Kinematics-Aware Diffusion Policy with Consistent 3D Observation and Action Space for Whole-Arm Robotic Manipulation

Anonymous Author(s)

***Abstract*—Full-configuration control of robotic manipulators with awareness of whole-arm kinematics is crucial for many manipulation scenarios involving body collision avoidance or body-object interactions, which makes it insufficient to consider only the end-effector poses in policy learning. The typical approach for whole-arm manipulation is to learn actions in the robot's joint space. However, the unalignment between the joint space and actual task space (i.e., 3D space) increases the complexity of policy learning, as generalization in task space requires the policy to intrinsically understand the non-linear arm kinematics, which is difficult to learn from limited demonstrations. To address this issue, this letter proposes a kinematics-aware imitation learning framework with consistent task, observation, and action spaces, all represented in the same 3D space. Specifically, we represent both robot states and actions using a set of 3D points on the arm body, naturally aligned with the 3D point cloud observations. This spatially consistent representation improves the policy's sample efficiency and spatial generalizability while enabling full-body control. Built upon the diffusion policy, we further incorporate kinematics priors into the diffusion processes to guarantee the kinematic feasibility of output actions. The joint angle commands are finally calculated through an optimization-based whole-arm inverse kinematics solver for execution. Simulation and real-world experimental results demonstrate higher success rates and stronger spatial generalizability of our approach compared to existing methods in body-aware manipulation policy learning. Supplementary materials are available at our Project Website: *https://kinematics-aware-diffusion-policy.github.io*.**

***Index Terms*—Imitation Learning, Deep Learning in Grasping and Manipulation, Learning from Demonstration.**

I. Introduction

IMITATION learning, where an agent learns to mimic the expert demonstrations, is an efficient approach to acquire complex manipulation skills from limited data. Recently, diffusion-based visual-motor policies [1]–[3] have shown many exciting results in imitation learning. Compared to traditional approaches, the remarkable abilities of diffusion models to learn multi-modal, high-dimensional action distributions are the key characteristics contributing to their success.

Due to the alignment between the action space and task space which simplifies the policy learning process, Cartesian-space end-effector pose representations are widely used in existing imitation learning methods. However, for whole-arm robotic manipulation tasks, precise control over the full robot configuration is required, so imitating only the 6D end-effector pose trajectories is naturally insufficient. In many scenarios, such as operating in confined environments, avoiding collisions between the robot arm and surrounding obstacles is crucial. Additionally, certain tasks require the robot to interact with objects using parts of its body rather than the end-effector, further necessitating the whole-arm control. Learning policies in joint space is a typical approach for whole-arm manipulation, which allows joint-level control of the entire configuration. However, joint space is inherently unaligned with the task space where the manipulation is conducted, forcing the policy to implicitly understand the complex non-linear kinematics. Thus, it is hard to learn a generalizable joint-to-task mapping from limited demonstrations, restricting the sample efficiency and spatial generalizability of joint-space policies.

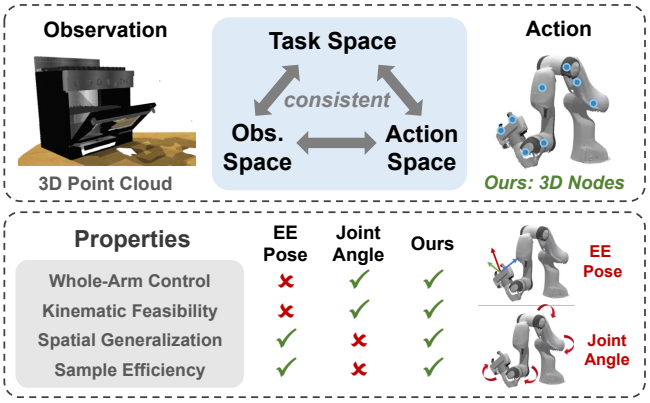


Fig. 1: The proposed approach uses a set of 3D nodes on the arm body as both robot state and action representation for whole-arm manipulation, which is consistent with the 3D point cloud observation space and task space. Compared with using end-effector poses or joint angles, our method achieves higher spatial generalizability and sample efficiency while ensuring kinematic feasibility.

To improve the policy learning performance for whole-arm manipulation, some previous works explore to combine Cartesian space and joint space via incorporating differentiable kinematics within the policy networks [4], [5] or concatenating redundant joint states upon the end-effector poses [6]. However, these methods still require the policy to predict joint-space actions, which cannot avoid the complexity brought by implicitly learning the non-linear kinematics.

In this paper, we propose **K**inematics-**A**ware **D**iffusion **P**olicy (**KADP**), with consistent task, observation, and action spaces. Instead of using joint angles, both robot states and actions are represented with a set of 3D nodes on the robot arm body, making it convenient for the policy to infer the spatial and geometric relationship between the robot configuration and

point cloud observations in the same 3D space. With such spatially consistent representation, the sample efficiency and spatial generalization of policy is improved while whole-arm control is also enabled. To guarantee the kinematic feasibility of predicted 3D nodes, we further incorporate kinematic constraints into diffusion models. For execution, the joint angle commands are finally computed through an optimization-based full-configuration inverse kinematics solver. In summary, the ***kinematics awareness*** of the proposed policy learning approach attributes to the following three aspects:

1) **Whole-Arm Control**: The proposed method enables manipulation over the entire robot configuration, overcoming the limitations of considering only Cartesian-space end-effector poses.
2) **Consistent Task-Observation-Action Spaces**: The node representation is in the 3D space, consistent with the observation and task spaces, allowing the policy to directly infer the spatial relationship between the arm body, objects, and environments.
3) **Kinematic Feasibility Guarantee**: By incorporating analytical joint-node mapping in both forward and reverse diffusion processes, our approach ensures that the generated node positions adhere to kinematic constraints.

Across 8 simulation tasks on RLBench [7] and 4 real-world tasks, we systematically evaluate the performance of the proposed approach, with comparison to several baseline methods using different action representations. KADP achieves higher success rate and stronger spatial generalizability, suggesting the effectiveness of utilizing such 3D node-based robot state and action representation in body-aware manipulation learning.

II. Related Works

A. Diffusion Policies for Imitation Learning

Diffusion models are a class of probabilistic generative models that learn to generate samples from the prior distribution, typically a Gaussian distribution, by an iterative denoising process. For visual imitation learning from demonstrations, Diffusion Policy [1] pioneers the generation of actions through a conditional diffusion model. This innovative formulation is able to effectively learn the multi-modal distribution of demonstration actions while ensuring training stability, which has also been employed as action decoding head in many large-scale generalist policy models. Subsequently, many follow-up works are introduced to improve the generalization ability, data efficiency and inference speed of diffusion policies. DP3 [2] and 3D Diffusion Actor [8] enhance 3D scene representations by using 3D point cloud as observation space instead of RGB images, while some other works further leverage object-centric representations [9] or semantic fields [10]. In this paper, we also adopt 3D point cloud as it has been proved to be more effective than images. Beyond vanilla diffusion models, BESO [3] and PointFlowMatch [11] build policies upon score-based diffusion model and flow matching perspective, respectively. Besides, some works explore integrating several policies trained on heterogeneous data by composition [12], [13] or accelerating diffusion policy with consistency distillation [14].

B. Kinematics-Aware Policy Learning

For robotic manipulation, the selection of action spaces, such as Cartesian space, joint space, and torque space, will greatly influence the performance of various tasks [15]–[17]. Cartesian space, which controls the end-effector pose, is kinematics-unaware but aligns with the 3D Euclidean space in which the robot interacts with, whereas joint space provides complete low-level joint position control but increases the complexity of policy learning, in contrast [18]. Recently, some works are proposed to combine advantages of different action spaces for kinematics-aware policy learning. Mazzaglia et al. [6] introduce a new family of action spaces for overactuated robot arms, which adds the joint position or angle of the redundant joint upon 6D end-effector pose. IKP [5] links Cartesian space and joint space through forward kinematics to learn multi-action space policies. Similarly but implemented in diffusion policy framework, HDP [4] generates both end-effector pose and joint trajectories with two diffusion branches and finally refines joint positions from kinematics-unaware poses. Compared to previous works, we introduce a novel node-based representation in the 3D space consistent with the observation and task space, which avoids requiring the policy to implicitly learn the non-linear kinematics when predicting joint-space actions.

C. Observation and Action Space Alignment

Aligning the observation and action space, which can significantly simplify the observation-to-action mapping, has been shown as an effective way to improve sample efficiency and spatial generalization capability. In 2D image space, R&D [19] renders the gripper virtually in images to jointly represent RGB observations and actions, while Genima [20] visualizes joint actions as colored spheres overlaid on RGB images to indicate visual targets. Extending to 3D space, ActionFlow [21] introduces a unified space composed of object pose and feature sequences to represent both observation and action, but requires additional pose estimators. Other works utilize 3D point clouds or voxels as a simple same observation and action space to avoid extra computation cost of creating new spaces. For instance, C2F-ARM [22], PerAct [23] and DNAct [24] learn per-voxel features from discretized 3D observation and formulate the action prediction as a voxel classification task. Act3D [25] and ChainedDiffuser [26] predict the next keyframe action by selecting a 3D point from uniformly-distributed point candidates, where observation and action lie in the same space. However, these methods only consider the end-effector. In contrast, our proposed KADP enables whole-arm control while preserving high spatial generalizability and sample efficiency afforded by the task-observation-action space alignment.

III. Preliminaries

A. Problem Formulation

A standard imitation learning problem is considered here, where the goal is to learn an observation-to-action mapping $\pi : \mathcal{O} \rightarrow \mathcal{A}$ from a set of expert demonstrations. Usually, the observation O and action A will both contain a few time steps, i.e. $O_t = \{o_{t-T_o+1}, \cdots, o_{t-1}, o_t\}$

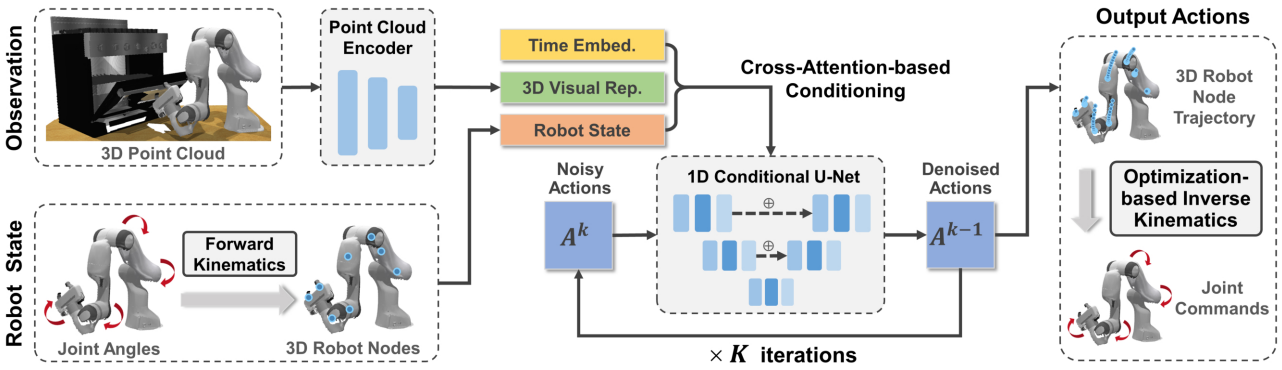


Fig. 2: Overview of Kinematics-Aware Diffusion Policy (KADP). Taking the encoded 3D visual representations, the 3D robot nodes and time embeddings as input, diffusion model predicts the denoised 3D node trajectory iteratively. For execution, the joint angle commands are computed through an optimization-based whole-arm inverse kinematics solver.

and $A_t = \{a_t, a_{t+1}, \cdots, a_{t+T_a-1}\}$, where T_o is the length of observation history horizon and T_a is the length of action prediction horizon. Given a demonstration dataset $D = \{(o_1, a_1, \cdots, o_{T_i}, a_{T_i})\}_{i=1}^n$ consisting of n trajectories with $\{T_i\}_{i=1}^n$ observation-action pairs, the imitation learning process is to train the visuomotor policy represented by a probability distribution $\pi(A|O)$ and then sample a robot action $A_t \sim \pi(A|O_t)$ from it during deployment.

B. Diffusion Policy for Action Generation

For the convenience of derivation in Sec. IV, here we briefly introduce the diffusion policy [1] for action generation. In the forward process, Gaussian noise is iteratively added to the action sample A^0 drawn from real distribution $q(A)$:

$$q(A^k|A^{k-1}) := \mathcal{N}(A^k; \sqrt{1-\beta^k}A^{k-1}, \beta^k I). \quad (1)$$

Given the coefficients $\beta^1, \cdots, \beta^k$ determined by a noise scheduler and $\bar{\alpha}^k = \prod_{i=1}^k (1-\beta^i)$, the noisy sample A^k can be directly sampled from:

$$q(A^k|A^0) := \mathcal{N}(A^k; \sqrt{\bar{\alpha}^k}A^0, (1-\bar{\alpha}^k)I). \quad (2)$$

Starting from an initial Gaussian noise $A^K \sim \mathcal{N}(0, I)$, the reverse process aims to construct the original noise-free data A^0 iteratively. Note that here the current observation O is treated as the diffusion condition, so the parameterized model p_θ can be formulated as:

$$p_\theta(A^{k-1}|A^k, O) := \mathcal{N}(A^{k-1}; \mu_\theta(A^k, O, k), \Sigma_\theta(A^k, O, k)). \quad (3)$$

At each diffusion step k, a denoising network ϵ_θ parameterized by θ is trained to predict the noise component of A^k. The iterative denoising process is

$$A^{k-1} = \alpha_k(A^k - \gamma_k \epsilon_\theta(A^k, O, k)) + \sigma_k \mathcal{N}(0, I). \quad (4)$$

Based on (2) and (3), the model can be trained by maximizing the evidence lower bound (ELBO). During training, a data sample A^0 is randomly sampled, and noise ϵ^k over k steps is added through the forward process. The training objective can be derived to minimize the difference between the added noise and the network ϵ_θ prediction:

$$\mathcal{L} = \text{MSELoss}(\epsilon^k, \epsilon_\theta(A^k, O, k)). \quad (5)$$

IV. Method

A. 3D Node-Based Robot State and Action Representation

An overview of the proposed KADP method is shown in Fig. 2, where a set of 3D nodes is introduced to represent the robot configuration within the diffusion policy framework. Denoted as $A_{\text{node}} = \{(x_0, y_0, z_0), \cdots, (x_m, y_m, z_m)\}$, each node (x_i, y_i, z_i) corresponds to the coordinates of the i^{th} joint, and m denotes the number of selected nodes. This novel node-based representation is defined in the 3D Euclidean space, consistent with the point cloud observation space and task space, allowing the denoising network ϵ_θ to learn within the same 3D space and thus improving its sample efficiency and spatial generalizability. To fully describe the robot configuration with the minimal number of nodes, we manually choose 8 nodes for the 7-DoF Franka Emika Panda robot arm, as shown in the bottom left of Fig. 2. The first 6 nodes are located on the robot arm from the 1^{st} joint (the base) to the 6^{th} joint, ensuring the state of each joint is reflected by the 3D position of the corresponding node. We further place two extra nodes on the left/right gripper fingers to represent both the states of the 7^{th} joint and gripper. Notably, an additional binary value indicating the gripper's open/close action is also included as discrete control of the gripper is empirically found to be more effective. For writing brevity, we will omit the straight-forward implementation in the following sections.

Note that the entire robot joint configuration is uniquely determined given feasible 3D node positions, which enables whole-arm control in contrast with end-effector-based policies. In addition, the space alignment between 3D point cloud observations and node-based states/actions offers higher sample efficiency and stronger spatial generalizability compared to joint-space policies. For instance, when the positions of manipulated objects change, the node-based policy can straightforwardly interpret the spatial relationship between new point cloud observations and 3D node positions. In contrast, reflecting task-space object variations in joint space is non-linear and complex, making joint-space policy learning more difficult.

In the diffusion policy framework, we can seamlessly take such 3D nodes as both the state and action representation for conditional action generation. For robot state, the corresponding node positions can be easily computed from joint angles via

forward kinematics, defined by the mapping $\mathrm{F_{fk}}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{3\times m}$. For execution, the joint angle commands are required to be transferred from the predicted 3D node trajectory, denoted as $\mathrm{F_{ik}}(\cdot) : \mathbb{R}^{3\times m} \rightarrow \mathbb{R}^n$. We achieve this through an optimization-based inverse kinematics solver. Given the predicted 3D node positions A_{node}, the optimization for joint angle commands A_{joint} is formulated as:

$$\begin{aligned} \min_{A_{\text{joint}}} \; & || \, \Lambda \cdot (\mathrm{F_{fk}}(A_{\text{joint}}) - A_{\text{node}}) \, ||^2 \\ \text{s.t. } & \Theta_{\min} \leq A_{\text{joint}} \leq \Theta_{\max}, \end{aligned} \tag{6}$$

where $\Theta_{\min}$ and $\Theta_{\max}$ denote the joint limits, and $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_m)$ is a weight matrix that reflects the relative importance of each node during optimization. In practice, since the accuracy of the last two gripper finger nodes are more critical for task success, they are assigned higher weights ($\lambda_7 = \lambda_8 = 5$), while the remaining nodes are set to 1.

B. Diffusion Model with Kinematic Constraints

Inherently, the 3D node representation is redundant with respect to the actuated DoFs of the arm. Thus, the original diffusion policy cannot guarantee that the generated 3D node positions correspond to a valid robot configuration, where the potential kinematic infeasibility will lead to inaccurate optimized joint commands and affect the manipulation performance. Consequently, We further incorporate kinematic constraints explicitly, ensuring the node positions are kinematic feasible throughout the training and inference process.

Inspired by related works [11], [27] exploring variations of the diffusion model on $SO(3)$ or $SE(3)$ manifold, we define the distance of two node representations within the transferred compact joint space, rather than the original 3D Euclidean space. The interpolation operation between the start nodes A^0 and target nodes A^1 is then expressed as $A^t = \mathrm{F_{fk}}(\, t\, \mathrm{F_{ik}}(A^0) + (1-t)\, \mathrm{F_{ik}}(A^1))$. Similarly, the noise perturbation of node representation is also defined on joint space and then transferred to nodes, so that the forward process can be denoted as:

$$A^k = \mathrm{F_{fk}}(\sqrt{\bar{\alpha}^k}\, \mathrm{F_{ik}}(A^0) + \sqrt{1-\bar{\alpha}^k}\, \epsilon), \tag{7}$$

where the standard Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$.

Following DDPM [28], the posterior distribution can be derived using Bayes' rule as:

$$q(\mathrm{F_{ik}}(A^{k-1}) | A^k, A^0) := \mathcal{N}(\mathrm{F_{ik}}(A^{k-1}); \tilde{\mu}^k(A^0, A^k), \tilde{\beta}^k I), \tag{8}$$

where $\tilde{\mu}^k(A^0, A^k) = \frac{\sqrt{\bar{\alpha}^{k-1}}\beta^k}{1-\bar{\alpha}^k}\mathrm{F_{ik}}(A^0) + \frac{\sqrt{\alpha^k}(1-\bar{\alpha}^{k-1})}{1-\bar{\alpha}^k}\mathrm{F_{ik}}(A^k)$ and $\tilde{\beta}^k = \frac{(1-\bar{\alpha}^{k-1})\beta^k}{1-\bar{\alpha}^k}$. We also follow DP3 [2] to adopt sample prediction for improved high-dimensional action generation. Accordingly, the training objective for the network μ_θ is:

$$\mathcal{L} = \mathrm{MSELoss}(A^0, \, \mu_\theta(A^k, O, k)). \tag{9}$$

To make the network trainable, the differentiability of the mappings $\mathrm{F_{fk}}$ and $\mathrm{F_{ik}}$ in (9) is required. For the joint-to-node mapping $\mathrm{F_{fk}}$, differentiable forward kinematics with a predefined robot URDF model can be adopted. However, the node-to-joint mapping implemented by optimization-based inverse kinematics solver, denoted as $\mathrm{F_{ik\_opt}}$, is non-differentiable, preventing gradients from passing through. To address this, we pretrain a lightweight MLP, denoted as $\mathrm{F_{ik\_mlp}}$, to approximate this ik mapping and freeze it during policy model training. Compared with the optimization-based $\mathrm{F_{ik\_opt}}$, the MLP-based $\mathrm{F_{ik\_mlp}}$ is differentiable but less accurate. Thus, $\mathrm{F_{ik\_mlp}}$ is only used during training and $\mathrm{F_{ik\_opt}}$ is employed for accurate node-to-joint mapping during inference.

Algorithm 1 Training Procedure of KADP

repeat
- $(O, A^0) \sim D$ ▷ sample dataset
- $k \leftarrow \mathrm{Randint}(0, K)$ ▷ sample diffusion step
- $\epsilon \sim \mathcal{N}(0, I)$ ▷ sample noise
- $A^k = \mathrm{F_{fk}}(\sqrt{\bar{\alpha}^k}\, \mathrm{F_{ik\_mlp}}(A^0) + \sqrt{1-\bar{\alpha}^k}\, \epsilon)$ ▷ (7)
- $L = \mathrm{MSELoss}(A^0, \, \mu_\theta(A^k, O, k))$ ▷ (9)
- $\theta = \theta - \alpha \nabla_\theta L$ ▷ update network params

until μ_θ converged

Algorithm 2 Sampling Procedure of KADP

- $A^K \sim \mathcal{N}(0, I)$ ▷ sample starting point
- **for** $k = K$ to 1 **do**
  - $z \sim \mathcal{N}(0, I)$ ▷ sample noise
  - $\hat{A}^0 = \mu_\theta(A^k, O, k)$ ▷ network prediction
  - $\tilde{\mu}^k = \frac{\sqrt{\bar{\alpha}^{k-1}}\beta^k}{1-\bar{\alpha}^k}\mathrm{F_{ik\_opt}}(\hat{A}^0) + \frac{\sqrt{\alpha^k}(1-\bar{\alpha}^{k-1})}{1-\bar{\alpha}^k}\mathrm{F_{ik\_opt}}(A^k)$
  - $A^{k-1} = \mathrm{F_{fk}}\big(\tilde{\mu}^k + \sqrt{\tilde{\beta}^k}\, z\big)$ ▷ (10)
- **end for**
- **return** A^0

Starting from a noise $A^K \sim \mathcal{N}(0, I)$, action generation via a iterative denoising process also follows the Diffusion Policy framework. The predicted original sample $\hat{A}^0 = \mu_\theta(A^k, O, k)$ is used to compute the mean value of the distribution of A^{k-1} in (8). The sampling process can be written as:

$$A^{k-1} = \mathrm{F_{fk}}\big(\, \tilde{\mu}^k(\hat{A}^0, A^k) + \sqrt{\tilde{\beta}^k}\, z\big), \tag{10}$$

where $z \sim \mathcal{N}(0, I)$ represents the random Gaussian noise.

The training and sampling procedures are summarized in Alg. 1 and Alg. 2. During training, the initial sample is mapped to joint space via the MLP-based $\mathrm{F_{ik\_mlp}}$, perturbed with noise, and then projected back to node space. During inference, at each denoising step, the network predicts a clean sample estimate and generates the one-step denoised sample using the optimization-based $\mathrm{F_{ik\_opt}}$. Note that although the nodes are initially transferred to joint space and later recovered, the space alignment between observation and action is still maintained throughout action process as both input and output of noise prediction network remain in the consistent 3D space.

C. Implementation Details

The 3D point cloud is first encoded using an MLP-style encoder, which has been shown to be simple yet effective in prior work [2], and then concatenated with the robot state, and sinusoidal diffusion timestep embeddings to form the condition. A DDIM [29] noise scheduler is adopted with 100 steps at

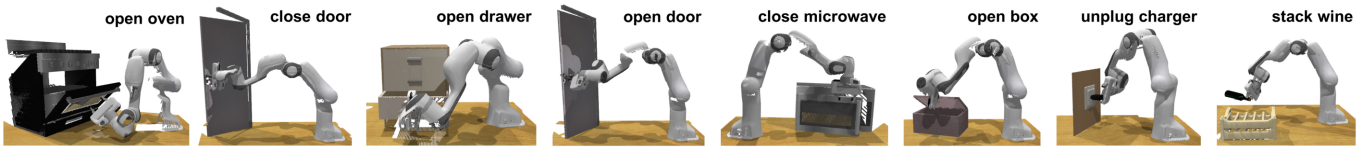


Fig. 3: Visualization of 8 RLBench tasks used for evaluation, where coordinated whole-arm motion contributes to successful manipulation.

TABLE I: Performance of our proposed KADP and the baselines on 8 RLBench simulation tasks. The average success rate (%) and its standard deviation with 3 individual evaluation runs on 100 episodes per task are reported, where the best results are bold.

Method	`open oven`	`open drawer`	`open box`	`open door`	`close door`	`close microwave`	`unplug charger`	`stack wine`	Average
DP3-EE	24.3 ±2.1	75.7 ±1.5	70.7 ±2.5	33.3 ±2.5	6.0 ±1.0	6.7 ±0.6	**36.7** ±5.7	68.0 ±2.0	40.2 ±0.5
DP3-Joint	31.3 ±4.0	23.3 ±4.2	77.7 ±5.0	28.0 ±2.0	18.7 ±2.5	79.7 ±4.0	15.3 ±1.5	68.3 ±2.1	42.8 ±1.3
DP3-ERJ [6]	23.7 ±3.2	55.7 ±3.2	68.0 ±2.6	37.0 ±3.6	15.0 ±2.0	52.3 ±0.6	30.7 ±2.5	**75.3** ±3.1	44.7 ±0.9
DP3-HDP [4]	39.3 ±3.1	62.7 ±1.2	**83.3** ±3.2	43.0 ±2.6	24.0 ±2.0	76.0 ±2.6	22.0 ±2.6	63.7 ±2.1	51.8 ±0.9
KADP (Ours)	**51.3** ±2.1	**92.7** ±2.3	76.3 ±1.5	**55.0** ±2.0	**50.0** ±2.6	**87.3** ±2.9	31.3 ±4.5	70.7 ±3.2	**64.3** ±1.3

training and 10 steps at inference. All policy models are trained for 3000 epochs using the AdamW optimizer, with a learning rate of 1e-4 and weight decay of 1e-6.

The optimization-based IK solver is implemented using Sequential Least Squares Programming (SLSQP) to minimize the weighted squared node position error under joint limit constraints, initialized with the joint configuration from the previous control step. The MLP-based IK model, consisting of three simple layers, is trained offline with randomly sampled joint angles and their corresponding node positions generated via forward kinematics as ground-truth supervision. Additional implementation details and IK approximation errors are provided in the Appendix (available on our Project Website).

V. Simulation Evaluation

A. Evaluation Settings

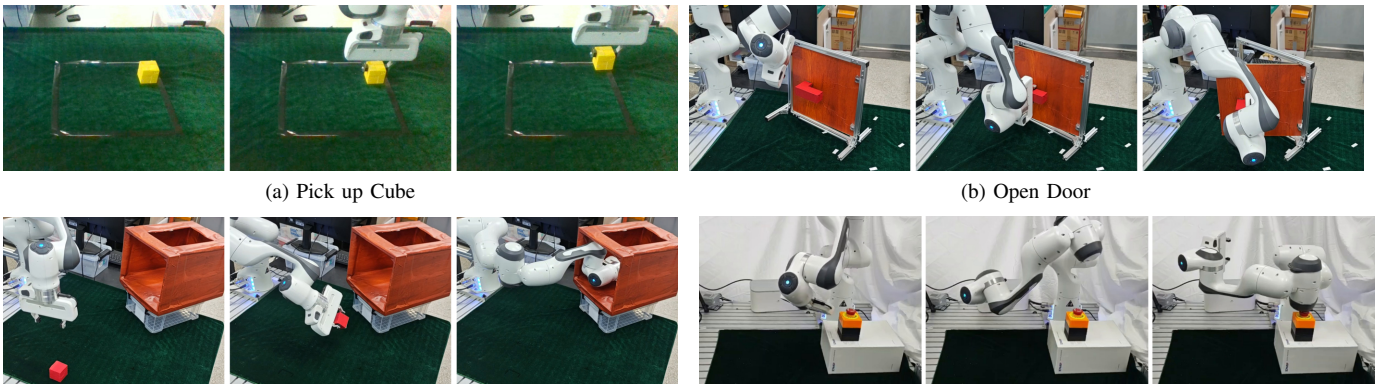
From the popular robot learning benchmark RLBench [7], we pick 8 challenging tasks for evaluation. Almost all the selected tasks are difficult to execute with only end-effector control, while whole-arm control contributes a lot to successful manipulation. The standard demonstration collection interface in RLBench is utilized to collect 20 expert trajectories for each task. The resolution of RGB-D images captured by five multi-view cameras is 128×128, from which the object region is segmented and projected to 3D space as the point cloud observation. For batch training, we downsample the point cloud to 1024 points with Farthest Sampling Point algorithm. Other hyper-parameters include the observation horizon $T_o = 2$, action horizon $T_a = 8$ and the execution horizon $T_e = 4$. For each task, the average success rate and its standard deviation with 3 individual evaluation runs on 100 episodes are reported.

B. Comparison with Baselines

We compare KADP against the following baselines: 1) *DP3-EE*: generates a sequence of 6D end-effector poses, and executes correspondent joint commands via inverse kinematics solvers; 2) *DP3-Joint*: generates a sequence of joint angles; 3) *DP3-ERJ*: uses the ERJ space [6], which concatenates end-effector poses with redundant joint positions; 4) *DP3-HDP*: generates both end-effector poses and joint positions using two diffusion branches, followed by two-branch refinement through differentiable kinematics [4]. For fair comparison, all the settings described in Sec. V-A are kept identical, except for the robot state and action representations.

As shown in Table I, although the performance of methods shows variability among tasks due to individual evaluation runs with different random seeds, KADP consistently achieves the best or second-best performance on nearly all the tasks with an overall average success rate of 64.3%. These experimental results demonstrate that KADP, benefiting from whole-arm control and the alignment between task, observation, and action spaces, is capable of considering whole-arm motion while maintaining precise manipulation and high sample efficiency. Among the baselines, DP3-EE and DP3-Joint exhibit relatively lower performance, highlighting the limitations of considering solely end-effector pose or learning in joint space without maintaining consistency with the observation and task spaces. By combining the advantages of Cartesian and joint spaces, DP3-ERJ and DP3-HDP achieve improved results, but their partial space alignment and lack of explicit kinematic awareness still limit their effectiveness compared to the proposed KADP.

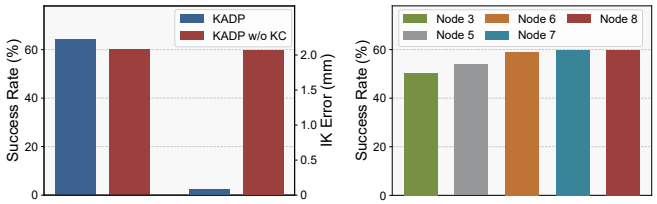
In our experiments, several typical failure modes of the baseline methods are observed. For instance, DP3-EE reaches only 6.7% success on the `close microwave` task, where most failures stem from the generated kinematically infeasible end-effector poses. DP3-HDP, which refines the end-effector poses to kinematics-aware joint positions, largely avoids IK errors and improves the performance of end-effector–based policies, reaching a 76.0% success rate on this task. On the `open oven` task, controlling only the end-effector pose by DP3-EE frequently causes the arm body to collide with the oven door during the arm lifting stage. In contrast, DP3-Joint enables smoother control of individual joints, but suffers from inaccurate task-space generalization, resulting in more frequent failures when grasping the oven's thin handle..



(a) Pick up Cube (b) Open Door

(c) Put Cube in Cabinet (d) Push Button Elbow

Fig. 4: Illustration of the 4 real-world manipulation tasks, which are designed for evaluating the capabilities of policies compresensively.



(a) Kinematic Constraints in DP (b) The Number of Nodes (w/o KC)

Fig. 5: Ablation on the kinematic constraints in DP and the number of nodes. IK Error refers to the average per-node IK optimization error when solving joint commands. *KADP w/o KC*: remove the kinematic constraints. *Node-3/5/6/7*: replace the full 8 nodes with fewer nodes.

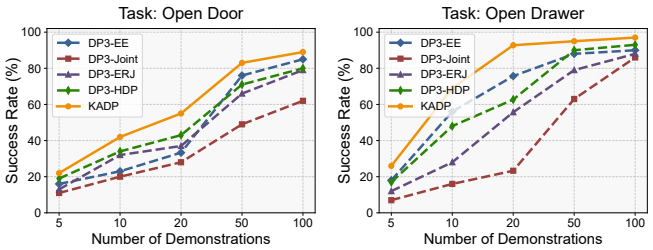


Fig. 6: Success rate of the proposed KADP and baselines on two representative RLBench tasks, `open door` and `open drawer`, with increasing number of expert demonstrations.

C. Ablation and Analysis

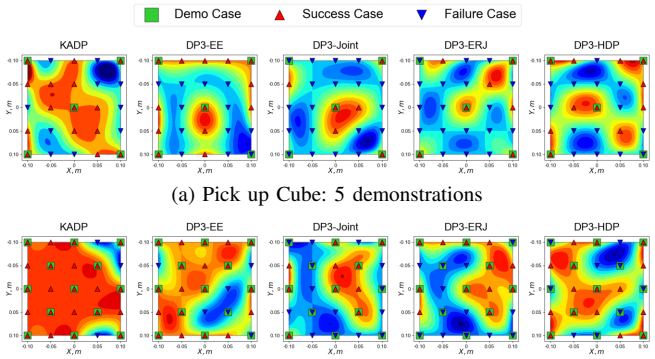
Kinematic Constraints in DP: Firstly, we conduct an ablation study on the effect of kinematics-aware diffusion process proposed in Sec. IV-B. To assess the kinematic feasibility of the generated node positions, we calculate the average per-node distance (i.e., IK error) between the diffusion policy's predicted 3D nodes and those corresponding to joint angles obtained by the optimization-based inverse kinematics solver. When taking the 3D nodes directly as the state and action representation in diffusion policy, the predicted actions cannot be theoretically guaranteed to be kinematically feasible. As shown in Fig. 5a, the predicted node positions remain approximately kinematically feasible in practice with an average IK error of 2.1mm, even in the absence of explicit constraints. In contrast, the full KADP framework, which incorporates kinematic constraints into the diffusion process, reduces the IK error to nearly zero and slightly improve the task success rate by 4.3%.

The Number of Nodes: We also conduct an ablation study on the number of nodes, considering several reduced sets denoted as *Node-3/5/6/7*. In addition to the two gripper nodes, these variants include a subset of the remaining nodes on the robot to approximate the full configuration. For a fair comparison, we compare these ablated settings against KADP without kinematic constraints with full 8 nodes, denoted as *Node-8*. As shown in Fig. 5b, success rate increases noticeably as the number of nodes grows from Node-3 to Node-6, clearly demonstrating steady performance gains from introducing additional intermediate nodes to better represent the robot's entire configuration. Performance begins to saturate beyond six nodes at around 60%, while explicitly incorporating kinematics based on 8 nodes (full KADP) further improves the success rate to 64.3%, as shown in Fig. 5a. Since the omitted joints in Node-6/7 correspond to those near the base, which rarely interact with objects or obstacles during manipulation, their comparable performance with Node-8 is expected. Given the negligible additional computational overhead between 7 and 8 nodes, we adopt the full 8-node configuration as the default setting for the 7-DoF robot arm without loss of generality.

The Number of Demonstrations: To study how the performance of KADP and baselines changes with the increasing number of expert demonstrations, we further evaluate two RLBench tasks, `open door` and `open drawer`, under varying counts ($N_{\text{demo}} \in \{5, 10, 20, 50, 100\}$). As shown in Fig. 6, while all methods benefit from a larger number of demonstrations, KADP (solid yellow line) exhibits consistent superior performance compared to baselines. In the limited demonstration regime, where the workspace is only sparsely covered, KADP shows a clear performance advantage with faster convergence. On the `open drawer` task, KADP achieves nearly 70% success with just 10 demonstrations and exceeds 90% with 20 demonstrations. While the performance gap between KADP and baselines narrows at 50 or 100 demonstrations, since sufficient demonstration density is expected to mitigates inconsistencies in the task, observation, and action spaces for the baselines as expected, KADP's superior sample efficiency in limited-data scenarios is clearly demonstrated.

TABLE II: Performance of the proposed KADP and baselines on 4 real-world manipulation tasks. For the `pick up cube` task, 25 trials are conducted to validate real-world spatial generalization and sample efficiency on 25 grid vertices.

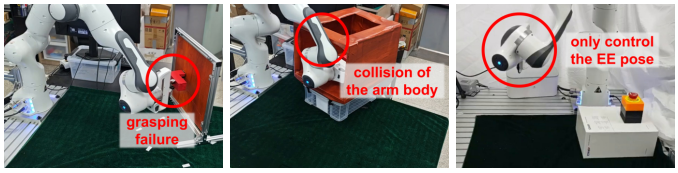
Method	`pick up cube` (5 demos)	`pick up cube` (13 demos)	`open door`	`put cube in cabinet`	`push button elbow`
DP3-EE	13/25	19/25	**8/10**	1/10	0/10
DP3-Joint	6/25	10/25	6/10	7/10	**10/10**
DP3-ERJ [6]	7/25	12/25	7/10	8/10	8/10
DP3-HDP [4]	11/25	16/25	**8/10**	7/10	**10/10**
KADP (Ours)	**15/25**	**22/25**	**8/10**	**9/10**	**10/10**



(a) Pick up Cube: 5 demonstrations

(b) Pick up Cube: 13 demonstrations

Fig. 7: Spatial generalization performance on `pick up cube` task.



(a) DP3-Joint (b) DP3-EE (c) DP3-EE

Fig. 8: Representative failure cases of baseline methods on real-world tasks. Additional examples are provided in the Appendix.

VI. Real-world Experiments

A. Environment Setup

A 7-DoF Franka Emika Panda robot arm is used as the real-world platform, equipped with a fixed front-view RealSense D435 camera to capture point clouds. Snapshots of 4 designed real-world tasks are shown in Fig. 4. For all tasks except `pick up cube`, we collect 10 demonstrations for training and perform 10 evaluation trials, with randomized object poses. With an average inference latency of 0.13s on an NVIDIA RTX 3090 GPU, we run the policy at 5Hz and control the robot at 10Hz by executing two of predicted actions. Aside from adjusting the prediction and execution horizons to $T_a = 4$ and $T_e = 2$ accordingly, all other hyper-parameters are kept consistent with simulation studies. The 4 tasks are as follows:

Pick up Cube: The robot only needs to grasp the object and lift it, which is designed to specifically analyze the spatial generalizability and sample efficiency of policies. Since accurately controlling the end-effector pose is sufficient, DP3-EE is expected to perform well.

Open Door: The robot should grasp the handle and follow a circular trajectory to open the door. The narrow handle makes the task sensitive to small positional errors, which can cause the gripper to lose contact with the handle. While end-effector control is also sufficient, this task is more challenging.

Put Cube in Cabinet: The robot must grasp a cube and place it into a deep, narrow cabinet. The main challenges are reaching the cabinet's deepest point without arm-body collisions, requiring the entire robot to remain nearly horizontal, and inserting the cube into a space only slightly wider than the gripper, making the task also sensitive to positional errors.

Push Button Elbow: The robot is required to press a button using its elbow instead of the gripper, making it meaningless to control only the end-effector pose. Learning directly from joint space is expected to yield good performance as only the angles of the first 3 joints change during the process.

B. Experimental Results and Comparison with Baselines

Spatial Generalization and Sample Efficiency: On the `pick up cube` task, the cube's initial positions are constrained within a 20cm$\times$20cm workspace, and a 5×5 grid of evaluation cases is uniformly sampled. Two demonstration settings are used: 1) 5 demos at the center and four corners, and 2) 13 demos including the center, corners, edge midpoints, and midpoints between the center and corners. DP3-Joint performs worst under both settings as reported in Table II, confirming the difficulty of learning effective policy from joint space with limited data. DP3-ERJ and DP3-HDP outperform DP3-Joint, but reliance on post-optimization or refinement can lead to occasional end-effector inaccuracies, yielding slightly lower performance than DP3-EE. KADP achieves success rates of 60% and 88% for two settings, surpassing all baselines. Heatmaps in Fig. 7, generated via cubic interpolation over evaluation cases, show substantially larger success regions of KADP (the first column), demonstrating superior spatial generalizability within the demonstration coverage.

On the `open door` task, KADP, DP3-EE, and DP3-HDP each achieve 8/10 successful trials, while DP3-Joint only attains 60% success. These results indicate that KADP maintains the effectiveness of end-effector–based policies when precise gripper control is sufficient, while offering better generalization and sample efficiency than joint-space learning. As shown in Fig. 8a, DP3-Joint typically fails due to inaccurate gripper positioning, whereas KADP's failures mainly occur under challenging out-of-distribution door poses.

Whole-Arm Manipulation: On the `put cube in cabinet` task, since the robot configuration cannot be fully determined

by end-effector poses alone, DP3-EE struggles with only a 10% success rate. As shown in Fig. 8b, although the predicted end-effector pose is often suitable for insertion, frequent collisions with the cabinet's top surface cause failures. Other baselines enable full-configuration control, but inaccurate task-space generalization often results in collisions with the side surfaces or insertion failures. In contrast, KADP effectively overcomes both issues above, achieving up to 90% success.

On the `push button elbow` task, DP3-EE fails in all trials, as it simply imitates end-effector trajectories without capturing the true task intent. As shown in Fig. 8c, although the end-effector reaches positions similar to successful cases, DP3-EE cannot properly coordinate the elbow to press the button. In contrast, KADP and the other baselines achieve near 100% success. This comparable performance is expected, since the joint-space action in this task is only three-dimensional, making the mapping to 3D task space relatively easy to learn.

VII. Conclusion

In this paper, we present Kinematics-Aware Diffusion Policy (KADP), an imitation learning framework that aligns task, observation, and action spaces in the consistent 3D space for effective whole-arm robotic manipulation. By representing both robot states and actions as a set of 3D nodes on the robot arm, KADP improves sample efficiency and spatial generalization compared to end-effector-pose or joint-space approaches, while also enabling full-configuration control. Extensive experiments in both simulation and real-world environments demonstrate the superiority of KADP in complex and body-aware manipulation tasks, underscoring its potential as a scalable and generalizable solution for learning whole-arm robot behaviors from limited demonstrations. Despite its effectiveness, KADP still has several limitations. Like all fixed-data imitation learning approaches, it is restricted to the distribution of the provided demonstrations and struggles with out-of-distribution generalization. Moreover, while the node-based representation integrates well with the diffusion policy framework, its high dimensionality may limit compatibility with other paradigms such as reinforcement learning. Future directions could involve exploring its performance in large-scale policy and extending to long-horizon, multi-task scenarios.

References

[1] 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*, 2024.

[2] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Robotics: Science and Systems*, 2024.

[3] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal conditioned imitation learning using score-based diffusion policies," in *Robotics: Science and Systems*, 2023.

[4] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[5] A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng, "Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning," in *International Conference on Robotics and Automation*, 2022.

[6] P. Mazzaglia, N. Backshall, X. Ma, and S. James, "Redundancy-aware action spaces for robot learning," *IEEE Robotics and Automation Letters*, 2024.

[7] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[8] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," in *8th Annual Conference on Robot Learning*, 2024.

[9] Q. Feng, H. Li, Z. Zheng, J. Feng, and A. Knoll, "Language-guided object-centric diffusion policy for collision-aware robotic manipulation," in *International Conference on Robotics and Automation*, 2025.

[10] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li, "GenDP: 3d semantic fields for category-level generalizable diffusion policy," in *8th Annual Conference on Robot Learning*, 2024.

[11] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, "Learning robotic manipulation policies from point clouds with conditional flow matching," in *8th Annual Conference on Robot Learning*, 2024.

[12] L. Wang, J. Zhao, Y. Du, E. H. Adelson, and R. Tedrake, "Poco: Policy composition from and for heterogeneous robot learning," in *Robotics: Science and Systems*, 2024.

[13] Y. Wang, Y. Zhang, M. Huo, T. Tian, X. Zhang, Y. Xie, C. Xu, P. Ji, W. Zhan, M. Ding, and M. Tomizuka, "Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning," in *8th Annual Conference on Robot Learning*, 2024.

[14] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency policy: Accelerated visuomotor policies via consistency distillation," in *Robotics: Science and Systems*, 2024.

[15] R. Martin-Martin, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[16] P. Varin, L. Grossman, and S. Kuindersma, "A comparison of action spaces for learning manipulation tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[17] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *IEEE International Conference on Robotics and Automation*, 2021.

[18] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt, "On the role of the action space in robot manipulation learning and sim-to-real transfer," *IEEE Robotics and Automation Letters*, 2024.

[19] V. Vosylius, Y. Seo, J. Uruç, and S. James, "Render and diffuse: Aligning image and action spaces for diffusion-based behaviour cloning," in *Robotics: Science and Systems*, 2024.

[20] M. Shridhar, Y. L. Lo, and S. James, "Generative image as action models," in *8th Annual Conference on Robot Learning*, 2024.

[21] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters, "Actionflow: Equivariant, accurate, and efficient manipulation policies with flow matching," in *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.

[22] 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.

[23] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *6th Annual Conference on Robot Learning*, 2022.

[24] G. Yan, Y.-H. Wu, and X. Wang, "Dnact: Diffusion guided multi-task 3d policy learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

[25] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation," in *7th Annual Conference on Robot Learning*, 2023.

[26] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki, "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation," in *7th Annual Conference on Robot Learning*, 2023.

[27] H. Jiang, M. Salzmann, Z. Dang, J. Xie, and J. Yang, "Se(3) diffusion model-based point cloud registration for robust 6d object pose estimation," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[28] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, 2020.

[29] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.