KDPE: A Kernel Density Estimation Strategy for Diffusion Policy Trajectory Selection

Andrea Rosasco[1,2] **Federico Ceola**[1] **Giulia Pasquale**[1] **Lorenzo Natale**[1]
[1]Istituto Italiano di Tecnologia, [2]University of Genoa
{andrea.rosasco, federico.ceola, giulia.pasquale, lorenzo.natale}@iit.it

Abstract: Learning robot policies that capture multimodality in the training data has been a long-standing open challenge for behavior cloning. Recent approaches tackle the problem by modeling the conditional action distribution with generative models. One of these approaches is Diffusion Policy, which relies on a diffusion model to denoise random points into robot action trajectories. While achieving state-of-the-art performance, it has two main drawbacks that may lead the robot out of the data distribution during policy execution. First, the stochasticity of the denoising process can highly impact on the quality of generated trajectory of actions. Second, being a supervised learning approach, it can learn data outliers from the dataset used for training. Recent work focuses on mitigating these limitations by combining Diffusion Policy either with large-scale training or with classical behavior cloning algorithms. Instead, we propose KDPE, a Kernel Density Estimation-based strategy that filters out potentially harmful trajectories output of Diffusion Policy while keeping a low test-time computational overhead. For Kernel Density Estimation, we propose a manifold-aware kernel to model a probability density function for actions composed of end-effector Cartesian position, orientation, and gripper state. KDPE overall achieves better performance than Diffusion Policy on simulated single-arm tasks and real robot experiments.

Additional material and code are available on our project page: https://hsp-iit.github.io/KDPE/.

Keywords: Behavior Cloning, Manipulation, Trajectory Selection.

1 Introduction

Diffusion Policy (DP) [1] has recently emerged as a powerful robotic policy representation due to its capability of handling multimodal behaviors. DP models the robot policy as a Denoising Diffusion Probabilistic Model (DDPM) [2], and during policy execution denoises a set of randomly sampled trajectory points into the trajectory that the robot is requested to execute with receding horizon control [3]. While enabling the robot to capture multimodality in the demonstrations used for training, the choice of the random points to initialize the denoising process performed at inference can lead the robot to execute different trajectories, given the same observation. This may be problematic if the sampled trajectory is an outlier with respect to modes that are most represented in the training data, leading the robot into a state which may be out of the distribution of the demonstrations dataset.

To overcome this limitation, we propose KDPE: a strategy to sample trajectories that are representative of the modes learned by the policy. We propose to compute a set of N trajectories, by performing in parallel N denoising processes from N different starting noise samples conditioned on the same current observation. We then estimate the Probability Density Function (PDF) on the last actions in the trajectories generated by Diffusion Policy, using Kernel Density Estimation (KDE), and select the trajectory associated to the action with the highest density.

The actions we consider are composed of end-effector pose and gripper state. Modeling a KDE on a population of actions requires a kernel function that relates end-effector position, orientation and gripper state. While probability distributions on end-effector poses have previously been proposed [4, 5], none of them integrate the gripper state. We modify the probability distribution introduced in [4] to handle the gripper state component.

In principle, the multimodality of the trajectories predicted by DP can be modeled with non-parametric clustering algorithms [6]. However, these approaches require non-negligible convergence time when applied for outlier rejection [7] in closed-loop control tasks. KDPE, instead, manages to filter and sample among a subset of the trajectories predicted by DP with a significantly lower computational overhead, making it suitable for the target visuomotor closed-loop control tasks. While KDPE uses DP for trajectory generation, the core idea of sampling multiple trajectories and filtering them through KDE can be applied to any probabilistic generative policy.

We evaluate KDPE against DP on the four RoboMimic [8] single-arm tasks used in [1] to benchmark DP, and on three tasks from the MimicGen benchmark [9]. We train DP on these tasks, and evaluate the models with KDPE, achieving an overall improvement in terms of average success rate. We then evaluate KDPE's robustness to visual perturbations by changing the color of an object in the task environments and show that KDPE maintains performance closer to the non-shifted setting than DP.

Finally, using a Franka Emika Panda [10] manipulator, we test KDPE on three real-world tasks: *PickPlush*, a tabletop picking task where a plush has to be grasped and lifted by the robot, *CubeSort*, a multi-step multimodal task where the robot has to pick two cubes and place them in the cup of the corresponding color, and *CoffeeMaking*, a fine-grained multi-step task where the robot has to load a coffee machine with a pod. Overall, KDPE outperforms DP in terms of success rate, and qualitatively shows smoother behavior.

In summary, the contributions of the paper are:

- **KDPE**: a KDE-based approach for selection of DP action trajectories that are representative of the different action modalities.
- A **manifold-aware kernel for KDE** on robotic actions composed of Euclidean end-effector position and gripper aperture, and non-Euclidean end-effector orientation.
- An **extended quantitative evaluation of KDPE** on seven simulated tasks from two different benchmarks and on three real-world tasks.
- A **trajectory visualizer** for qualitative analysis of manipulation policies.

2 Related Work

There is a huge effort in the robotics community to create generalist robot policies [11, 12, 13] from large-scale datasets [14, 15], either with large transformer-based models [16, 13, 11, 17, 18], or parameterizing robot policies as language tokens [19, 12, 20]. However, for single tasks, there is evidence [12] that DP [1] achieves better performance than large generalist models [11, 12].

DP [1], together with its flow matching variant [21], is currently one of the most widely used approaches for behavior cloning for robotic manipulation tasks. Its policy parameterization as a Denoising Diffusion Probabilistic Model (DDPM) allows to model the multimodality in the training data. Several recent works extend the original DP problem formulation to learn goal-conditioned [22], or language-conditioned [23, 24] tasks. Octo [11], instead, uses DP to parameterize a generalist robot manipulation policy on top of a large transformer-based modular backbone for transferability across tasks and embodiments. Approaches like the one presented in [25] combine DP with more classical behavior cloning algorithms like DAgger [26] to reduce the compounding errors typical of pure imitation learning algorithms.

Recent work has explored failure detection and mitigation strategies to enhance the reliability of robot policies. Sentinel [27] identifies potential failures by analyzing the consistency of overlapping trajectory segments sampled at different time-steps, and leverages vision-language models to detect misalignment between actions and visual context. While effective, such methods often detect failures after the policy has already entered an out-of-distribution state, making recovery challenging. RoboMD [28] addresses robustness by systematically identifying failure modes through DRL-guided exploration across diverse environmental conditions. RoboMD fine-tunes policies on these identified failure modes. This *train-deploy-analyze-retrain* pipeline, however, can be impractical for real-world deployment. V-GPS [29] improves robot policies at inference time by using a value function network to rank actions. However, this requires to train a value function network with offline Reinforcement Learning and design a reward function. In contrast, KDPE operates at inference time without requiring any additional training. By leveraging the inherent diversity of DP's trajectory sampling process, KDPE mitigates failures proactively, selecting trajectories that align with the statistical modes of the learned distribution, rather than relying on post-hoc detection or costly retraining.

A growing body of literature in Natural Language Processing studies *inference or test-time scaling* methods that select one among multiple output samples generated (e.g. by a Large Language Model) to improve the performance of the trained model [30, 31, 32]. Furthermore, there is recent research interest in the application of inference-time scaling to, e.g., text-to-image diffusion models [33, 34]. In [35] the effectiveness of the easiest *best-of-N sampling* (generation of a high number of random samples and selection of the best one by using an external evaluator) is proved experimentally. To the best of our knowledge, however, KDPE is the first method that improves DP at inference time based on statistical properties of the trajectories distribution.

3 Methodology

3.1 Diffusion Policy

DP adapts DDPM, a generative diffusion model which is commonly used for image generation, to behavior cloning for robotic manipulation tasks in a receding horizon fashion. DP, given an observation of the environment, outputs action trajectories of shape $T \times D$, where T is the number of time-steps and D is the action dimensionality. These trajectories are generated through a denoising process that progressively refines Gaussian noise by iteratively subtracting a learned gradient field. This process allows the policy to model multimodal trajectory distributions and maintain the multimodality of task demonstrations. The denoising process to generate a trajectory is defined as:

$$\mathbf{A}^{k-1} = \alpha(\mathbf{A}^k - \gamma\epsilon_\theta(\mathbf{A}^k, k) + \mathcal{N}(0, \sigma^2 I)), \tag{1}$$

where ϵ_θ is the noise prediction network with parameters θ and $\mathbf{A}^k$ is a noisy sample going through a denoising step. This process is repeated for K steps, starting with $\mathbf{A}^K$ as randomly sampled Gaussian noise, to output the trajectory $\mathbf{A}^0$. During training, a trajectory is sampled from the dataset and is perturbed by adding the appropriate amount of noise corresponding to a random denoising step k. The noise prediction network ϵ_θ takes as input the noisy sample and is optimized to predict the noise ϵ^k that has been added to the ground-truth trajectory $\mathbf{A}^0$ with the following loss function:

$$\mathcal{L} = MSE(\epsilon^k, \epsilon_\theta(\mathbf{A}^0 + \epsilon^k, k)). \tag{2}$$

3.2 KDPE

KDPE enhances Diffusion Policy by scoring the population of trajectories via Kernel Density Estimation (KDE) [36] and selecting the best using *best-of-N sampling*. Our approach combines a manifold-sensitive kernel with multi-hypothesis sampling from the diffusion process. Given an observation $\mathbf{o}_t$, we sample N independent and identically distributed action trajectories $\{\mathbf{A}_i\}_{i=1}^N \sim p_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{o}_t)$, where $\mathbf{A}_i \in \mathbb{R}^{T\times D}$, and estimate the PDF of the last action of the trajectories, hereinafter defined as $\mathbf{a}_i \in \mathbb{R}^D$, via KDE. This allows to obtain the probability density of every action and use it to discard outlier trajectories.

To model the PDF, KDE requires a unified kernel over all the action components representing end-effector position and orientation, and gripper aperture. While the multivariate Gaussian kernel is

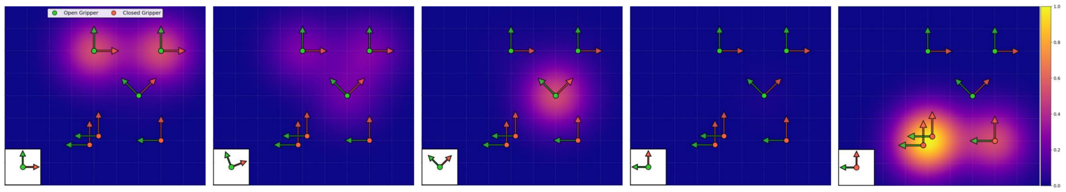

Figure 1: Visualization of the PDF estimated via KDE with the proposed manifold-aware kernel. We perform KDE on a population of 6 planar end-effector actions represented as reference frames in the plots. Three of them represent open grippers (green circle), while the other three represent closed grippers (red circle). The color of each point in the heatmaps represents the density value of an action at the corresponding 2D location, that has orientation and gripper state showed by the indicator frame in the white square of each plot. From left to right we vary the rotation of the indicator frame from 0 to 90 degrees and observe that the densities returned by KDE at different locations vary accordingly, spiking when the probed actions are close to the ones used for PDF modeling. The plots show how KDE correctly handles multimodality by providing the highest density values for the most well represented samples. The two rightmost plots show how the gripper state is correctly handled by the KDE.

a natural choice for Euclidean domains, it cannot directly handle data lying on different manifolds such as rotations in $\mathrm{SO}(3)$, being a non-Euclidean space.

Manifold-Aware Kernel Density Estimation We propose a modification of the probability distribution presented in [4] to handle the pose of the end-effector together with the gripper state. Although DP outputs orientation in the 6D matrix representation introduced in [37], for the sake of clarity, we consider their matrix representation in $\mathrm{SO}(3)$. Our kernel function is defined by the equation

$$k(\mathbf{a}_i, \mathbf{a}_j) = \frac{1}{\sqrt{(2\pi)^D |\mathbf{H}|}} \exp\left(-\frac{1}{2}\boldsymbol{\Delta}_{ij}^\top \mathbf{H}^{-1} \boldsymbol{\Delta}_{ij}\right), \tag{3}$$

where $\mathbf{a}_i = [\mathbf{t}_i; \mathbf{R}_i; \mathbf{g}_i]$ and $\mathbf{a}_j = [\mathbf{t}_j; \mathbf{R}_j; \mathbf{g}_j]$ are actions composed of position, rotation, and gripper aperture components. $\mathbf{H}$ is the covariance matrix defined as $\mathbf{H} = \mathrm{diag}\big(\sigma^2_{\text{pos}}\mathbf{I}_3,\, \sigma^2_{\text{rot}}\mathbf{I}_3,\, \sigma^2_{\text{grip}}\mathbf{I}_1\big)$, and $\boldsymbol{\Delta}_{ij} \in \mathbb{R}^D$ represents the difference between manifold-specific components defined as:

$$\boldsymbol{\Delta}_{ij} = [\underbrace{\mathbf{t}_i - \mathbf{t}_j}_{\text{Euclidean}}; \underbrace{\log(R_j^T R_i)^\vee}_{\mathrm{SO}(3)}; \underbrace{\mathbf{g}_i - \mathbf{g}_j}_{\text{Euclidean}}] \tag{4}$$

For position and gripper components, standard Euclidean differences suffice. For rotations represented as rotation matrices $R_i, R_j \in \mathrm{SO}(3)$, we compute the transformation from R_j to R_i and convert it to its axis-angle representation using the Lie group logarithm map $\log : \mathrm{SO}(3) \to \mathfrak{so}(3)$ followed by the vee operator $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ to obtain a representation in the tangent space. From the definitions above, we can rewrite $-\frac{1}{2}\boldsymbol{\Delta}_{ij}^\top \mathbf{H}^{-1} \boldsymbol{\Delta}_{ij}$ as

$$-\frac{1}{2}\left(\frac{\|\mathbf{t}_i - \mathbf{t}_j\|^2}{\sigma^2_{\text{pos}}} + \frac{\|\log(R_j^\top R_i)^\vee\|^2}{\sigma^2_{\text{rot}}} + \frac{(\mathbf{g}_i - \mathbf{g}_j)^2}{\sigma^2_{\text{grip}}}\right), \tag{5}$$

where $\|\log(R_j^\top R_i)^\vee\|^2$ is the geodesic distance representing the minimal rotation angle θ_{ij} required to align R_i and R_j:

$$d_{SO(3)}(R_i, R_j) = \|\log(R_j^\top R_i)^\vee\|^2 = \|\boldsymbol{\theta_{ij}}\| = \theta_{ij}.$$

In Fig. 1, we show how the proposed manifold-aware kernel allows to model PDFs for actions comprising end-effector position and orientation, and gripper aperture.

Density Estimation and Action Selection We model the PDF underlying the N actions $\{\mathbf{a}_i\}_{i=1}^N$ predicted by DP with KDE, by combining kernel densities across all actions. Specifically, we compute the density of a generic action $\tilde{\mathbf{a}}$ as:

$$\rho(\tilde{\mathbf{a}}) = \frac{1}{N}\sum_{i=1}^{N} k(\tilde{\mathbf{a}}, \mathbf{a}_i) \tag{6}$$

Finally, we compute densities $\rho_i = \rho(\mathbf{a}_i)$ for each of the N final actions predicted by DP and we select the trajectory $\mathbf{A}_i$ corresponding to the action $\mathbf{a}_i$ with the highest ρ_i.

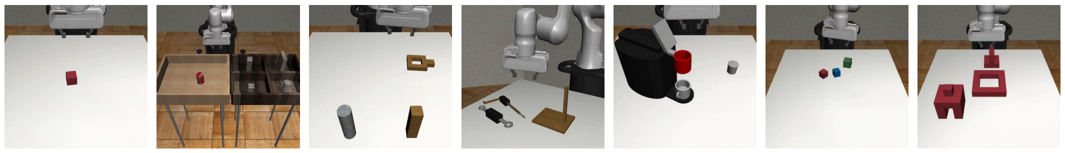

Figure 2: RoboMimic (*Lift*, *Can*, *Square* and *ToolHang*) and MimicGen (*Coffee*, *Stack* and *Assembly*) tasks considered for KDPE's evaluation.

4 Experimental Setup

4.1 Simulated Environments

We evaluate KDPE on seven tasks from two different benchmarks: RoboMimic [38] and MimicGen [9]. We consider four RoboMimic tasks: *Lift*, *Can*, *Square*, and *ToolHang* (illustrated in Fig. 2). This subset of RoboMimic tasks corresponds to the single-arm image-based tasks used to evaluate DP in [1]. RoboMimic provides two datasets for *Lift*, *Can*, and *Square*: one collected by proficient human operators (*ph*) and another by a mix of proficient and non-proficient operators (*mh*), resulting in lower data quality for *mh*. For *ToolHang*, only the *ph* dataset is available. We also test KDPE on three MimicGen tasks: *Coffee*, *Stack Three* (*Stack* for conciseness) and *Three Piece Assembly* (*Assembly*). We chose this benchmark because it shares the same structure as RoboMimic, making the integration with the evaluation framework seamless and allowing us to test KDPE on additional challenging tasks.

To assess the robustness of KDPE under visual domain shift, we compare its performance with DP on perturbed variations of the original environments where we slightly modify the color of an object. Specifically, we remove the original texture from specific objects and set their new color as the average value of the original texture with a decreased 10% lightness value under the HSL color scheme. The specific objects modified for this experiments are shown in App. A.

4.2 Comparison with Diffusion Policy

To compare KDPE with Diffusion Policy (DP), we train DP for $80k$ training steps. For each experimental setting, we rollout the methods for 100 random resets of the environment. The random sequence of environment initializations and the DDPM noise schedules are kept fixed across experiments. During policy rollout, we sample a population of 100 trajectories at every inference step, and one trajectory is selected differently for each method. The DP baseline uniformly samples the output trajectory from the population. For KDPE, we perform KDE on the eighth actions of the population of trajectories, as eight is the action execution horizon used in DP. If the selection method is non-deterministic, as it is the case for DP, we run the full set of rollouts three times and report the average result.

Additionally to the comparison with DP, we report performance on two modified versions of KDPE: KDPE-OOD and Tr-KDPE. KDPE-OOD is a modification of KDPE, where we choose the trajectory associated to the least represented action, i.e., the action with the minimum density $\{\rho_i\}_{i=1}^{N}$ (see Sec. 3.2). We evaluate this method to further support the need for outliers rejection in the output trajectories of DP. Tr-KDPE, instead, is a modification of KDPE which uses conditional KDE and a first order Markovian assumption to estimate the PDF of the population of whole trajectories. We report the details of Tr-KDPE in App. B.

We present results for both the CNN-based (DP-C, KDPE-C, KDPE-OOD-C, Tr-KDPE-C) and the Transformer-based (DP-T, KDPE-T, KDPE-OOD-T, Tr-KDPE-T) models. For all methods, we use the same hyperparameters, e.g. KDE kernel bandwidths, reported in App. C.

4.3 Real Robot

We collect datasets to train the policies by using a *Meta Quest 3* VR headset to teleoperate a *Franka Emika Panda* robotic arm equipped with a *Robotiq 2f-85* gripper. We provide visual RGB observa-

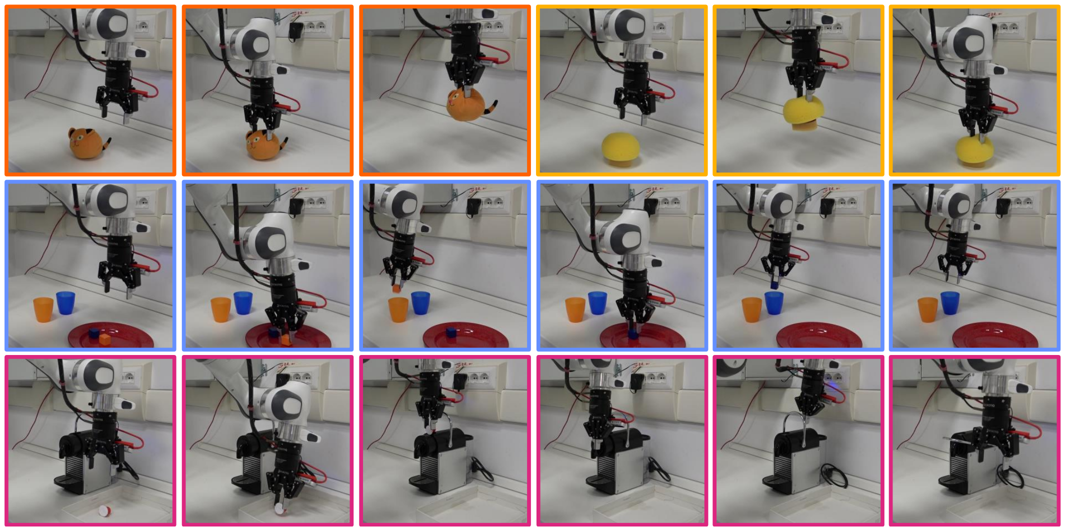

Figure 3: KDPE autonomously executing the real-robot tasks: *PickPlush* (orange border) and its variant *PickSponge* (yellow), *CubeSort* (blue) and *CoffeeMaking* (purple).

tions to the robot from an *Intel(R) RealSense D415* mounted on the wrist of the robot and an external *Intel(R) RealSense D405* (Fig. 3). We collect 50 demonstrations for *PickPlush*, 135 demonstrations for *CubeSort* and 200 demonstrations for *CoffeeMaking*.

We first test DP and KDPE for 50 episodes on *PickPlush* (Fig. 3), a picking task that consists of grasping and lifting a cat plush from the table. Additionally, mimicking the visual domain shift experiment performed in simulation, we test the same checkpoints on a version of *PickPlush* where the plush is replaced with a yellow sponge, named *PickSponge*. Then, we compare DP and KDPE for 100 episodes on *CubeSort*. The robot is required to grasp a blue and an orange cube from a red plate and place them in the cups of the corresponding colors. This requires the policy to learn a task with a high degree of multimodality both in the order in which the cubes are picked and in the way they are manipulated. In the task demonstrations, the two cubes are grasped in a random sequence and, during model evaluation, the cubes are randomly tossed on the plate at the beginning of each episode. Finally, we evaluate for 10 episodes the two methods on the *CoffeMaking* task, a real-world experiment to understand the impact of KDPE in a realistic scenario requiring long-horizon planning capabilities and considerable precision. The *CoffeeMaking* task requires the robot to complete the following steps: picking up a coffee pod, inserting the pod in the coffee machine with poor visual conditions, pushing the front part of the coffee machine to close it, and pulling down the metal handle. Some of these steps require high precision, e.g. for pod insertion, as it can be noticed from Fig. 3, where the task is autonomously performed by KDPE.

Following the methodology described in Section 4.2, we train the CNN-based version of the DP model for $80k$ steps from images of size 480×480 and we evaluate DP and KDPE on the same checkpoint, using DDIM [39] as noise scheduler to speed-up inference.

5 Results

5.1 Benchmark on RoboMimic and MimicGen Tasks

Results in the benchmark reported in Tab. 1 show that KDPE outperforms DP on both architectures, and that KDPE-C is overall the best-performing method, achieving an average success rate of 88.2%. KDPE improves DP by 3.3% when using the CNN backbone, and by 1.9% with the Transformer-based model. These gaps are even more pronounced (4.2% and 2.4%) if we exclude from the analysis the *Lift* task which saturates to a success rate of 100% for both methods. It is also worth noting that, in the three experiments where DP outperforms KDPE, the performance gap is limited

	Lift		Can		*Square*		*ToolHang*	*Coffee*	*Stack*	*Assembly*	Average
	ph	*mh*	*ph*	*mh*	*ph*	*mh*	*ph*				
DP-C	**100**	**100**	96.7	94.0	**93.0**	83.0	62.0	85.0	80.7	54.3	84.9
KDPE-OOD-C	**100**	**100**	96.0	84.0	83.0	49.0	7.0	85.0	72.0	40.0	71.6
Tr-KDPE-C	**100**	**100**	97.0	94.0	90.0	78.0	68.0	76.0	78.0	**61.0**	84.2
KDPE-C	**100**	**100**	**98.0**	**96.0**	92.0	**86.0**	**76.0**	**88.0**	**85.0**	**61.0**	**88.2**
DP-T	**100**	**100**	96.0	90.0	84.0	73.0	61.3	91.7	73.7	**23.3**	79.3
KDPE-OOD-T	**100**	99.0	95.0	91.0	84.0	65.0	2.0	**93.0**	61.0	23.0	71.3
Tr-KDPE-T	**100**	99.0	93.0	92.0	81.0	81.0	62.0	86.0	**74.0**	18.0	78.6
KDPE-T	**100**	**100**	**97.0**	**92.0**	**88.0**	**83.0**	**66.0**	91.0	72.0	23.0	**81.2**

Table 1: Average success rate (%) of DP, KDPE-OOD, Tr-KDPE and KDPE on RoboMimic and MimicGen. The last column reports the average success rate over all the tasks.

	Lift		*Can*		*Square*		*ToolHang*	*Coffee*	*Stack*	*Assembly*	Average
	ph	*mh*	*ph*	*mh*	*ph*	*mh*	*ph*				
DP-C	**100**	99.6	92.3	**91.3**	93.7	79.7	59.0	92.3	75.3	44.7	82.8
KDPE-C	**100**	**100**	**94.0**	90.0	**94.0**	**90.0**	**69.0**	**97.0**	**88.0**	**50.0**	**87.2**
DP-T	98.7	**100**	89.0	84.0	**79.3**	68.3	58.0	**73.3**	67.7	15.6	73.4
KDPE-T	**99.0**	**100**	**90.0**	**85.0**	75.0	**77.0**	**62.0**	72.0	**68.0**	**20.0**	**74.8**

Table 2: Success rate (%) of DP and KDPE under object color perturbation.

and the success rate of KDPE is always within the standard deviation over the three trials of DP (see App. D).

KDPE achieves a larger performance improvement w.r.t. DP on precision tasks as *ToolHang*, and on tasks learned from lower quality data (*mh*). The first observation indicates that the success rate of high-precision tasks is more influenced by outlier trajectories. The second, paired with the intuition that policies trained on noisy data learn to generate the outliers contained in the dataset, underlines the importance of our filtering mechanism. These findings are further supported by the performance of KDPE-OOD that, in a specular way, experiences its largest drops in performance on the tasks where KDPE performs best. The trajectory-based version of KDPE achieves similar performance to DP, resulting in no relevant improvement on either architecture. We attribute this to the fact that, to analyze the whole trajectory, Tr-KDPE needs a higher-dimensional kernel. This might lead to curse of dimensionality and poor characterization of the population's distribution. While the population size could be increased, it is important to notice that the required sample size increases exponentially with respect to dimensionality. This would lead to an exponential increase of the computational cost.

5.2 Analysis Under Visual Environment Perturbations

To study the robustness of DP and KDPE to domain shift, we perturb the model observations through the object color modification presented in Sec. 4.1, and measure the success rate on the same set of RoboMimic and MimicGen tasks considered in Sec. 5.1. Results in Tab. 2 show that, on average, KDPE outperforms DP with both CNN and Transformer-based models by a similar margin to that presented in Tab. 1 for the benchmark experiments in the unperturbed setting. This result further supports the effectiveness of KDPE in filtering out trajectory outliers, even in noisier settings.

5.3 Real Robot Results

We compare KDPE to DP on three real-world tasks to study whether the improvement in performance observed in the simulated tasks translates to better performance on the real robot. Results in Tab. 3 show that in the four experiments KDPE outperforms DP in terms of success rate. Moreover, for all the tasks, we noticed a smoother behavior of robot end-effector and gripper aperture. This suggests that KDPE allows to choose trajectories, representing task modalities, more consistently.

PickPlush We evaluate performance for 50 episodes on the two different task configurations described in Sec. 4.3. We initialize each episode with the plush in a random position. However, when one of the two methods fails to grasp the object, we perform a rollout of the other method starting

	PickPlush	*PickSponge*	*CubeSort*	*CoffeeMaking*
DP-C	90	88	41	60
KDPE-C	**96**	**90**	**44**	**70**

Table 3: Success rate (%) of DP-C and KDPE-C on the real-world tasks. We test *PickPlush* and *PickSponge* for 50 episodes, *CubeSort* for 100 episodes and *CoffeeMaking* for 10 episodes.

from the same object position. KDPE solves the task with the orange plush, *PickPlush*, 48 times, while DP 45 (96% vs. 90% success rate), being in line with the KDPE-C results in Tab. 1. In the three failure cases of DP, the choice of a suitable trajectory with KDPE has been critical to solve the task. We also test the same model (trained only to grasp the orange plush) with the yellow sponge (*PickSponge*). Similarly to the experiments under object color perturbation presented in Sec. 5.2, KDPE outperforms DP, but the gap in performance w.r.t. *PickPlush* reduces. This may indicate that, on the failed episodes with the yellow sponge, DP (and therefore KDPE) could not predict suitable in-distribution trajectories.

CubeSort We evaluate the two algorithms for 100 episodes, by randomly tossing the two cubes on the plate at the beginning of each episode. Differently from *PickPlush*, we could not test the methods on the same object positions, as the failure state of the task is often partial (i.e., only one cube is positioned in the matching cup). KDPE achieves a higher success rate (3% higher) which reflects the results observed in simulation for KDPE-C, e.g. on the *Stack* task which requires to manipulate cubes of a similar size.

CoffeeMaking We report performance for the *CoffeeMaking* task for 10 episodes. We noticed that, for this task, the initial position of the pod on the table has a huge impact on the final success of the task. Therefore, for a fair comparison, we tested both methods with the same pod initializations. KDPE managed to solve the task seven times, while DP six.

5.4 Inference Time

We measure the inference time of DP-C, KDPE-C and Tr-KDPE-C with DDIM sampling on the machine used for the real-world experiments (equipped with an NVIDIA RTX 3080 GPU). DP-C, which in the original implementation generates a single trajectory, requires 77ms, while generating a population of 100 trajectories takes an additional 13ms, and performing KDE on them adds only 3ms. Therefore, DP-C can be executed at a control frequency of $\sim$ 12.99Hz, while KDPE-C at $\sim$ 10.75Hz, meaning that KDPE-C adds a computational overhead of only $\sim$ 2.24Hz. For Tr-KDPE-C, instead, performing KDE requires 30ms, and can be executed at $\sim$ 8.33Hz.

6 Conclusion

DP has recently gained popularity as one of the most effective methods to train behavior cloning policies, thanks to its policy parameterization as a DDPM which allows to model the multimodality in the training data. While being effective, however, the sampling process is not constrained in any way. This can lead the model to predict trajectories that take the robot out-of-distribution. We propose to overcome this limitation with KDPE, a strategy to select with KDE the most representative trajectory computed by different DP denoising processes.

We quantitatively benchmarked KDPE against DP on four RoboMimic and three MimicGen simulated tasks, and on three real robot experiments: a tabletop plush picking, a multimodal and multi-step cube sorting, and a long-horizon coffe making task that requires high precision to be completed.

We showed that KDPE achieves better performance in the lower demonstration-quality regime and on tasks that require higher precision, also in presence of visual environment perturbations.

Two interesting directions for future research are using KDPE to guide the denoising process of DP, and applying KDPE to higher-dimensional problems, such as bimanual or dexterous manipulation tasks with anthropomorphic hands. However, such applications could incur the additional challenge of addressing the curse of dimensionality for KDE, a known challenge for kernel-based methods.

7 Limitations

Limiting assumptions KDPE selects certain output trajectories among the ones produced by DP. If this latter does not generate a representative distribution of trajectories, e.g. due to poor training, KDPE cannot provide any performance improvement.

Failure modes One of the strengths of DP is that the stochasticity of the trajectory generation can help the robot recover from out-of-distribution states by generating trajectories that may be associated to smaller probability density. KDPE aims to reduce the probability that the robot ends up in such states. However, detecting when this happens (e.g., from the estimated PDF from KDE) and switching to a different trajectory selection method would be an interesting improvement to KDPE.

Limitations of the results and experiments In this paper we proposed KDPE as a component that can be plugged on top of DP. However, it would be interesting to apply KDPE to other generative models like flow matching [21], or to generalist policies [11, 18, 12].

Acknowledgements

This research was supported by the PNRR MUR project PE0000013-FAIR, by the National Institute for Insurance against Accidents at Work (INAIL) project ergoCub-2.0, and by the Brain and Machines Flagship Programme of the Italian Institute of Technology.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[2] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[3] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 464–465. IEEE, 1988.

[4] T. D. Barfoot and P. T. Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014. doi:10.1109/TRO.2014.2298059.

[5] I. Gilitschenski, G. Kurz, S. J. Julier, and U. D. Hanebeck. A new probability distribution for simultaneous representation of uncertain position and orientation. In *17th International Conference on Information Fusion (FUSION)*, pages 1–7, 2014.

[6] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi:10.1109/34.1000236.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[8] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, pages 1678–1690. PMLR, 2022.

[9] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[10] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022. doi: 10.1109/MRA.2021.3138382.

[11] D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, Q. Vuong, T. Xiao, P. R. Sanketi, D. Sadigh, C. Finn, and S. Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi:10.15607/RSS.2024.XX.090.

[12] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[13] K. Bousmalis, G. Vezzani, D. Rao, C. M. Devin, A. X. Lee, M. B. Villalonga, T. Davchev, Y. Zhou, A. Gupta, A. Raju, A. Laurens, C. Fantacci, V. Dalibard, M. Zambelli, M. F. Martins, R. Pevceviciute, M. Blokzijl, M. Denil, N. Batchelor, T. Lampe, E. Parisotto, K. Zolna, S. Reed, S. G. Colmenarejo, J. Scholz, A. Abdolmaleki, O. Groth, J.-B. Regli, O. Sushkov, T. Rothörl, J. E. Chen, Y. Aytar, D. Barker, J. Ortiz, M. Riedmiller, J. T. Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=vsCpILiWHu`.

[14] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[15] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[16] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[17] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[18] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[19] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[20] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[21] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *7th Annual Conference on Robot Learning*, 2023.

[22] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[23] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[24] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[25] X. Zhang, M. Chang, P. Kumar, and S. Gupta. Diffusion meets dagger: Supercharging eye-in-hand imitation learning. *arXiv preprint arXiv:2402.17768*, 2024.

[26] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[27] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 689–723. PMLR, 2025.

[28] S. Sagar, J. Duan, S. Vasudevan, Y. Zhou, H. B. Amor, D. Fox, and R. Senanayake. From mystery to mastery: Failure diagnosis for improving manipulation policies. *arXiv preprint arXiv:2412.02818*, 2025.

[29] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=6FGlpzC9Po.

[30] Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=j7DZWSc8qu.

[31] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[32] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[33] N. Ma, S. Tong, H. Jia, H. Hu, Y.-C. Su, M. Zhang, X. Yang, Y. Li, T. Jaakkola, X. Jia, and S. Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[34] S. Li, K. Kallidromitis, A. Gokul, A. Koneru, Y. Kato, K. Kozuka, and A. Grover. Reflect-dit: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. *arXiv preprint arXiv:2503.12271*, 2025.

[35] E. Xie, J. Chen, Y. Zhao, J. Yu, L. Zhu, C. Wu, Y. Lin, Z. Zhang, M. Li, J. Chen, H. Cai, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025.

[36] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[37] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019. doi:10.1109/CVPR.2019.00589.

[38] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

[39] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

[40] R. J. Hyndman, D. M. Bashtannyk, and G. K. Grunwald. Estimating and visualizing conditional densities. 5(4):315–336. ISSN 1061-8600, 1537-2715. doi:10.1080/10618600.1996.10474715. URL http://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474715.

[41] Rerun Development Team. Rerun: A visualization sdk for multimodal data, 2024. URL https://www.rerun.io. Available from https://www.rerun.io/ and https://github.com/rerun-io/rerun.

A Analysis Under Visual Environment Perturbations

In the experiments under visual environment perturbation presented in Sec. 4.1, we slightly modify the color of an object in each environment. Specifically, if the object has a texture we compute its average color, we reduce its lightness (third channel under the HSL color scheme) by 10% and set it as the object color. The object modified in the simulated task environments are shown in Fig. 4.

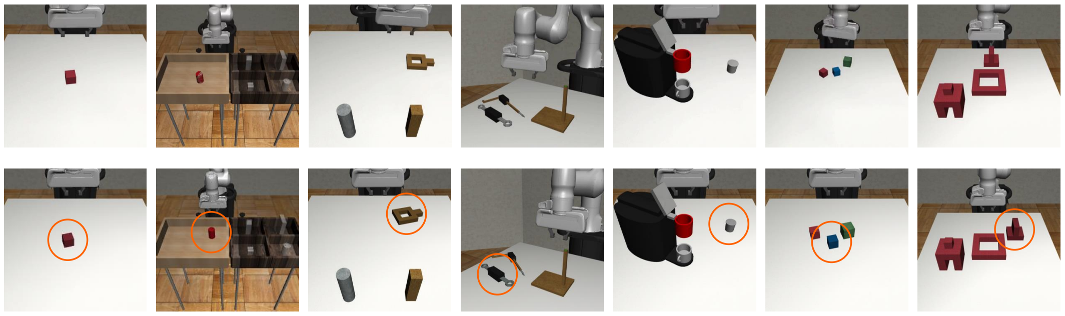

Figure 4: First row: unperturbed task environments. Second row: perturbed environments with the objects modified in the color perturbation experiments highlighted in the orange circles.

B Definition of Tr-KDPE

Tr-KDPE (Trajectory-KDPE) is a modification of the KDPE algorithm that estimates the probability density function (PDF) over a population of complete action trajectories. We recall that KDPE instead selects action trajectories based only on the density associated to the final action. Therefore, Tr-KDPE mitigates the risk of favoring outlier trajectories that, while ending in an in-distribution state for KDPE, may overall represent out-of-distribution trajectories.

To model the joint PDF $p(\mathbf{a}_1, ..., \mathbf{a}_T)$ of the entire action sequence, Tr-KDPE uses the Markov assumption to decompose this joint probability into a product of conditional probabilities: $p(\mathbf{a}_1, ..., \mathbf{a}_T) = p(\mathbf{a}_1) \prod_{t=2}^{T} p(\mathbf{a}_t|\mathbf{a}_{t-1})$. The estimation relies on modeling the transition probabilities $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ with conditional Kernel Density Estimation [40]:

$$\hat{f}(y|x) = \frac{\hat{g}(x,y)}{\hat{h}(x)}, \tag{7}$$

where $\hat{g}(x,y)$ is the KDE of the joint density $g(x,y)$

$$\hat{g}(x,y) = \frac{1}{N} \sum_{j=1}^{N} k\left(x, x_j\right) k\left(y, y_j\right) \tag{8}$$

and $\hat{h}(x)$ is the KDE of the marginal density $h(x)$

$$\hat{h}(x) = \frac{1}{N} \sum_{j=1}^{N} k\left(x, x_j\right). \tag{9}$$

Therefore, we can compute the kernel density estimator of $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ as:

$$\rho(\mathbf{a}_t|\mathbf{a}_{t-1}) = \frac{\hat{g}(\mathbf{a}_{t-1}, \mathbf{a}_t)}{\hat{h}(\mathbf{a}_{t-1})}. \tag{10}$$

Consequently, estimating the full trajectory density $p(\mathbf{a}_1, ..., \mathbf{a}_T)$ involves computing one standard KDE for $p(\mathbf{a}_1)$ and $T-1$ conditional KDEs for the terms $t = 2, ..., T$.

C Hyperparameters

The hyperparameters used during training and evaluation are reported in Tab. 4.

Hyperparameter	*Value*
Optimizer	Adam
Learning Rate	1×10^{-4}
Betas	$(0.95, 0.999)$
Epsilon	1×10^{-8}
Weight Decay	1×10^{-6}
Noise scheduler	DDPM (simulation) / DDIM (real robot)
Diffusion Inference Steps	100 (simulation) / 10 (real robot)
Total Training Steps	80,000
Population Size (N)	100
Execution Horizon (T)	8
Action Dimensionality (D)	10
Position Bandwidth (σ_{pos})	0.05
Rotation Bandwidth (σ_{rot})	0.25
Gripper Bandwidth (σ_{grip})	1.0

Table 4: Hyperparameters used for training and evaluation of KDPE. Symbols in the parentheses correspond to those used in Sec. 3.

D DP Standard Deviations

As reported in Sec. 3.2, since the baseline performance (DP-C or DP-T) depends on the random sampling of a trajectory, we repeated its evaluation three times with three different seeds. We report the performance of all the methods on the benchmark experiments with DP standard deviations in Tab. 5, and on the experiments under object color perturbation in Tab. 6.

	Lift		Can		*Square*		*ToolHang*
	ph	*mh*	*ph*	*mh*	*ph*	*mh*	*ph*
DP-C	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	96.7 ± 1.00	94.0 ± 2.00	$\mathbf{93.0 \pm 3.00}$	83.0 ± 7.00	62.0 ± 2.00
KDPE-OOD-C	**100**	**100**	96.0	84.0	83.0	49.0	7.00
Tr-KDPE-C	**100**	**100**	97.0	94.0	90.0	78.0	68.0
KDPE-C	**100**	**100**	**98.0**	**96.0**	92.0	**86.0**	**76.0**
DP-T	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	96.0 ± 2.00	90.0 ± 3.00	84.0 ± 4.00	73.0 ± 7.00	61.3 ± 10.0
KDPE-OOD-T	**100**	99.0	95.0	91.0	84.0	65.0	2.00
Tr-KDPE-T	**100**	99.0	93.0	92.0	81.0	81.0	62.0
KDPE-T	**100**	**100**	**97.0**	**92.0**	**88.0**	**83.0**	**66.0**

	Coffee	*Stack*	*Assembly*
DP-C	85.0 ± 6.00	80.7 ± 1.00	54.3 ± 1.00
KDPE-OOD-C	85.0	72.0	40.0
Tr-KDPE-C	76.0	78.0	**61.0**
KDPE-C	**88.0**	**85.0**	**61.0**
DP-T	91.7 ± 6.00	73.7 ± 2.00	$\mathbf{23.3 \pm 2.00}$
KDPE-OOD-T	**93.0**	61.0	23.0
Tr-KDPE-T	86.0	**74.0**	18.0
KDPE-T	91.0	72.0	23.0

Table 5: Benchmark results presented in Tab. 1, reporting also the standard deviation of the success rate (%) over three runs of DP. Top table: RoboMimic (*Lift*, *Can*, *Square*, *ToolHang*). Bottom table: MimicGen (*Coffee*, *Stack*, *Assembly*).

	Lift		Can		*Square*		*ToolHang*
	ph	*mh*	*ph*	*mh*	*ph*	*mh*	*ph*
DP-C	$\mathbf{100 \pm 0}$	99.6 ± 0.57	92.3 ± 4.16	$\mathbf{91.3 \pm 3.21}$	93.7 ± 2.52	79.7 ± 4.16	59.0 ± 1.53
KDPE-C	**100**	**100**	**94.0**	90.0	**94.0**	**90.0**	**69.0**
DP-T	98.7 ± 0.58	$\mathbf{100 \pm 0}$	89.0 ± 4.58	84.0 ± 3.61	$\mathbf{79.3 \pm 1.53}$	68.3 ± 2.51	58.0 ± 5.57
KDPE-T	**99.0**	**100**	**90.0**	**85.0**	75.0	**77.0**	**62.0**

	Coffee	*Stack*	*Assembly*
DP-C	92.3 ± 5.03	75.3 ± 0.58	44.7 ± 1.50
KDPE-C	**97.0**	**88.0**	**50.0**
DP-T	$\mathbf{73.3 \pm 7.37}$	67.7 ± 4.16	15.6 ± 4.16
KDPE-T	72.0	**68.0**	**20.0**

Table 6: Results of experiments under object color perturbation presented in Tab. 2, reporting also the standard deviation of the success rate (%) over three runs of DP. Top table: RoboMimic (*Lift*, *Can*, *Square*, *ToolHang*). Bottom table: MimicGen (*Coffee*, *Stack*, *Assembly*).

E Visualizer

During the development and study of KDPE and comparative baselines, we developed a visualizer using the *rerun* data visualization library [41]. This tool facilitates the analysis of populations of trajectories, including action positions, orientations, and gripper states. Through the help of the visualizer, we studied how the KDE bandwidths (reported in the last three rows of Tab. 4) influence the action densities, and found suitable values that let KDE capture DP output multimodality. Fig. 5 shows a snapshot of the visualizer.

We open-source the code of the visualizer, hoping it will assist others in the analysis of generative robotic policies. The code for the visualizer is available on the project page at `https://hsp-iit.github.io/KDPE/`.

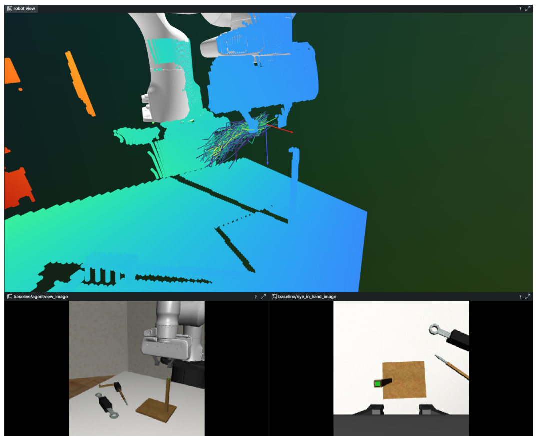

Figure 5: The trajectory visualizer being used to analyze trajectories on the RoboMimic *ToolHang* task. The scene in the 3D view (*robot view* window) is represented as a point-cloud, since object meshes are not readily available for real-world environments. The visualizer supports assigning different colormaps to the population of trajectories. The colormap in the picture represents the densities assigned by KDPE to each trajectory.