics. Experiments and analysis demonstrate that our method achieves state-of-the-art performance in articulated object manipulation and dramatically improves generalization performance.

KEY WORDS： imitation learning, object-centric, robot manipulation

CONTENTS

1 Introduction

1.1 Motivation

The core problem in robot imitation learning is the strong dependency between a robot' s learned policy and its specific configuration, camera perspectives, and the relative positions of objects. This makes it difficult to generalize the robot' s learned skills to new situations. This problem is crucial in enabling robots to perform tasks in real-world environments. For example, a service robot may need to open doors of varying sizes and placements within its environment, and we want the policy to work cross embodiment. Coupling learning of visual perception and robot control makes generated action highly overfit to noise and task-irrelevant inputs. By developing methods for object-centric learning, we decouple observed object motions and robot actions to make our policy generalize over various object translation, scaling and robot configurations. The current state-of-the-art robot learning methods often rely on pixel-based visual inputs and trajectories anchored in the robot's base frame. However, we want to decouple vision-based decision making and robot execution. Inspired by the grasp detection algorithm, we want to "detect" object-centric executable trajectories for door opening. Object-centric means the trajectories are attached to the object itself, irrelevant to specific robot configurations, and we study observations of object motions irrelevant to the object' s placements or camera views.

1.2 Literature Review

1.2.1 Articulated Object Manipulation

Manipulating articulated objects is challenging, with some approaches using analytical methods [2–6] and data-driven learning-based methods [7, 8], including techniques to model manipulation with explicit parameters [9–12], visual affordances [7, 13, 14] and articulation flows[1, 15]. Different from previous learning-based works which tend to learn the representation through a pure object model, we incorporate contact point concept and leverage the point for prediction.

1.2.2 Learning from Human Demonstrations

Recent studies focus on imitation learning including utilizing Transformer architectures [16–18, 20] to predict robot actions. To break the limitations of 2D observations, the 3D policy models C2F-ARM [19] and PerAct [16] voxelize the robot's workspace and are trained to identify the specific 3D voxel that contains the next key pose for the end effector. ACT3D

[24] represents the scene as a continuous resolution 3D feature field rather than voxel. Another method is to apply diffusion models to robotic applications. Chi et al. [25] introduced the groundbreaking Diffusion Policy (DP), while 3D-DP [26] learns to predict full trajectories by employing a modified MLP as an efficient backbone for processing point clouds. A combination can be seen in ChainedDiffuser [27], which first predicts high-level key points using ACT3D, then generates low-level trajectories that connect these key points through standard trajectory diffusion techniques. We show in our paper that the actions can be directly generated taking point cloud and oracle robot states as input based on our object-centric representation.

1.2.3 Object-Centric Representation for Manipulation

Several works focus on extracting object features and transforming observation frames to guide robot actions. [29] introduces an affordance-centric coordinate frame for representing robot actions in long-horizon, multi-object tasks, while [30] detects grasp points from depth recordings with a novel grasp representation. [31] mines multiple coordinate frames by combining transformed point cloud data to propose actions.

1.3 Main Work

We consider the problem of enabling a robot to imitate a door-opening task from a limited number of expert demonstration videos. The goal is to develop a policy that generalizes to unseen object placements, camera viewpoints, and robot configurations that are out of distribution.

Each task instance involves a robot equipped with a parallel-jaw gripper interacting with a target object, such as a door. The task is defined by a trajectory of object poses over time. During the demonstration phase, the robot performs the door-opening task while a stationary camera records a sequence of observations $o_t = \{(P_t, \bar{o}_t)\}_{t=1}^{N}$, where N is the task horizon. Here, $P_t \in \mathbb{R}^{N_p \times 3}$ denotes the observed object point cloud, and $\bar{o}_t$ is the robot's oracle state. At inference time, given a robot observation o_t, the objective is to learn a policy π that maps o_t to an end-effector trajectory $a_{t:t+k} = (p_{t:t+k}, R_{t:t+k}, w_{t:t+k})$, where $p_{t:t+k} \in \mathbb{R}^3$ and $R_{t:t+k} \in SO(3)$ denote the position and orientation of the end-effector, and $w_{t:t+k} \in \mathbb{R}$ is the gripper width. The goal is to maximize the likelihood of successful task execution.

Assumptions. During implementation, we make the following assumptions: (1) Visual observations are captured from a single view camera and do not include action labels. (2) The robot has no prior knowledge of object geometry, task dynamics, or 3D models. (3) All doors are modeled as rigid bodies with a handle attached to the surface and can be opened by directly pulling or pushing using the designated gripper.

2 Theoretical Overview

2.1 Action Chunking With Transformers

2.1.1 conditional variational autoencoder (CVAE)

The CVAE encoder and decoder is implemented with transformers, as transformers are designed for both synthesizing information across a sequence and generating new sequences. (1)The encoder of the CVAE

The CVAE encoder is implemented with a BERT-like transformer encoder. The inputs to the encoder are the current joint positions and the target action sequence of length k from the demonstration dataset, prepended by a learned ”[CLS]” token similar to BERT. The output is the mean and variance of the ”style variable” z, which is then used as input to the decoder. (2)The decoder of the CVAE

We use ResNet image encoders, a transformer encoder, and a transformer decoder to implement the CVAE decoder. the transformer encoder synthesizes information from different camera viewpoints, the joint positions, and the style variable, and the transformer decoder generates a coherent action sequence. The whole model is trained to maximize the log-likelihood of demonstration action chunks, i.e.

$$\min_{\theta} - \sum_{s_t, a_{t:t+k} \in D} \log \pi_\theta(a_{t:t+k} \mid s_t), \tag{2-1}$$

where D is the demonstration dataset, s_t is the input state at time t, and $a_{t:t+k}$ is the target action sequence of length k starting at time t. with the standard VAE objective which has two terms: a reconstruction loss and a term that regularizes the encoder to a Gaussian prior.

2.1.2 Action Chunking and Temporal Ensemble

To mitigate the compounding errors in imitation learning while maintaining compatibility with pixel-to-action policies, we adopt an action chunking strategy that reduces the effective horizon of long trajectories collected at high frequency. Instead of modeling individual action distributions, the policy models a distribution over action chunks, i.e., $\pi_\theta(a_{t:t+k} \mid s_t)$, leading to a k-fold reduction in the effective horizon. In our implementation, the agent generates and executes k actions sequentially at every k steps after receiving an observation.

To further stabilize action execution, we employ a temporal ensemble strategy that performs a weighted average over predicted actions using an exponential weighting scheme: $w_i = \exp(-m \cdot i)$,where w_0 corresponds to the oldest action. The rate at which new observations influence action selection is controlled by m, with smaller values allowing faster incorporation

of new information[22].

2.1.3 Algorithm Workflow

The observation space consists of four RGB images, each with a resolution of 480×640, along with joint positions for two robotic arms, totaling 14 degrees of freedom (DoF). The action space is represented as a 14-dimensional vector containing absolute joint positions for both robots. With action chunking, the policy outputs a $k \times 14$ tensor given the current observation.

For visual feature extraction, we employ ResNet18 [21] as the backbone, processing $480\times 640\times 3$ RGB images into feature maps of size $15\times 20\times 512$. These feature maps are then flattened along the spatial dimension to obtain a sequence of 300×512. To retain spatial structure, a 2D sinusoidal positional embedding is added to each feature sequence. Repeating this for all four images results in a final visual feature sequence of dimensions 1200×512.

In addition to image features, we append two additional inputs: the current joint positions and a learned ”style variable” z. Both are projected to 512-dimensional embeddings using linear layers, yielding a complete input representation of size 1202×512 for the transformer encoder.

The transformer decoder conditions on the encoder output via cross-attention. Its input consists of a fixed positional embedding of size $k\times 512$, while keys and values are derived from the encoder outputs. The decoder generates an output of size $k \times 512$, which is subsequently down-projected via an MLP into a $k \times 14$ matrix, corresponding to the predicted target joint positions for the next k steps. For training, we use L1 loss instead of L2 loss, as we find that L1 loss results in more precise action sequence modeling.

2.2 3D Diffusion Policy

2.2.1 Diffusion Policy

Diffusion policy is a new form of robot visuomotor policy that generates behavior via a ”conditional denoising diffusion process on robot action space”. In this formulation, instead of directly outputting an action, the policy infers the action-score gradient, conditioned on visual observations, for K denoising iterations. This formulation allows robot policies to inherit several key properties from diffusion models –significantly improving performance. The core design of DP3 is the utilization of a compact 3D visual representation, extracted from sparse point clouds with an efficient point encoder [26].

2.2.2 Incorporation of 3D visual representation

The generality of visual imitation learning comes at a cost of vast demonstrations. To enable (offline) imitation learning algorithms to learn robust and generalizable skills with as few demonstrations as possible, DP3 encodes sparsely sampled point clouds into a compact 3D representation using a straightforward and efficient MLP encoder. Subsequently, DP3 denoises

random noise into a coherent action sequence, conditioned on this compact 3D representation and the robot poses. This integration leverages not only the spatial understanding capabilities inherent in 3D modalities but also the expressiveness of diffusion models. This exhibits notable advantages over 2D-based diffusion policies in efficiency and generalization.

2.2.3 Algorithm Workflow

(1) Perception

3D Diffusion Policy (DP3) represents 3D scenes using sparse point clouds, as they offer greater efficiency compared to alternative explicit representations such as RGB-D images, depth maps, and voxels. In both simulation and real-world settings, depth images (84×84) are captured from a single camera and converted into point clouds using camera extrinsics and intrinsics, while omitting color information to enhance appearance generalization. To improve data quality, redundant points from surfaces like tables and the ground are removed via bounding box cropping, and the remaining points are downsampled using farthest point sampling (FPS) to ensure sufficient coverage and reduce randomness. The downsampled point cloud is then encoded into a compact 3D feature representation using a lightweight DP3 Encoder, which consists of a three-layer MLP, a max-pooling function for order-equivariant feature aggregation, and a projection head. With interleaved LayerNorm layers for training stability, the final encoded feature vector is only 64-dimensional. Ablation studies demonstrate that this simple encoder can outperform large pretrained models like PointNeXt, reinforcing the effectiveness of task-specific lightweight architectures in visuomotor control.

(2) Decision

The decision module in DP3 employs a conditional denoising diffusion model to generate actions by iteratively refining Gaussian noise while conditioning on 3D visual features and robot poses. Given an initial Gaussian noise a^K, the denoising process follows:

$$a^{k-1} = \alpha_k \left(a^k - \gamma_k \epsilon_\theta \left(a^k, k, v, q \right) \right) + \sigma_k \mathcal{N}(0, \mathrm{I}), \tag{2-2}$$

where $\mathcal{N}(0, \mathrm{I})$ is Gaussian noise, and $\alpha_k, \gamma_k, \sigma_k$ are noise scheduling parameters. The training objective minimizes the mean squared error (MSE) between the predicted and actual noise:

$$\mathcal{L} = \mathrm{MSE} \left(\epsilon^k, \epsilon_\theta \left(\bar{\alpha}_k a^0 + \bar{\beta}_k \epsilon^k, k, v, q \right) \right). \tag{2-3}$$

For training, DDIM is used as the noise scheduler with 100 diffusion timesteps, and inference is performed with 10 timesteps. The model is trained for up to 3000 epochs with a batch size of 128, ensuring efficient high-dimensional action generation.

3 Decoupling Vision and Motion: Object-Centric Representations for Enhanced Manipulation

In this work, we propose an object-centric learning framework for robotic manipulation. Our method leverages a learned contact confidence to establish a dynamic object-centric frame, within which action sequences are processed and predicted. This allows generalization over varying cameare poses and object placements. We train our policy using end-effector positions and orientations as the action space, which enables generalization across different robot configurations. The ability to generalize over camera viewpoints and object placements is illustrated in Figure 3-1.

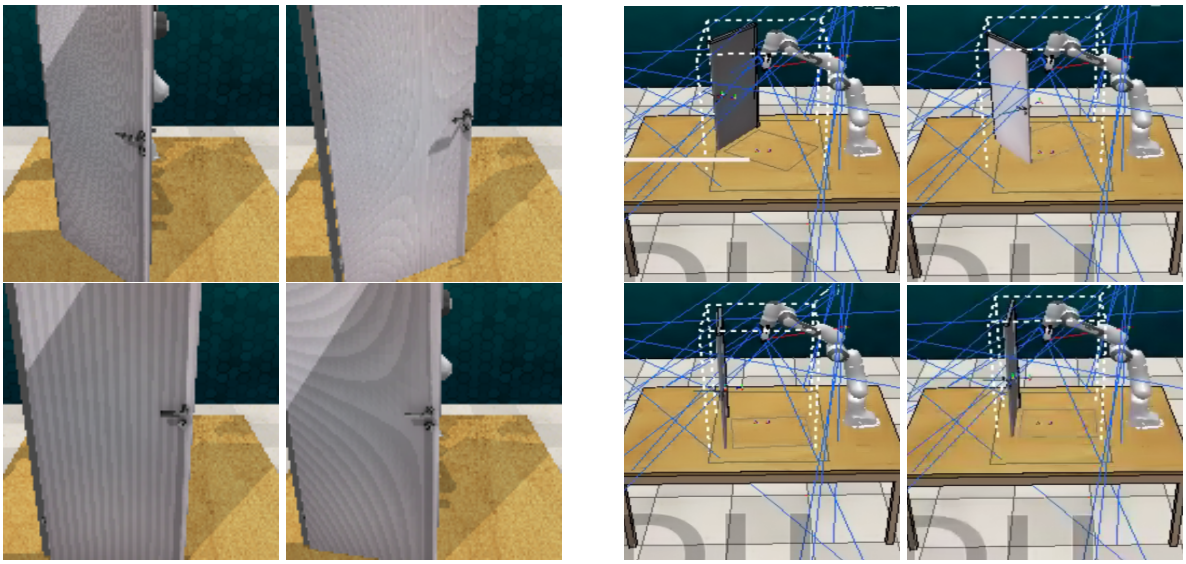
We aim to train an imitation learning policy $\pi^*(a \mid o)$ using a demonstration dataset $D = \{(o_t, a_t)\}_{t=1}^{N}$, where all observations and actions are initially expressed in the world frame $C \in SE(3)$. At each time step t, the policy takes the current observation o_t and predicts a sequence of k future actions $(a_t, \ldots, a_{t+k-1})$. To enable object-centric reasoning, we define an object-centric frame $A_t = \left({}^{A_t}_{C}R_t, {}^{A_t}_{C}T_t\right) \in SE(3)$, where the rotation ${}^{A_t}_{C}R_t \in SO(3)$ aligns with the estimated object orientation, and the translation ${}^{A_t}_{C}T_t = p^* = (x^*, y^*, z^*) \in \mathbb{R}^3$ corresponds to a detected contact point on the object. During training, we transform both the robot actions and oracle states $\bar{o}_t$ into the object-centric frame A_t to form object-centric representations. Similarly, object motion P_t is normalized with respect to the object's orientation. At inference time, A_t is computed using a PointNet-based segmentation model and an object pose estimator. The predicted object-centric actions are transformed back into the robot's base frame $B \in SE(3)$ for execution.

3.1 Object-Centric Representation for Vision-based Decision Making

We propose an object-centric representation to model the robot's observations and actions. Specifically, in the door-opening task, the robot must first establish and maintain *contact* with the door before *manipulating* its single degree-of-freedom (DoF) rotation along a fixed pivot axis. The configuration and geometric properties of the door suggest a natural definition of an object-centric coordinate frame, anchored at the contact point.

Given a cropped object point cloud $P_t \in \mathbb{R}^{N_p \times 3}$, we define a contact confidence vector $c_t \in \{0, 1\}^{N_p}$, where $c_t^i = 1$ indicates that the i-th point in P_t is a contact point. The ground truth contact point is then denoted as:

$$p^* = P_t^i, \quad \text{where } i = \{i \mid c_t^i = 1\}.$$



(a) Different camera viewpoints (b) Different object placements

Figure 3-1 **Generalization across camera viewpoints and object placements.** Left: Four examples showing variations in camera perspective. Right: The same object under different placement orientations. Our object-centric policy enables consistent action generation across these conditions.

The orientation of the object-centric frame is aligned with the geometry of the door. Specifically:

- The Z-axis is aligned with the hinge axis of the door (pivot axis).
- The X-axis is defined as perpendicular to the door surface to which the handle is attached.
- The Y-axis is computed as $Y = X \times Z$.

This object-centric frame enables consistent and transferable representations of observations and actions across varying door placements and camera views.

Perception We transform the point cloud $P_t \in \mathbb{R}^{N_p \times 3}$ to be aligned with the object's estimated rotation ${}^{A_t}_{C}R_t$ and centered at the point cloud's centroid p'_t. This transformation is defined as:

$$P'_t = {}^{A_t}_{C}R_t \cdot (P_t - p'_t) + {}^{A_t}_{C}T_t,$$

where p'_t is the centroid of P_t. This transformation ensures that the object-centric representation is invariant to the object's absolute position and camera view.

Actuation We record the oracle state $\bar{o}_t$ using the ground-truth action a_t, and transform it to the object-centric frame A_t to evaluate the robot's interaction with the object. The robot actions $a_{t:t+k} = (p_{t:t+k}, R_{t:t+k}, w_{t:t+k})$ are transformed into the object-centric frame A_t as follows:

$$\hat{p}^{A_t}_{t:t+k} = {}^{A_t}_{C}R_t \cdot (p_{t:t+k} - p^*) + {}^{A_t}_{C}T_t, \quad \hat{R}^{A_t}_{t:t+k} = {}^{A_t}_{C}R_t \cdot R_{t:t+k}, \quad \hat{w}^{A_t}_{t:t+k} = w_{t:t+k}.$$

This object-anchored representation allows the policy to remain invariant to the robot's base

frame and configuration. As a result, the learned policy generalizes across different object placements and robot embodiments by reasoning in an object-centric reference frame.

3.2 Object-centric Policy Learning

We train a seperate point cloud segmentation model to determine the most probable contact point for reference frame transformation, and a visuomotor policy for action generations. An overview of our poliy is in Figure 3-2.

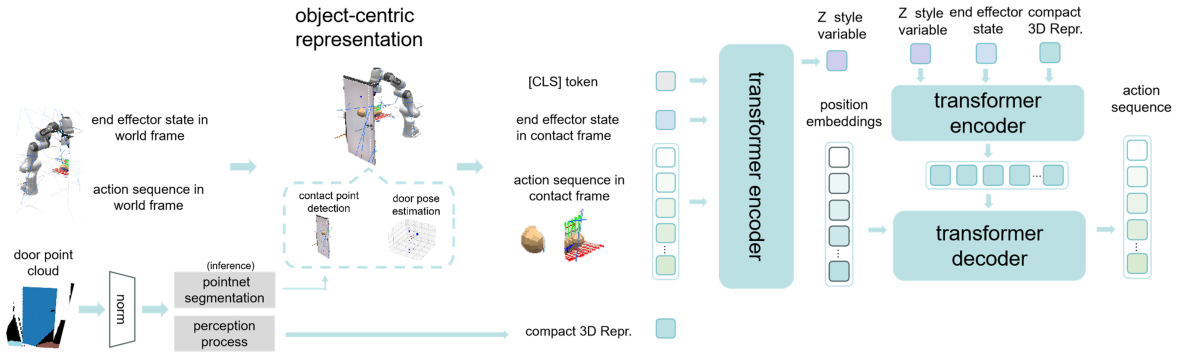


Figure 3-2 **Overview of object-centric policy.** During training, the framework jointly learns a PointNet-based segmentation model and a decision-making policy from expert demonstrations. At evaluation time, it first establishes the object-centric frame using the predicted contact point from the segmentation network, then generates actions within this frame by conditioning on the 3D point cloud representation and the robot's state, using a CVAE-based policy network.

Pointnet segmentation To dynamically determine the most probable contact point for reference frame transformation, we adopt the PointNet Segmentation Network [33] to predict the contact confidence c_t for each point in the cloud. After we normalize input point clouds to unit sphere, the network processes point clouds through a series of 1D convolutions with increasing feature dimensions ($64 \rightarrow 128 \rightarrow 512 \rightarrow 2048$), followed by hierarchical feature aggregation. The final representation is passed through log-softmax activation to produce per-point confidence scores for two classes: contact and non-contact.

The loss function combines negative log-likelihood with a regularization term that encourages orthogonality in the learned transformation matrix:

$$\mathcal{L} = \mathcal{L}_{\text{nll}}(\text{pred}, \text{target}) + \lambda \cdot \mathcal{L}_{\text{reg}}(A), \quad \text{where} \quad \mathcal{L}_{\text{reg}}(A) = \left\| I - AA^{\top} \right\|_F^2,$$

with A being the feature transformation matrix and $\lambda = 0.001$. This regularization preserves the geometric structure of input points. We train with batch size 4 and 1024 points per cloud, using mixed precision to optimize GPU memory usage.

Conditional variational autoencoder We generally follow the conditional variational autoencoder described in Section 2.1.1 [22]. The CVAE encoder only serves to train the CVAE decoder (the policy) and is discarded at test time. Specifically, the CVAE encoder predicts the mean and variance of the style variable z's distribution, which is parameterized as a diagonal

Gaussian, given the current observation, $\bar{o}_t^{A_t}$ and action sequence $a_{t:t+k}^{A_t}$ as inputs. The CVAE decoder, i.e. the policy, conditions on both z and the current observations (point cloud + end effector positions) to predict the action sequence.

Inferenece When deploying our policy, we first perform object pose estimation (Section 3.3) to obtain the normalized object point cloud. Next, we segment the normalized point cloud to identify the contact point and establish the object-centric frame A_t. This is used to transform the robot's oracle state $\bar{o}_t$ to be object-centric, which is later processed as a token to the CVAE decoder. The normalized point cloud is also passed to the perception process (Section 3.3) to form a compact 3D representation. The CVAE decoder takes the current observation and the style variable z as inputs, generating a sequence of actions in the object-centric frame. The predicted action sequence $\hat{a}_{t:t+k}^{A_t}$ is converted from the contact frame A_t back to the robot's base frame for execution：

$$\hat{a}_{t:t+k} = (\hat{p}_{t:t+k}, \hat{R}_{t:t+k}, w_{t:t+k})$$

$$\hat{p}_{t:t+k} = \left({}^{A_t}_{C}R_t\right)^{\top} \cdot \hat{p}_{t:t+k}^{A_t} + p^*$$

$$\hat{R}_{t:t+k} = \left({}^{A_t}_{C}R_t\right)^{\top} \cdot \hat{R}_{t:t+k}^{A_t}$$

The process can be viewed in Figure 3-3.

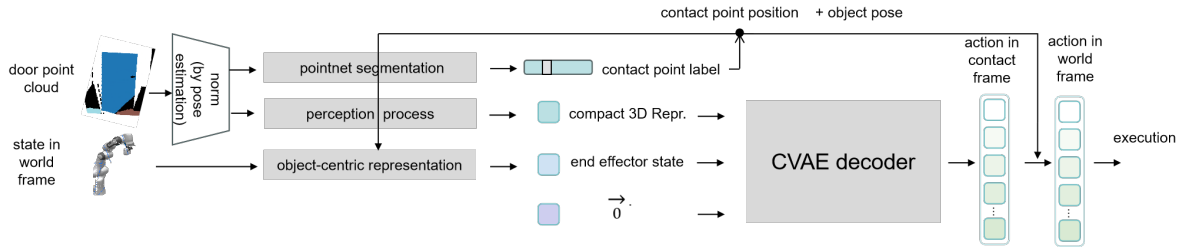


Figure 3-3 **The pipeline of the inference process.** The predicted contact point and estimated object pose are used both to construct the object-centric representation for action generation and to transform the predicted actions back to the world frame for execution.

算法 3.1 Object-centric Policy Training

Given: Demo dataset $\mathcal{D} = \{(o_t, a_t)\}_{t=1}^{N}$, weight β.
Initialize Point segmentation model f_ψ with parameters ψ.
Initialize encoder $q_\phi(z|a_{t:t+k}^{A_t}, \bar{o}_t^{A_t})$.
Initialize decoder $\pi_\theta(\hat{a}_{t:t+k}^{A_t}|o_t^{A_t}, z)$
for iteration $n = 1, 2, \ldots$ **do**
    Sample $P_t, c_t, \bar{o}_t, a_{t:t+k}$ from $\mathcal{D}$.
    Establish A_t using ${}_C^{A_t}R_t$ and contact point p^*.
    Obtain object-centric representations $a_{t:t+k}^{A_t}, \bar{o}_t^{A_t}$ and normalize P_t by ${}_C^{A_t}R_t$.
    Compute per-point logits: $\hat{c}_t = f_\psi(P_t)$
    Compute contact confidence loss:

$$\mathcal{L}_{\text{seg}} = \text{CE}(\hat{c}_t, c_t)$$

    Backpropagate and update ψ using optimizer
    Sample z from $q_\phi(z|a_{t:t+k}^{A_t}, \bar{o}_t^{A_t})$.
    Encoder P_t to obtain compact 3D representation.
    Predict $\hat{a}_{t:t+k}^{A_t}$ from $\pi_\theta(\hat{a}_{t:t+k}^{A_t}|o_t^{A_t}, z)$.
    Compute action reconstruction loss and cross-entropy loss:

$$\mathcal{L}_{\text{reconst}} = \lambda_p \cdot \text{MSE}(\hat{p}_{t:t+k}, p_{t:t+k}) + \lambda_r \cdot \text{MSE}(\hat{R}_{t:t+k}, R_{t:t+k}) + \lambda_w \cdot \text{MSE}(\hat{w}_{t:t+k}, w_{t:t+k})$$

Here:λ_p, λ_r, and λ_w are the scaling coefficients for position, rotation, and gripper width, respectively. These coefficients are determined by ensuring that the corresponding loss components are on the same scale when training reaches one-tenth of the total training steps.

    Compute latent regularization loss:

$$\mathcal{L}_{\text{reg}} = D_{KL}(q_\phi(z|a_{t:t+k}^{A_t}, \bar{o}_t) \parallel \mathcal{N}(0, I))$$

    Update parameters using ADAM:

$$\mathcal{L} = \mathcal{L}_{\text{reconst}} + \beta\mathcal{L}_{\text{reg}}$$

end for

算法 3.2 Object-centric Policy Inferenece

Given: trained policy π_θ, episode length T, chunk size k.
Initialize FIFO buffers $\mathcal{B}[0:T]$, where $\mathcal{B}[t]$ stores actions predicted for timestep t.
for timestep $t = 1, 2, \ldots, T$ **do**
  if $t \mod k == 0$ **then**
    Given P_t, perform pose estimation to obtain ${}^{A_t}_{C}R_t$, which is used to normalize P_t.
    Compute predicted contact point $p^* = P_t^j \in P_t$:

$$j = \arg\max_j \hat{c}_t[j, 1], \quad \text{where } \hat{c}_t = \text{softmax}(f_\psi(P_t))$$

    Obtain object-centric observation $o_t^{A_t}$ based on ${}^{A_t}_{C}R_t$ and p^*.
    Encode P_t to obtain compact 3D representation.
    Predict the next k actions $\hat{a}_{t:t+k}^{A_t}$ from:

$$\pi_\theta(\hat{a}_{t:t+k}^{A_t} \mid o_t^{A_t}, z), \quad \text{where } z = 0$$

    Transform predicted actions back to base frame, denoted as $\hat{a}_{t:t+k}$.
    Store the predicted actions in the buffer:

$$\mathcal{B}[t : t+k] = \hat{a}_{t:t+k}$$

  end if
  Obtain the current step action $a_t = \mathcal{B}[t]$.
  Execute action a_t.
end for

3.3 Contact-Aware Implementation

The key of our design is incorporating point cloud observations to construct an object-centric representation and condition action generation on 3D visual features. Therefore, besides point segmentation to identify the contact point, we also need to perform object pose estimation to obtain the ${}^{A_t}_{C}R_t$ and encode the point cloud into a compact 3D representation.

Pose estimation In the door-opening task, we compute the oriented bounding box (OBB) of the door plane using Open3D's built-in method. The estimated OBB is then rotated to align with the desired coordinate frame as defined in Section 3.1.

Perception process We encode point clouds into compact 3D representations with a lightweight MLP network as shown in Figure 3-4, following a method proposed in [26]. This network, called the DP3 Encoder, employs a straightforward design: it combines a three-layer MLP with a max-pooling operation that preserves order-equivariance, followed by a projection head that compresses these features into a compact vector. To enhance training stability, we incorporate LayerNorm layers [32] between components. The resulting 3D feature vector (v) is remarkably efficient, containing just 64 dimensions.

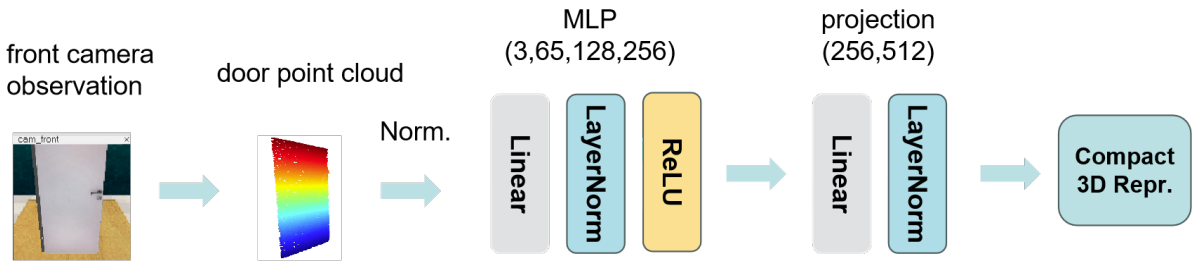


Figure 3-4 **Perception process of point clouds.**At each time step, the cropped object point cloud is transformed into a compact 3D representation, which serves as input to condition the decision-making process.

4 Experiments

4.1 Experimental Setup

Data Collection We collect 40 expert demonstrations per task for opening a door using a simulated Panda arm. In our simulation environment, each task comprises one or more variations, and each variation can generate an infinite number of episodes. In order to test the ability to generalize, we collect multiple variations in object position and camera viewpoint, and these variations are out-of-distribution (OOD) during inference. The benchmark environment is built using the V-REP simulator and the PyRep interface [34].

Baselines The primary focus of this work is to highlight the importance of object-centric representations in robotic manipulation. We compare our method against two baselines: (1) an rgbd-based transformer behavior cloning policy (ACT) [22], and (2) a 3d diffusion policy (DP3) [25]. The different vision modalities are visualized in Figure 4-1. Herein, ACT employs RGBD, while 3D diffusion utilizes the scene point cloud, which is uncropped. Our method uses the cropped point cloud that solely encompasses the object (door).

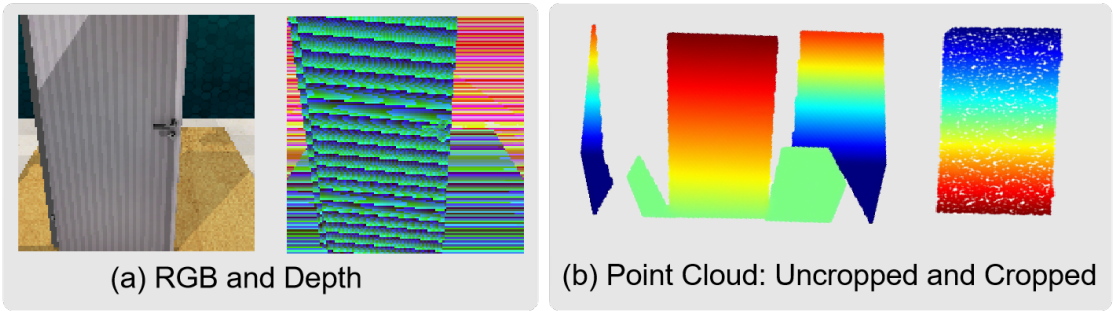


Figure 4-1 **Different vision modalities in the simulation,** include images, depths, and point clouds.

Evaluation Metrics We conducted each experiment using three different random seeds (0, 1, 2). For each seed, we evaluated performance over 40 episodes. The success rates are reported across unseen object displacements and the resulting different camera views. The variations—which correspond to the door's spatial position—are categorized into four types, denoted by numbers and colors as shown in Figure 4-2. If the door is opened to an angle equal to or larger than 25 degrees within episode length T steps, we will consider the task as a success.

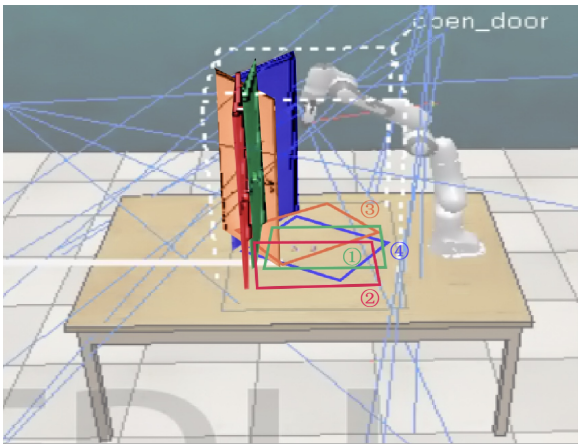


Figure 4-2 **Variations of the object positions.** The training data episodes are generated based on variation ①, while we evaluate our policy's generalization capability on the test positions (variations ②, ③, ④).

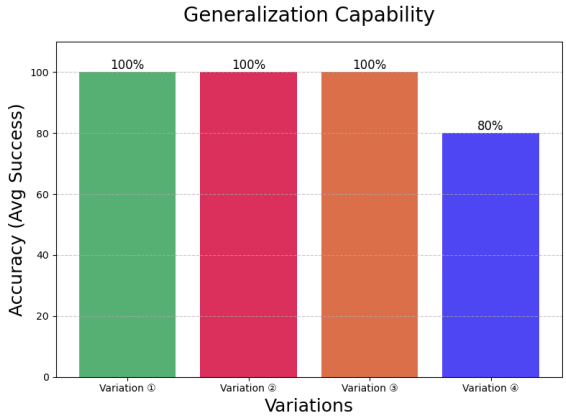


Figure 4-3 **Average success rates of generalization.** We show the average success rates for opening doors in four different position variations. This indicates that our approach demonstrates strong generalization ability in most cases.

4.2 Effectiveness and Efficiency

Generalization Accuracy We Compare our method with more baselines in simulation and the results are shown in Table 4-1. Figure 4-3 shows the detailed generalization results. Our method achieved a 100% success rate for variations 1, 2, and 3, while for variation 4, the success rate was 80%. We will discuss this in the TODO Result Analysis. Our method demonstrates strong generalization to unseen door placements, despite being trained solely on the variation marked in green in Figure 4-2. The prediction results under 4 variations of our method are visualized in Figure 4-4. The task progress of the spatial generalization results are shown in Figure 4-5.

Table 4-1 Spatial Generalization for Different Policies

Spatial Generalization	①	②	③	④	**average (%)**
ACT	✓	✗	✗	✗	30.0
DP3	✓	✗	✗	✗	15.0
Ours (Object-Centric)	✓	✓	✓	✓	95.0

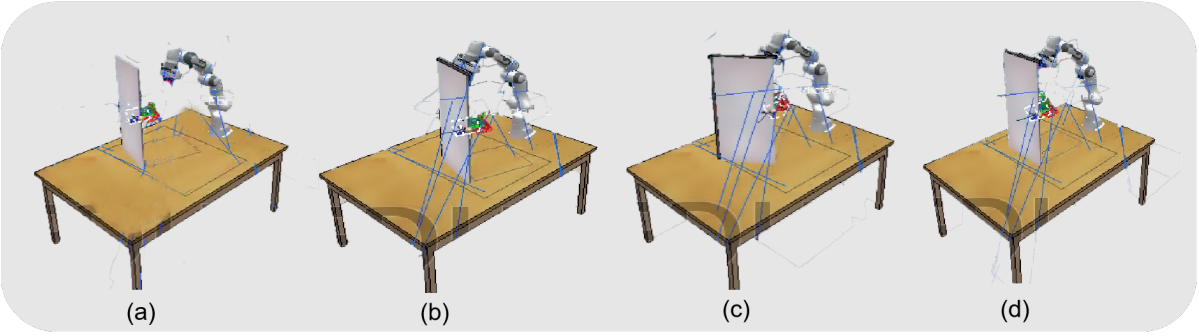


Figure 4-4 **Qualitative predictions of the robot trajectory and contact point.** We visualize predicted actions from our policy under four variations of object displacements and camera viewpoints. Each frame shows the predicted contact location and end-effector trajectory overlaid on the scene.

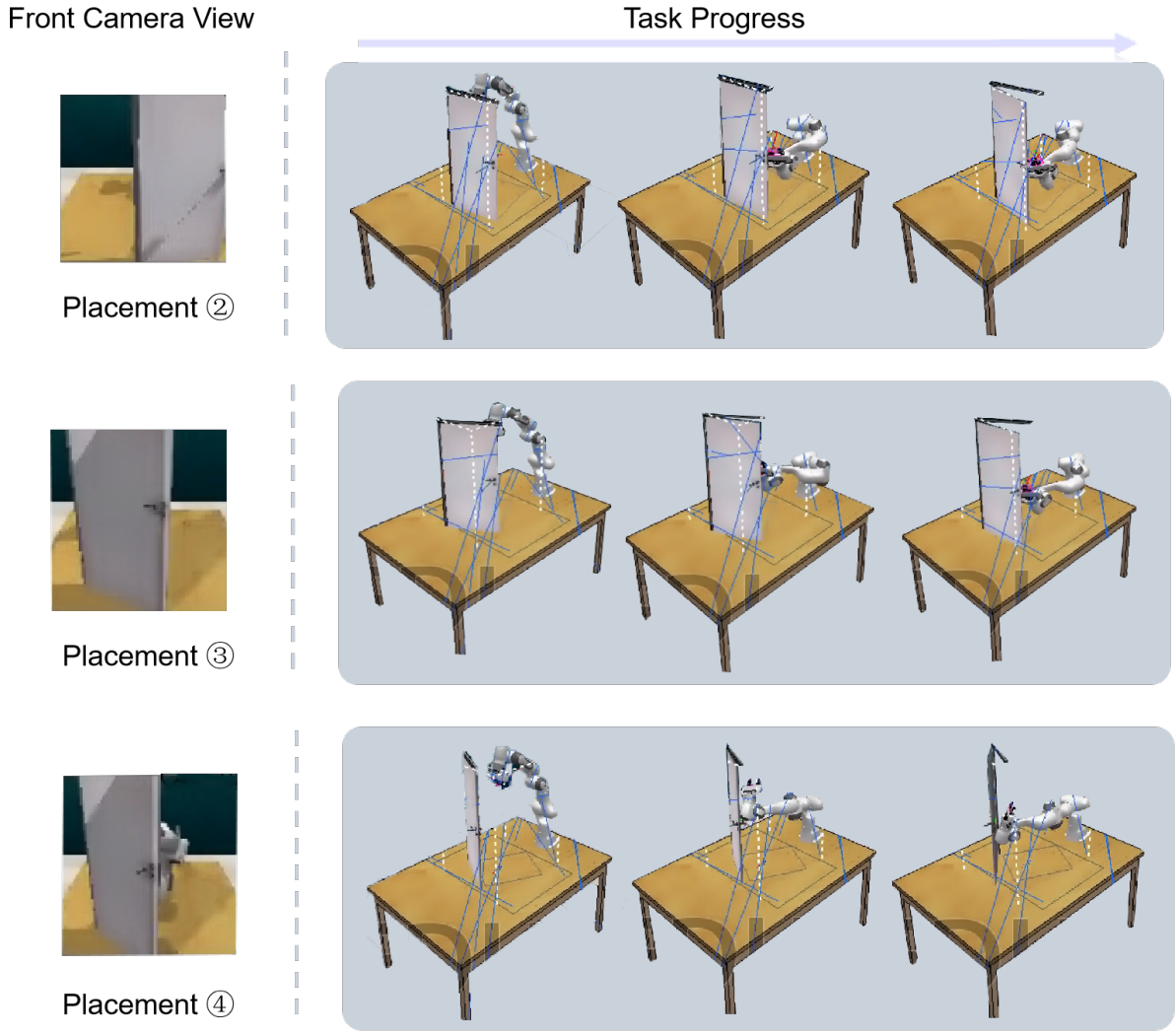


Figure 4-5 **Task progress of the spatial generalization results.** We place the door at different positions that are unseen in the training data. Each position is evaluated with 10 trials.

Learning and Inference Efficiency The training curve of our method, shown in Figure 4-6, demonstrates a rapid convergence towards high accuracy. We train our policy to predict a

sequence of future actions given the current observations. To enhance efficiency, we aim to reduce the effective horizon of long trajectories collected at high frequency.

In our implementation, we fix the chunk size to be K. Specifically, every K steps, the agent receives an observation, generates the next K actions, and executes them sequentially. This strategy effectively reduces the task's horizon by a factor of K, thereby improving computational efficiency. After careful consideration, we set $k = 100$ given that our episode length $T = 180$ in our final implementation, balancing computational cost and accuracy.

Interestingly, we do not apply temporal augmentation [22] during inference. Despite this, our policy still demonstrates robust performance and increased efficiency. As for the observations, we extract 1,024 points from the cropped object out of the 10,000 scene points in the point cloud, significantly improving the point cloud processing speed. Our method achieves an inference speed of 9.68 FPS, highlighting its superior efficiency.

Result Analysis In variation ④, the failure cases occur because the robot sometimes collides with the door with its elbow while approaching the handle. Although the door eventually opens, it does so without the proper manipulation of the handle, which would be hazardous in a real-world scenario. While our method is capable of generalizing across placements, the learned object-centric trajectory of the end effector must also conform to the robot's operational workspace. Otherwise, planning may fail or occlusion may occur.

In our baseline methods, the primary sources of failure arise from the gripper's inability to reach and manipulate the handle, as shown in Figure 4-7(b). In contrast, our method successfully overcomes this limitation by consistently predicting an object-centric trajectory that remains aligned with the door and handle, even when encountering previously unseen positions, as demonstrated in Figure 4-7(a).

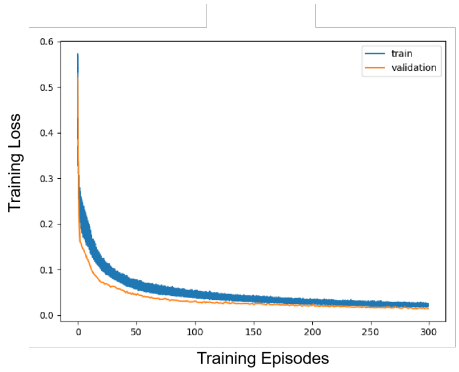


Figure 4-6 **Training curve.**The close alignment between the training and validation loss suggests that the model does not suffer from overfitting.

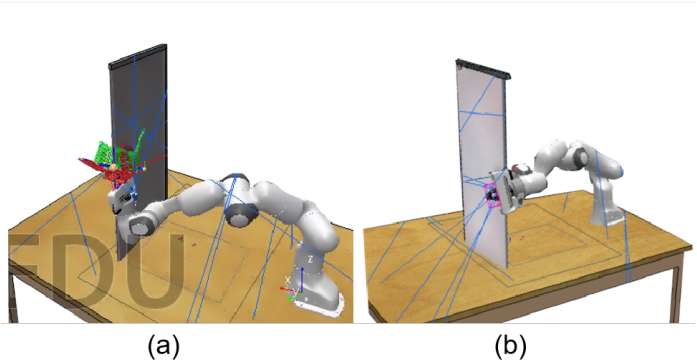


Figure 4-7 **Visualization of the gripper manipulating the handle during inference:** (a) Successful manipulation by our method and (b) Failure case by other baselines. The predicted trajectory is visualized in (a).

5 Conclusion

In this work, we introduce **object-centric representations** for enhanced robot manipulation in imitation learning, enabling the learned policy to generalize effectively across diverse object placements, camera views, and robot configurations.

The essence of our method lies in decoupling the observed object motions from the agent's actions. By defining an object-centric frame $A_t \in SE(3)$ aligned with the object's pose and attached to the contact point p^* on the object, both observations and robot actions are transformed into the object-centric frame. This approach captures invariance under variations in object placements, camera views, and robot configurations, thereby facilitating generalization. Our method demonstrated strong generalization in simulation, achieving a 100% success rate in three variations and 80% in the fourth, despite being trained solely on a single object placement.

Limitations Our method is tested only with the Panda robot arm in a simulated environment for the task of door opening. Therefore, its performance on other robotic embodiments, in real-world settings, or with different manipulation tasks remains unverified. Future work should focus on validating the approach with various robot arms and real-world experiments, as well as exploring its applicability to other articulated object manipulation tasks.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Prof. David Hsu and my mentor Hanbo Zhang, as well as the other members of the AdaComp lab at NUS, for their invaluable support and insightful feedback throughout this work.

REFERENCES

REFERENCES

[1] B. Eisner, H. Zhang, and D. Held, "Flowbot3d: Learning 3d articulation flow to manipulate articulated objects," *arXiv preprint arXiv:2205.04382*, 2022.

[2] R. Martín-Martín, S. Höfer, and O. Brock, "An integrated approach to visual perception of articulated objects," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 5091–5097.

[3] K. Desingh, S. Lu, A. Opipari, and O. C. Jenkins, "Factored pose estimation of articulated objects using efficient nonparametric belief propagation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7221–7227.

[4] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," in *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[5] R. Staszak, M. Molska, K. Młodzikowski, J. Ataman, and D. Belter, "Kinematic structures estimation on the rgb-d images," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 675–681.

[6] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "Screwnet: Category-independent articulation model estimation from depth images using screw theory," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 670–13 677.

[7] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 097–11 107.

[8] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 909–918.

[9] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep part induction from articulated object pairs," *arXiv preprint arXiv:1809.07417*, 2018.

[10] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3706–3715.

[11] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2motion: Joint analysis of motion parts and attributes from 3d shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8876–8884.

[12] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang, "Learning to predict part mobility from a single static snapshot," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.

[13] Z. Xu, Z. He, and S. Song, "Universal manipulation policy network for articulated objects," *IEEE robotics and automation letters*, vol. 7, no. 2, pp. 2447–2454, 2022.

[14] R. Wu, Y. Zhao, K. Mo, Z. Guo, Y. Wang, T. Wu, Q. Fan, X. Chen, L. Guibas, and H. Dong, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," *arXiv preprint arXiv:2106.14440*, 2021.

[15] H. Zhang, B. Eisner, and D. Held, "Flowbot++: Learning generalized articulated objects manipulation via articulation projection," *arXiv preprint arXiv:2306.12893*, 2023.

[16] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[17] P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid, "Instruction-driven history-aware policies for robotic manipulations," in *Conference on Robot Learning*. PMLR, 2023, pp. 175–187.

[18] H. Liu, L. Lee, K. Lee, and P. Abbeel, "Instruction-following agents with jointly pre-trained vision-language models," 2022.

[19] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 739–13 748.

[20] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[21] 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 (CVPR)*, 2016. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2016.90

[22] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[23] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[24] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation," *arXiv preprint arXiv:2306.17817*, 2023.

[25] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[26] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[27] Z. Xian and N. Gkanatsios, "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation," in *Conference on Robot Learning/Proceedings of Machine Learning Research*. Proceedings of Machine Learning Research, 2023.

[28] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "en-USDiffusion policy: Visuomotor policy learning via action diffusion," Mar 2023.

[29] K. Rana, J. Abou-Chakra, S. Garg, R. Lee, I. Reid, and N. Suenderhauf, "Affordance-centric policy learning: Sample efficient and generalisable robot policy learning using affordance-centric task frames," *arXiv preprint arXiv:2410.12124*, 2024.

[30] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," *arXiv preprint arXiv:2103.14127*, 2021.

[31] M. Liu, X. Li, Z. Ling, Y. Li, and H. Su, "Frame mining: a free lunch for learning robotic manipulation from 3d point clouds," *arXiv preprint arXiv:2210.07442*, 2022.

[32] N. Hansen, H. Su, and X. Wang, “Td-mpc2: Scalable, robust world models for continuous control,” *arXiv preprint arXiv:2310.16828*, 2023.

[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[34] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, “Rlbench: The robot learning benchmark learning environment,” 2019.