Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Fast Flow-based Visuomotor Policies via Conditional Optimal Transport Couplings

Andreas Sochopoulos, Nikolaos Tsagkas, Joao Moura, Nikolay Malkin, Michael Gienger, Sethu Vijayakumar

2025

Preprint:

This is an accepted article published in Conference on Robot Learning. The final authenticated version is available online at: https://doi.org/[DOI not available]

Fast Flow-based Visuomotor Policies via Conditional Optimal Transport Couplings

Andreas Sochopoulos[1,2], **Nikolay Malkin**[1], **Nikolaos Tsagkas**[1], **João Moura**[1], **Michael Gienger**[2], **Sethu Vijayakumar**[1]
[1]University of Edinburgh [2]Honda Research Institute Europe
ansocho.github.io/cot-policy

Abstract: Diffusion and flow matching policies have recently demonstrated remarkable performance in robotic applications by accurately capturing multimodal robot trajectory distributions. However, their computationally expensive inference, due to the numerical integration of an ODE or SDE, limits their applicability as real-time controllers for robots. We introduce a methodology that utilizes conditional Optimal Transport couplings between noise and samples to enforce straight solutions in the flow ODE for robot action generation tasks. We show that naively coupling noise and samples fails in conditional tasks and propose incorporating condition variables into the coupling process to improve few-step performance. The proposed few-step policy achieves a 4% higher success rate with a 10× speed-up compared to Diffusion Policy on a diverse set of simulation tasks. Moreover, it produces high-quality and diverse action trajectories within 1–2 steps on a set of real-world robot tasks. Our method also retains the same training complexity as Diffusion Policy and vanilla Flow Matching, in contrast to distillation-based approaches.

Keywords: Flow Matching, Optimal Transport, Imitation Learning

1 Introduction

Continuous-time generative models, such as Diffusion Models (DM) [1] and continous normalizing flows trained by flow matching (FM) [2], have recently proven to be an excellent choice for imitation learning of visuomotor robot policies from datasets of expert demonstrations [3, 4, 5]. This success stems from these models' ability to capture high-dimensional multimodal distributions, in this case, over actions conditioned on observations. Conversely, explicit models [6, 7, 8] try to learn a direct mapping between robot observations and actions, most often using Behavior Cloning (BC).

Continuous-time generative models come with a high computational cost for sampling actions, as inference requires numerically solving an ODE [9, 2] or SDE [10]. The cost of generating a sample grows with the number of time discretization steps used for integration, and the number of steps needed to generate high-quality samples may be large. ODEs can typically produce accurate actions in fewer integration steps than SDEs, but even 20 steps [11] is prohibitive for real-time inference. Over the past years, there have been significant efforts to reduce the computational cost of inference in continuous-time generative models. These efforts have been initiated mostly in the context of image generation [12, 13, 14].

The high latency of action generation using DM/FM policies has a direct impact on their real-world applicability, since it significantly reduces the frequency with which the policy can provide new actions to the robot. This in turn can lead to intermittent robot motions [5], where the robot follows a short-horizon trajectory and then stops to compute the next trajectory, or inability to solve tasks within dynamic environments (*e.g.*, manipulating deformable objects [15]).

Recently, interest has also emerged in accelerating robot policies [11, 16, 17]. The most common methodology for learning high-quality one-step models is by using distillation techniques that amor-

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

tize integration of an ODE over a time interval into a single-pass computation [12, 13, 14]. However, distillation requires a trained expert or multiple sequential re-trainings of a flow model [18], significantly increasing the time required to train high-quality one-step models. This can be prohibitive when training large policies on rich datasets with many high-dimensional exteroceptive robot observations.

In this paper, we present *COT Policy*, a FM methodology for training high-quality few-step visuomotor policies which preserve multimodality without a need for additional training phases. Our method is based on the use of Optimal Transport (OT) couplings [19] between noise and target actions. We demonstrate that naively using unconditional OT in FM for training conditional robot policies yields biased flows and is sometimes even worse than vanilla FM. Instead, we reformulate the OT problem to account for joint distributions of target samples and conditions, similar to recent works in conditional OT (COT) couplings [20]. Our main claims and contributions are the following:

(1) We introduce COT couplings and a technique for handling continuous conditions by dimensionality reduction and clustering.
(2) We use COT to train flow-based policies that outperform standard FM and OT-CFM in few-step performance on several robotic tasks.
(3) We analyze the diversity of sampled trajectories and show that COT Policy maintains the multimodality of the action distribution necessary for robust control at low inference cost.

2 COT Policy

In this section, we review the preliminaries on flow matching as relevant to our setting and present the proposed approach. Additional background and related work can be found in Appendix A.

2.1 Preliminaries: Conditional Flow Matching for Generative Modeling

Neural ODE generative models. The problem of generative modeling with continuous normalizing flows, or neural ODEs [21], asks to find an ODE that transforms an initial noise distribution p_0 over $\mathbb{R}^d$ into a target distribution p_1, where samples from both p_0 and p_1 are available. Given an ODE

$$dx = v_\theta(t, x)\, dt, \tag{1}$$

where $v_\theta : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is a vector field parametrized by a neural network with parameters θ taking x and t as input, one aims to find θ such that if $x(0) = x_0 \sim p_0$, then the distribution over $x(1)$ induced by integration of the ODE (1) from $t = 0$ to $t = 1$ with initial conditions $x(0)$ matches p_1. In our setting, p_0 is a fixed Gaussian noise distribution $N(0, I_d)$, so approximate samples from p_1 can be generated from the trained model by sampling $x(0) \sim \mathcal{N}(0, I_d)$ and integrating the ODE (*e.g.*, by `euler` integration). (The time variable t in the ODE is part of the probabilistic model and is not to be confused with the robot timestep τ, to be introduced later.)

Conditional flow matching. Conditional flow matching (CFM) assumes the choice of two objects: a *coupling* distribution $q(x_0, x_1)$ whose marginals over x_0 and x_1 equal p_0 and p_1, respectively, and a choice of an interpolating path from x_0 to x_1 for every (x_0, x_1). The latter is an ODE $dx = u(t, x \mid x_0, x_1)\, dt$ whose solution with initial conditions $x(0) = x_0$ has $x(1) = x_1$. In this paper, we always use a linear interpolant: $u(t, x \mid x_0, x_1) = x_1 - x_0$, yielding $x(t) = tx_1 + (1 - t)x_0$, so the solution moves from x_0 to x_1 along a line segment at a uniform rate as t moves from 0 to 1.

In CFM with linear interpolants, the network v_θ is trained through the stochastic regression objective:

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathop{\mathbb{E}}_{\substack{t\sim\mathcal{U}([0,1]) \\ x_0,x_1\sim q(x_0,x_1)}} \Big\| \underbrace{v_\theta(t, \overbrace{tx_1 + (1-t)x_0}^{\text{interpolant}})}_{\text{learned vector field}} - \underbrace{(x_1 - x_0)}_{\text{target}} \Big\|^2. \tag{2}$$

The key mathematical fact making this objective correct is that (under smoothness conditions that we always assume to be satisfied), the vector field v_θ that minimizes the loss (2) among all time-conditional vector fields transforms the initial distribution p_0 at $t = 0$ into the target distribution p_1 at $t = 1$, that is, solves the generative modeling problem introduced above. Notably, training with the objective (2) does not require any ODE integration, making CFM a *simulation-free* algorithm.

Optimal transport couplings. Different choices exist for the coupling $q(x_0, x_1)$. In Independent Coupling CFM (I-CFM), which we use interchangeably with CFM for the rest of the paper, $q(x_0, x_1) = q(x_0)q(x_1)$. Another option, which was shown in [19, 22] to reduce objective variance and straighten integration curves, takes q to be a minibatch optimal transport (OT) plan computed from batches of samples x_0, x_1. Such a coupling approximates the OT plan between p_0 and p_1.

We next recall some preliminaries about OT. The static OT problem aims to find a coupling of minimal cost between two distributions. Given a cost function $C(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we search for a solution to the optimization problem:

$$\mathrm{OT}(p_0, p_1) = \inf_{\pi \in \Pi(p_0, p_1)} \mathbb{E}_{(x_0, x_1) \sim \pi}[C(x_0, x_1)], \tag{3}$$

where $\Pi(p_0, p_1)$ denotes the space of probability measures whose left and right marginals equal p_0 and p_1, respectively. Intuitively, we seek a way to (stochastically) transport particles distributed according to p_0 so as to make them distributed according to p_1, minimizing the total cost of transportation. The infimum in (3) is called the OT cost, and a minimizer is called an OT plan.[1] In the case where $C(x_0, x_1) = ||x_0 - x_1||^2$ is the squared Euclidean distance, the square root of the distance in (3) is called the 2-Wasserstein distance and denoted $W_2(p_0, p_1)$.

Under basic conditions, the 2-Wasserstein OT plan exists, is unique up to a null set, and *deterministically* transports $x_0 \sim p_0$ to $x_1 \sim p_1$. Describing this transport plan as the solution of an ODE, which translates particles from x_0 to x_1 through time, leads to the dynamic form of the OT problem, also called the Benamou-Brenier form [24]. We refer to [19] for details, but remark that (1) the dynamic OT plan moves points in a straight line segment at uniform rate from x_0 to x_1; (2) if $q(x_0, x_1)$ is taken to be the 2-Wasserstein OT plan, then the minimizer of the CFM objective (2) among all vector fields is precisely the dynamic OT ODE. These properties motivate the use of approximate OT plans as the coupling $q(x_0, x_1)$: the resulting integration paths are more straight and thus the ODE can be solved to suitable precision with fewer `euler` integration steps.

Conditional generative modeling with CFM. The CFM framework can be extended to conditional generative modeling. If the distribution p_1 varies with a additional variable $c \in C$, we simply use a coupling $q(x_0, x_1 \mid c)$ whose marginals are p_0 and $p_1(\cdot \mid c)$ and fit a conditional vector field $v_\theta(x, t \mid c)$, integration of which from $t = 0$ to $t = 1$ would transform the noise distribution p_0 to the conditional target $p_1(\cdot \mid c)$. A conditional form of the static OT problem exists, which aims to minimize transport cost for all values of the condition c simultaneously [25], and Kerrigan et al. [20] introduced an equivalent Benamou-Brenier dynamic form for the conditional problem, which would be solved by performing CFM with conditional OT couplings $q(x_0, x_1 \mid c)$.

The technical challenge that motivates this paper is that when the condition variables c differ for all samples in a dataset, OT couplings are no longer possible to construct, since we cannot construct batches of samples that share the same condition c and use them to approximate the OT plan from p_0 to $p_1(\cdot \mid c)$; see §2.2.

Imitation learning as conditional generative modeling. As will be described in detail in §2.3, we treat the learning of actions given expert demonstrations as a conditional generative modeling problem. Assume a dataset of observation-action sequence pairs $\mathcal{D} = \{(o^{(i)}, a^{(i)})\}_{i=1}^n$ is given. We treat the $a^{(i)}$ as samples from a target distribution $p_1(a \mid o)$ with conditions $o = o^{(i)}$. Approximating this distribution by learning a neural ODE, conditioned on o, that transforms p_0 to $p_1(\cdot \mid o)$, would allow sampling actions a given observations o not seen in training.

In our setting, o and a represent observation histories and action sequences over short time horizons, respectively. The dataset $\mathcal{D}$ of short-horizon sequences is formed by extracting short sequences from expert demonstrations of longer duration. The observations consist of raw RGB images and proprioception [5, 12] but could also include point clouds [26] or force-torque data [27, 28].

[1] Under some conditions, *e.g.*, squared Euclidean cost and p_0 absolutely continuous with finite variance, the infinimum is achieved and the minimizer is unique [23].

2.2 Optimal Transport Couplings for Finite Conditional Tasks

For unconditional tasks, taking the coupling q in CFM to be the OT plan between p_0 and p_1 results in a flow that solves the dynamic OT problem. However, if we are aiming to sample a conditional distribution $p_1(\cdot \mid c)$, then naively taking q to be the OT plan between the noise distribution p_0 and p_1's marginal distribution over x_1 would not yield a solution to the conditional generative modeling problem, as can be seen in Fig. 1, where we train an ODE that transforms p_0 (mixture of Gaussians) to $p_1(\cdot \mid c)$ (two moons, where $c = 0, 1$ indexes the two halves of the moon). Such a naive use of OT-CFM causes noise samples from the top Gaussians to pair almost exclusively with targets from the top moon. During inference, while the paths remain nearly straight, sampling noise from one of the upper Gaussians with $c = 1$ leads to out-of-distribution targets. This occurs because few regression targets were established during training to connect the bottom moon to the upper Gaussians.

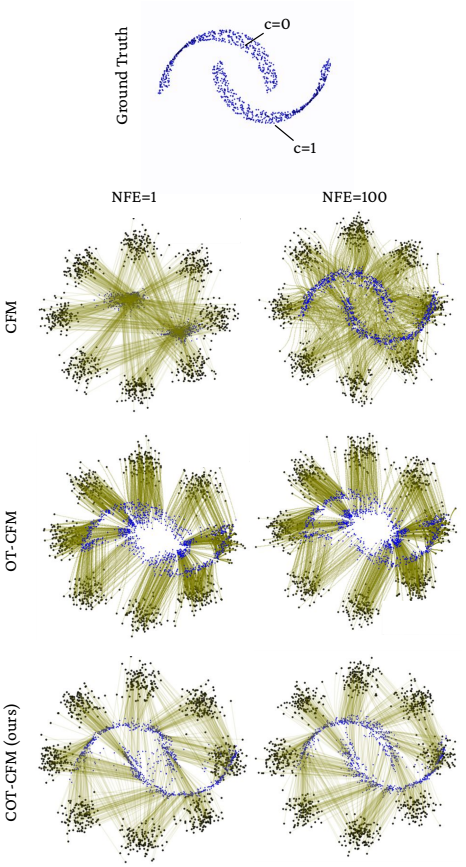

Figure 1: (I-)CFM, OT-CFM, and COT-CFM flows trained to generate the two moons distribution from the 8 Gaussians distribution. Generation with 100 (top row) and 1 (bottom row) `euler` integration steps is shown. CFM gives curved flows that cannot be integrated accurately in one step, while OT-CFM gives biased samples. Our proposed COT-CFM avoids both issues.

We first consider conditional tasks with condition variables (or observations) $c \in C$, where $C = \{c_1, c_2 \ldots, c_K\}$ is a finite set. Given a batch of N samples with conditions $B_1 = \{(x_1^{(1)}, c^{(1)}), (x_1^{(2)}, c^{(2)}), \ldots, (x_1^{(N)}, c^{(N)})\}$, where $x_1^{(i)} \in \mathbb{R}^d$ and $c^{(i)} \in C$, from the dataset, we construct a batch of noise observations $B_0 = \{(x_0^{(1)}, c_0^{(1)}), (x_0^{(2)}, c_0^{(2)}), \ldots, (x_0^{(N)}, c_0^{(N)})\}$, where the $x_0^{(i)}$ are independent samples from p_0 and $c_0^{(1)}, \ldots, c_0^{(N)}$ is a uniform-random permutation of $c^{(1)}, \ldots, c^{(N)}$. We then compute the empirical OT map between the empirical sets of sample-condition tuples B_0 and B_1, using the cost function

$$c((x_0, c_0), (x_1, c_1)) = \|x_0 - x_1\|^2 + \|\gamma(c_0 - c_1)\|^2 \tag{4}$$

where γ is a hyperparameter. This results in a empirical coupling $\pi(x_0, x_1)$, which is used for training the conditional vector field $v_\theta(t, x \mid c)$ conditioned on $c^{(i)}$ using the CFM loss:

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathop{\mathbb{E}}_{\substack{t\sim\mathcal{U}([0,1]) \\ (x_1^{(i)}, c^{(i)})\sim\mathcal{U}(B_1) \\ (x_0, c_0)\sim\pi(x_0 \mid x_1^{(i)})}} \Big\| \underbrace{v_\theta(t, \overbrace{tx_1^{(i)} + (1-t)x_0}^{\text{interpolant}} \mid \overbrace{c^{(i)}}^{\text{condition}})}_{\text{learned conditional vector field}} - \underbrace{(x_1^{(i)} - x_0)}_{\text{target}} \Big\|^2. \tag{5}$$

The c_0 are only used in computing the OT plan, and taking the c_0 to be a permutation of the observed conditions c_1 ensures that the two batches of conditions come from the same distribution. We term CFM augmented with this COT pairing as *COT-CFM* (conditional OT conditional flow matching).

The purpose of γ is to prioritize pairing of tuples with similar conditions. In the limit of $\gamma \to \infty$, the second term in (4) dominates over the first: samples (x_0, c_0) are paired with samples (x_1, c_1) with $c_0 = c_1$, and for each given c, π is the OT plan between the batches of x_0 and x_1 having condition c. On the other hand, if $\gamma = 0$, we recover OT-CFM, which ignores the conditions when computing the coupling even while the vector field takes c as an input, leading to biased conditional generation. We

can thus control the trade-off between an exact solution to conditional dynamic OT ($\gamma \to \infty$) and smoothness and generalization ($\gamma \to 0$) [20]. An investigation of the effect of γ on the solutions of minibatch OT can be found in Appendix E.1.

The COT-CFM algorithm is applicable even when the set of conditions C is not finite. However, as we explain next, when the conditions lie in high-dimensional continuous spaces, they will undergo dimensionality reduction and quantization before the OT is computed.

2.3 COT-CFM as a Robot Policy

Continuous condition tasks. In robotic tasks, the conditions for computing robot actions are derived from the robot's sensor observations. In our case, these observations o include RGB images and proprioceptive data, both of which take continuous values.

Intuitively, the conditional distribution of robot actions is not sensitive to small changes in the state of the environment and the robot. Small deviations in the robot's position or the workspace it operates in do not influence the plan of actions. Therefore, we can treat similar states or observations as a single condition in the calculation of $\pi(x_0, x_1)$. Effectively, we can obtain an approximation of the continuous conditional OT plan by discretizing the observations and applying the approximation to the conditional OT pairing described in §2.2. This involves the operations: $e = \mathcal{E}(o), \quad c = \mathcal{Q}(e)$, where $\mathcal{E}$ is an encoder for the observations and $\mathcal{Q}$ is a discretization function. The encoder is solely dependent on the modality of the observation, while the discretization function can be any suitable discretization scheme, such as a quantizer [29, 30] or a clustering algorithm.

Robot policies. Our pipeline for learning a generative policy includes encoding raw RGB observations using a vision encoder that is trained end-to-end with the policy, as is often done in imitation learning using generative models [5]. We implement $\mathcal{E}$ as PCA on each image batch, leaving the proprioceptive vector unchanged. For the discretizer $\mathcal{Q}$, we perform K-means clustering into K clusters and represent each point by the centroid of the cluster it belongs to. The resulting representations $c = \mathcal{Q}(\mathcal{E}(o))$ are used to compute the OT coupling, while the unprocessed observations o are used for conditioning the flow $v_\theta(t, x \mid o)$ (see Appendix B.1 and Appendix B.3 for details of the architecture of the model $v_\theta(t, x \mid o)$). This choice of $\mathcal{E}$ and $\mathcal{Q}$, although simple, is easy to extend to any observation modality, adds negligible computational overhead, and avoids the training of additional networks. The pipeline for our policy, called COT Policy, can be seen in Algorithm 1.

Algorithm 1 COT Policy training

Input: Noise distribution p_0, data-conditions dataset $\mathcal{D}$, condition weight γ, initial network v_θ, encoder $\mathcal{E}$, discretizer $\mathcal{Q}$.
while training **do**
  Sample data batch $\{(a^{(i)}, o^{(i)})\}_{i=1}^N \sim \mathcal{D}$, noise batch $\{x_0^{(i)}\}_{i=1}^N \sim p_0$, $\{t^{(i)}\}_{i=1}^N \sim \mathcal{U}(0,1)$
  // Noise-action pairing with COT
  $e^{(i)} \leftarrow \mathcal{E}(o^{(i)})$, $c_a^{(i)} \leftarrow \mathcal{Q}(e^{(i)})$
  $c_{x_0}^{(1,\dots,N)} \leftarrow \mathrm{randperm}(c_a^{(1,\dots,N)})$
  $\pi \leftarrow$ OT plan with cost (4)
    from $\{(x_0^{(i)}, c_{x_0}^{(i)})\}_{i=1}^N$ to $\{(a^{(i)}, c_a^{(i)})\}_{i=1}^N$
  $x_0^{(i)} \sim \pi(\cdot \mid x_1^{(i)})$ {paired initial point}
  // CFM loss and update
  $x_t^{(i)} \leftarrow t^{(i)} a^{(i)} + (1 - t^{(i)}) x_0^{(i)}$ {interpolant}
  Update θ with loss (5):
    $\sum_i \| v_\theta(t^{(i)}, x_t^{(i)} \mid o^{(i)}) - (a^{(i)} - x_0^{(i)}) \|^2$
end while

Training complexity. In contrast to distillation methods, which require an increased number of forward model passes [14] or two separate training phases [11, 16, 31], COT policies do not require extra training time. Although they include extra processing steps compared to their CFM counterparts, *i.e.*, performing dimensionality reduction and clustering, they add negligible computation time if implemented in a GPU-accelerated framework.

3 Experiments

We evaluate COT Policy's action-distribution learning capabilities and ability to capture multimodality. Details of demonstration datasets, and implementation can be found in Appendix B and Appendix C.

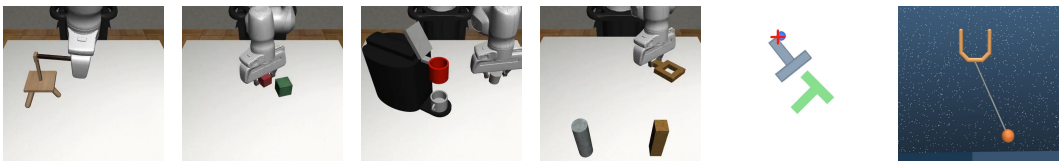

Figure 2: Tasks used for the evaluation of COT Policy.

Method	NFE	threading_d0	stack_d1	coffee_d1	square_d0	push-t	cup	Avg
CFM (Tong et al. [19])	4	0.730	0.927	0.720	**0.753**	0.877	0.776	0.797
OT-CFM (Tong et al. [19])	4	0.723	**0.940**	0.623	0.697	0.712	0.743	0.740
DP (DDIM) (Chi et al. [5])	4	0.720	0.917	0.650	0.650	0.877	0.763	0.763
DP (DDIM) (Chi et al. [5])	20	0.720	**0.940**	0.650	0.707	**0.881**	0.787	0.781
Adaflow (Hu et al. [17])	1.29	0.707	0.937	0.717	0.667	0.849	0.823	0.783
CFM (Tong et al. [19])	2	0.737	0.913	0.730	0.690	0.870	0.797	0.790
OT-CFM (Tong et al. [19])	2	0.667	0.897	0.613	0.630	0.694	0.753	0.709
COT Policy (Ours)	2	**0.750**	0.937	**0.787**	0.707	0.878	**0.847**	**0.818**

Table 1: Performance comparison of COT Policy with all baselines on the robot manipulation tasks.

Baselines. COT Policy is compared against **1)** CFM, **2)** OT-CFM [19], **3)** Diffusion Policy (DP) [5], and **4)** Adaflow [17], a variance-based adaptive CFM sampler. These choices allow us to support our claim that COT Policy outperforms ODE-based methods in few-step generation without compromising action diversity and with comparable training cost. Although Adaflow requires two-phase training, the second phase trains only linear layers and therefore adds negligible training overhead. Diffusion Policy with a large number of function evaluations per step (NFE) is used for comparisons as a state-of-the-art policy. For all results, unless stated otherwise, COT Policy is implemented with 64 clusters for discretization of the conditions (§2.3). See Appendix B.1 for all implementation details.

Tasks. To validate the proposed COT approach, we first consider two low-dimensional synthetic tasks with discrete and continuous conditions (Fig. 1). Because the focus of this paper is imitation learning for robot policies, this study is relegated to the Appendix (Appendix D.1).

In the main text, we consider three sets of simulation tasks: **1)** four robot manipulation tasks (Fig. 2) from *MimicGen* [32], which is a RoboSuite [33] and RoboMimic [6] extension, with realistic manipulation environments and the ability to easily generate demonstrations (§3.1), **2)** two toy decision making tasks, namely, `push-t` (commonly used in Behavior Cloning [34, 5]) and `cup` from *dm-control* [35] (§3.1), and **3)** two Maze navigation tasks from D4RL [36] (§3.2).

Furthermore, we evaluate our policy on real hardware in three manipulation tasks (§3.4).

3.1 Simulated Robot Manipulation Tasks

We assess the performance of COT Policy in robot manipulation tasks. We choose four tasks from the MimicGen benchmark along with `push-t` and `cup`. We evaluate all policies on 150 rollouts and 2 different seeds for the noise distribution, except for *Push-T*, which is evaluated on 600 rollouts in total. For each task we report the average success rate over all rollouts and seeds, except `push-t`, for which we report the average percentage of coverage of the goal from the T block, consistent with evaluation metrics from prior work. For Adaflow we use the trained CFM policy and fine-tune the variance predictor as explained in [17]. For CFM, OT-CFM, and COT Policy we use the `midpoint` solver to sample action trajectories, for Diffusion Policy we use DDIM [37], and for Adaflow we use the variance adaptive solver with 2 maximum steps to have a fair comparison with COT Policy.

2-step COT Policy outperforms all baselines. From Table 1 it can be seen that COT Policy outperforms all compared methods on average. COT Policy achieves the highest success rate across tasks with NFE = 2 and even exceeds CFM, OT-CFM, and DP when those models are evaluated with NFE = 4 or DP with NFE = 20. OT-CFM again lags behind all the other methods since it does not

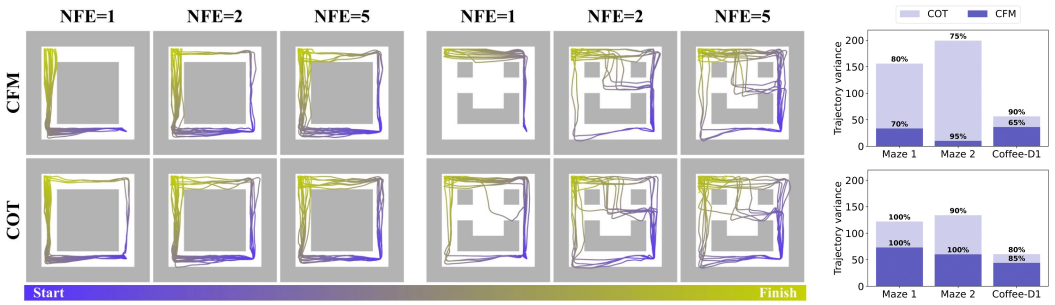

Figure 3: **Left:** Evaluation of the ability of CFM and COT Policy to encode multiple modes in the maze navigation task for 1, 2, and 5 `euler` steps (NFE). **Right:** Trajectory Variance of CFM and COT Policy in the two Maze tasks and `coffee_d1` task for NFE = 1 on the top and NFE = 2 on the bottom. The percentage of successful rollouts is written above each bar.

take into account the observations when coupling action trajectories with noisy trajectories, yielding a biased optimal flow. We hypothesize that Adaflow performs worse than 2-step CFM, despite having an adaptive step size, because it takes a maximum of 2 `euler` steps instead of the 2 steps that are needed for one iteration of the `midpoint` method in the other flow-based policies.

3.2 Evaluating Multimodality

We further evaluate the diversity of actions generated by COT Policy and by a CFM policy. To measure trajectory diversity, we introduce the *Trajectory Variance (TV)* metric. For a set of n trajectories $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, we calculate TV as the mean squared Dynamic Time Warping [DTW; 38] distance between each trajectory in $\mathcal{A}$ and the barycenter (or Fréchet mean) of $\mathcal{A}$, calculated using DWT Barycentric Average [DBA; 39], as implemented in the *tslearn* package [40]. The barycenter μ_a acts as a 'mean trajectory' and is defined as $\mu_a = \arg\min_\mu \sum_{i=1}^n d^2_{\text{DWT}}(a_i, \mu)$, where d_{DWT} is the DWT distance between two trajectories. Using μ_a, we calculate TV as $TV = \frac{1}{n}\sum_{i=1}^n d^2_{\text{DWT}}(a_i, \mu_a)$. We view TV as an analogue of variance in the space of trajectories and use it to measure the diversity of action plans generated by the two policies. (In the case of one-step trajectories with the same initial point, TV exactly recovers the trace of the empirical covariance matrix of the velocity vectors.)

Out of 50 rollouts with the same environment seed, only successful ones are considered for the calculation of TV in the results presented in Fig. 3.

COT Policy generates diverse actions even for low NFE. Based on the results shown in Fig. 3, COT Policy exhibits more diversity (higher TV) for all tasks, which is in accordance with the trajectories shown on the left side of the same figure. The TV and success rate of CFM increases when we use 2 `euler` steps instead of 1, while for COT Policy we observe an average increase in success rate but slightly lower TV. This is potentially due to discarding unsuccessful rollouts in the calculation of TV. Discarding certain trajectories can lead to either an increase or decrease of TV depending on which mode they represent. See Appendix D.2 for additional results.

3.3 Ablations

Effect of cluster number in COT Policy. The number of clusters K used in COT Policy directly impacts the performance. From Fig. 4 it is evident that COT with too few clusters exhibits similar degradation to OT-CFM. For a number of clusters close to the batch size we observe the greatest advantage of COT over CFM. This is potentially due to the diversity of clusters in each batch this choice induces, which should not be too low (exhibiting close to OT-CFM behavior) or too close to the batch size (exhibiting close to CFM behavior as each condition

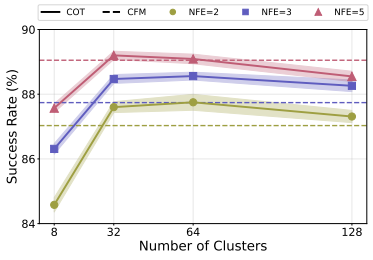

Figure 4: Success rates of COT Policy on the `push-t` task for $K = 8, 32, 64, 128$ and NFE = $2, 4, 10$.

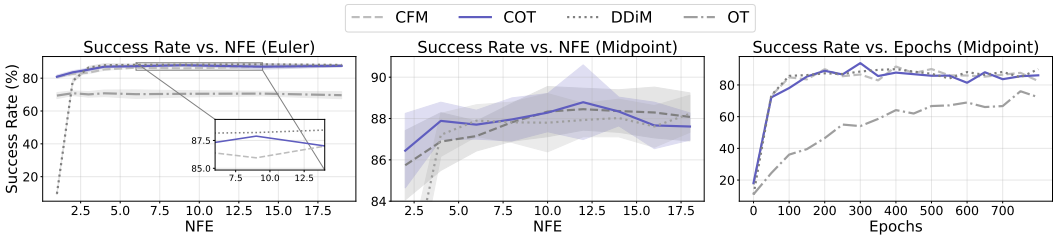

Figure 5: Success rates on the `push-t` task with varying NFE with the **Left:** `euler` solver and **Middle:** `midpoint` solver. **Right:** success rate over training epochs.

is unique). As the number of clusters increases and approaches the dataset size, the behavior of COT-CFM asymptotically resembles that of CFM. Further discussion can be found in Appendix E.2. Additionally, we show the importance of observation discretization by providing a comparison of COT Policy with and without clustering in Appendix D.3.

Effect of NFE and solver type. From Fig. 5 and the rest of the results presented in this section, it is clear that NFE has a direct impact on the quality and diversity of the generated samples from any method. The number of solver steps, despite heavily impacting all methods, has a more subtle effect on COT Policy as sample diversity and success rates only slightly benefit from increases in NFE. Furthermore, the choice of the fixed-step solver used plays a role according to Fig. 5 as there is a consistent increase in performance with the `midpoint` solver for the same NFE.

Training complexity. In Fig. 5, we see that COT Policy training converges at approximately the same time as CFM and DP. Moreover, our choices for the encoder $\mathcal{E}$ and the discretizer Q (PCA and K-means respectively) as well as minibatch OT add negligible computational overhead per batch when implemented in GPU-accelerated frameworks, compared to performing a backward pass on the flow model. Therefore, COT Policy requires approximately the same time as CFM and DP to train.

3.4 Real Robot Experiments

We evaluated the low-NFE performance of COT Policy and CFM Policy on three real-world robot tasks, shown in Fig. 6, using a KUKA IIWA 14 robot and two Realsense D415 cameras—one mounted on the end-effector and one providing external view of the workspace. We collected expert demonstrations using teleoperation and executed the policy in a PC with a NVIDIA GeForce RTX 2080.



Figure 6: Visualization of real tasks.

Task	CFM policy		COT policy	
	SR ↑	*TTC* ↓	*SR* ↑	*TTC* ↓
`push-T`	0.2 / 0.4	54.4 / 44.7	**0.8** / 0.6	**35.7** / 35.8
`cup-stacking`	0.4 / 0.2	44.5 / 51.1	0.6 / **0.8**	32.2 / **24.0**
`cup-in-drawer`	0.2 / 0.2	53.3 / 54.0	**0.8** / 0.2	**35.3** / 52.9

Table 2: Success Rate (SR; 5 rollouts) and Time to Completion (TTC; seconds) in the real tasks. Results in the format (`euler`–NFE=1 / `midpoint`–NFE=2).

In all three tasks, CFM underperforms compared to COT Policy, while also requiring more time to solve the task (Table 2). The increased NFE required by CFM to accurately generate actions from the multimodal distribution of expert human demonstrations leads to intermittent motions. In contrast, COT policy is able to solve all tasks with at most 2 inference steps.

4 Conclusions

We present COT Policy, a flow matching policy for fast inference in robot tasks and multimodal action generation. We achieve this by pairing noise and action trajectories during training with an approximation to the Conditional Optimal Transport plan. We have shown that 2-step COT Policy outperforms other generative models in sequential decision making tasks while maintaining action diversity. This allows COT Policy to be used for high-quality real-time action generation while requiring only a handful of network evaluations to perform inference.

5 Limitations

One major limitation of our method is the introduction of two hyperparameters. Although the scale factor γ does not significantly impact performance, the choice of cluster K does. Different dataset and batch sizes can impact the optimal choice of K. For robot manipulation tasks we suggest setting K equal to the batch size however it is not evident if that choice is sensible for any task. Additionally, although we are able to perform high quality 2-step action generation with the `midpoint` solver, in some cases there is a gap between low and high NFE performance, especially as dimensionality grows. It would be an interesting direction to address this gap by combining OT coupling with distillation techniques that require a single training phase [14, 13].

Acknowledgments

This work is supported by the Honda Research Institute Europe and the JST Moonshot R&D (Grant No. JPMJMS2239). We appreciate all reviewers for the valuable feedback.

References

[1] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

[2] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[3] J. Urain, A. Mandlekar, Y. Du, M. Shafiullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024.

[4] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations (ICLR)*, 2023.

[5] 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 (IJRR)*, 2023.

[6] 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 (CoRL)*, 2021.

[7] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[8] A. Sochopoulos, M. Gienger, and S. Vijayakumar. Learning deep dynamical systems using stable neural odes. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[9] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems (NeurIPS)*, 2022.

[10] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[11] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems (RSS)*, 2024.

[12] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

[13] L. Yang, Z. Zhang, Z. Zhang, X. Liu, M. Xu, W. Zhang, C. Meng, S. Ermon, and B. Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[14] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

[15] 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 (RSS)*, 2024.

[16] Z. Wang, Z. Li, A. Mandlekar, Z. Xu, J. Fan, Y. Narang, L. Fan, Y. Zhu, Y. Balaji, M. Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[17] X. Hu, qiang liu, X. Liu, and B. Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. In *Advances in neural information processing systems (NeurIPS)*, 2024.

[18] X. Liu, C. Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

[19] A. Tong, K. FATRAS, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

[20] G. Kerrigan, G. Migliorini, and P. Smyth. Dynamic conditional optimal transport through simulation-free flows. In *Advances in neural information processing systems (NeurIPS)*, 2024.

[21] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems (NeurIPS)*, 2018.

[22] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning (ICML)*, 2023.

[23] Y. Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[24] J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[25] B. Hosseini, A. W. Hsu, and A. Taghvaei. Conditional optimal transport on function spaces. *arXiv preprint arXiv:2311.05672*, 2023.

[26] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *Conference on Robot Learning (CoRL)*, 2024.

[27] Y. Hou, Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, S. Feng, B. Burchfiel, and S. Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. *arXiv preprint arXiv:2410.09309*, 2024.

[28] M. Aburub, C. C. Beltran-Hernandez, T. Kamijo, and M. Hamaya. Learning diffusion policies from demonstrations for compliant contact-rich manipulation. *arXiv preprint arXiv:2410.19235*, 2024.

[29] K. Tian, Y. Jiang, Z. Yuan, B. PENG, and L. Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in neural information processing systems (NeurIPS)*, 2024.

[30] R. Zheng, C.-A. Cheng, H. Daumé III, F. Huang, and A. Kolobov. Prise: Llm-style sequence compression for learning temporal action abstractions in control. *arXiv preprint arXiv:2402.10450*, 2024.

[31] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, and Y. Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

[32] 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 *Conference on Robot Learning (CoRL)*, 2023.

[33] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[34] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning (CoRL)*, 2022.

[35] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[36] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[37] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

[38] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[39] F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3):678–693, 2011.

[40] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, et al. Tslearn, a machine learning toolkit for time series data. *Journal of machine learning research*, 21(118):1–6, 2020.

[41] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

[42] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems (NeurIPS)*, 2022.

[43] Y. Jin, Z. Sun, N. Li, K. Xu, K. Xu, H. Jiang, N. Zhuang, Q. Huang, Y. Song, Y. MU, and Z. Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

[44] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[45] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[46] S. Saxena, C. Herrmann, J. Hur, A. Kar, M. Norouzi, D. Sun, and D. J. Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[47] M. Gui, J. S. Fischer, U. Prestel, P. Ma, D. Kotovenko, O. Grebenkova, S. A. Baumann, V. T. Hu, and B. Ommer. Depthfm: Fast monocular depth estimation with flow matching. *arXiv preprint arXiv:2403.13788*, 2024.

[48] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[49] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

[50] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[51] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2023.

[52] F. Zhang and M. Gienger. Affordance-based robot manipulation with flow matching. *arXiv preprint arXiv:2409.01083*, 2024.

[53] H. Ding, N. Jaquier, J. Peters, and L. Rozo. Fast and robust visuomotor riemannian flow matching policy. *arXiv preprint arXiv:2412.10855*, 2024.

[54] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters. Actionflow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching. *arXiv preprint arXiv:2409.04576*, 2024.

[55] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[56] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

[57] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

[58] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[59] A. Tong, N. Malkin, K. Fatras, L. Atanackovic, Y. Zhang, G. Huguet, G. Wolf, and Y. Bengio. Simulation-free schrödinger bridges via score and flow matching. *arXiv preprint 2307.03672*, 2023.

[60] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 2021.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[62] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2023.

[63] N. Tsagkas, O. Mac Aodha, and C. X. Lu. Vl-fields: Towards language-grounded neural implicit spatial representations. In *International Conference on Robotics and Automation Workshops (ICRA)*, 2023.

[64] N. Tsagkas, J. Rome, S. Ramamoorthy, O. Mac Aodha, and C. X. Lu. Click to grasp: Zero-shot precise manipulation via visual diffusion descriptors. In *International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[65] G. Li, N. Tsagkas, J. Song, R. Mon-Williams, S. Vijayakumar, K. Shao, and L. Sevilla-Lara. Learning precise affordances from egocentric videos for robotic manipulation. *arxiv preprint arXiv:2408.10123*, 2024.

[66] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning (CoRL)*, 2023.

[67] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. In *Robotics: Science and Systems (RSS)*, 2023.

[68] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *Conference on Robot Learning (CoRL)*, 2024.

[69] N. Tsagkas, A. Sochopoulos, D. Danier, C. X. Lu, and O. M. Aodha. When pre-trained visual representations fall short: Limitations in visuo-motor robot learning. *arxiv preprint arXiv:2502.03270*, 2025.

[70] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

A Related Work

Diffusion and flow-based models in robotics. Diffusion and flow-based models have demonstrated remarkable performance across various domains, ranging from image and video generation [41, 42, 43, 44] to monocular depth estimation [45, 46, 47]. Their ability to model complex, high-dimensional probability distributions has naturally led to their widespread use in robotics, particularly in policy learning [3, 4]. Chi et al. [5] proposed DMs as an alternative to standard explicit policy models [6], learning distributions of short-horizon trajectories conditioned on the robot's proprioceptive and exteroceptive observations. Other methods [48, 49] have leveraged DMs to develop accurate planners that integrate rewards into trajectory generation through conditional generation techniques, such as diffusion guidance [50].

Recent work has explored the effectiveness of generative model (GM)-based policies conditioned on robot-relevant observations, including point clouds [26], text [51, 52], and force-torque data [27, 28]. In many cases, the geometric structure of the observations has been explicitly considered by incorporating manifold variants of Flow Matching [53, 26] or employing equivariant Transformer architectures for policy learning [54].

Despite their success in various robotic tasks, DMs and flow-based models suffer from slow inference speeds, posing a significant limitation for real-time deployment or scenarios with constrained computational resources. Some approaches leverage FM to reduce the number of steps required for action sampling. However, even in flow-based policies, few-step generation can significantly impact the diversity of action trajectories and, potentially, the policy's success rate.

Accelerated inference in continuous-time generative models. Efforts to accelerate inference in DMs began with the development of alternative samplers [37, 55, 9], which replace computationally expensive iterative denoising with more efficient, often deterministic, sampling schemes. While these methods significantly improve inference speed, they still require multiple steps to generate high-quality samples. To achieve competitive one-step inference quality, distillation methods [56, 13, 12, 57, 14, 58, 18] are commonly employed.

This family of approaches primarily seeks models capable of directly predicting the solution to the denoising SDE at any time step, enforcing consistency objectives [12, 57, 14] rather than explicitly distilling few-step models using generated denoising SDE solutions. A major drawback of distillation methods is their reliance on costly training procedures, often requiring multiple training phases involving teacher DMs and student one-step models. Some high-quality one-step models [56, 18] necessitate progressive training phases. While certain methods can be trained in a single phase [12, 14], they require more training steps to converge, each step being computationally expensive, and the performance gap between few-step and one-step inference remains large. Most robot policies based on distillation methods [11, 16, 31] face similar limitations.

In this work, we build upon Conditional Optimal Transport principles to develop policies that train as efficiently as CFM while achieving superior few-step performance. *Our method serves as an efficient alternative to CFM by implicitly improving few-step performance via learning straighter paths. This is in contrast to distillation methods which explicitly learn few-step solutions of the denoising ODE.*

B Implementation Details: Simulation

B.1 Training details

Training on manipulation and Maze tasks. All the baselines have been trained using the same CNN-based U-Net architecture from [5]. The flow network for all flow-based techniques and the noise network of DP utilize the U-Net as their architecture, with the same hyperparameters, totaling approximately 240 million parameters (excluding the vision encoders). The image observations first pass through a vision encoder. For more details on the vision encoder see Appendix B.3. We detail further training decisions below:

- **CFM**. For the implementation of the CFM-based policy we followed the methodology of Tong et al. [19]. The only discrepancy is the addition of the observations as a conditioning variable in the flow model.
- **OT-CFM**. OT-CFM uses the same setup as CFM with the addition of coupling noise and action samples using OT. For that we use the tools provided by *torchcfm* [19, 59] for calculating the minibatch OT plan with conditions. Since the noise and action samples are rearranged based on the OT plan, the conditions are rearranged as well with the respective actions. To calculate the OT plan we use the Earth Mover's Distance (EMD) algorithm [60].
- **Adaflow**. We use the trained CFM models as the base flow models of Adaflow for each task. We then train the variance estimation network, implemented as a 2 layer MLP, according to the training procedure detailed in [17], for an additional 200 epochs. The flow network was frozen during training of the variance estimation network. Since we were unable to find a recommended number of training epochs for the variance estimation network we opted for a large enough epoch count despite updating only the weights of some linear layers. The learning rate was held constant throughout training. We set $\eta = 0.5$ and $\epsilon_{min} = 2$. The choice of $\epsilon_{min} = 2$ was made so that Adaflow uses a maximum of 2 steps during inference, to ensure a fair comparison with other 2-step methods. This value of η was chosen since slightly higher values led to Adaflow using NFE=2 and slightly lower NFE=1 at every inference step.
- **DP**. The implementation of DP is identical to the one in Chi et al. [5].

Hyperparameter	Moons & Fork		Maze		MimicGen, Push-T & Ball-in-cup		
	CFM & OT-CFM	COT Policy	CFM & OT-CFM	COT Policy	DP	CFM & OT-CFM	COT Policy
Learning rate	1e-3	1e-3	1e-4	1e-4	1e-4	1e-4	1e-4
Optimizer	Adam	Adam	AdamW	AdamW	AdamW	AdamW	AdamW
Batch size	256	256	64	64	64	64	64
Weight decay	0.0	0.0	1e-6	1e-6	1e-6	1e-6	1e-6
Epochs	–	–	1000	1000	1000	1000	1000
Training Steps	50000	50000	–	–	–	–	–
Learning rate schedule	constant	constant	cosine	cosine	cosine	cosine	cosine
EMA decay rate	–	–	0.9999	0.9999	0.9999	0.9999	0.9999
Warmup steps	–	–	1500	1500	1500	1500	1500
Inference steps in training rollouts	-	-	2 (Euler)	2 (Euler)	100 (DDIM)	2 (Midpoint)	2 (Midpoint)
Action prediction horizon	-	-	60	60	16	16	16
Observation horizon	-	-	2	2	2	2	2
Action execution horizon	-	-	50	50	8	8	8
Image size	2(Moons)/1(Fork)	2(Moons)/1(Fork)	96x96	96x96	96x96	96x96	96x96
Number of Clusters	–	2	–	16	–	–	64
PCA features	–	–	–	100	–	–	100

Table 3: Training hyperparameters for COT Policy and baselines

Training on distribution tasks. We train CFM, OT-CFM, and COT-CFM using an MLP with 4 Linear layers with 64 neurons each and SELU activations. The conditioning and time variables are concatenated with the samples and passed as inputs. For these two tasks we do not use positional embeddings for the time variable. For the Fork task we do not encode the conditions as they are already 1-dimensional, however we cluster them using K-Means with $K = 2$. We found that this number of clusters works best for this task, potentially due to the fact that the distribution changes based on two different sets of values for the condition variable, namely $x \leq 0$ and $x > 0$.

B.2 Clustering and dimensionality reduction in COT Policy

As mentioned in the main text, for COT Policy, we implement the encoder $\mathcal{E}$ as a PCA transformation of the images and the discretizer Q as K-means clustering, with the centroids of the clusters being the discretized values of the conditions. Both algorithms were implemented in *pytorch* to allow GPU acceleration. The eigenvector matrix for PCA is calculated using SVD on the images from the entire dataset before training and during training dimensionality reduction is performed by multiplying the data matrix with the eigenvector matrix. Similarly the clusters are computed using the entire dataset before training and during training each pair of PCA image features and proprioception is matched to its closest cluster, based on Euclidean distance. Although the speed of these operations is negligible compared to forward and backward passes of the U-Net, the storage of the eigenvector matrix for PCA can have a significant impact on the GPU memory required during training.

B.3 Visual Encoder

We train the visual encoder used for generating the embeddings that condition the flow, end-to-end alongside the flow network, using a randomly initialized ResNet-18 [61]. This choice aligns with [5], which reported that conditioning a DP with features extracted from pre-trained visual representations (PVRs) (*e.g.*, R3M [62]) led to suboptimal performance. However, recent advances in 3D spatial representations, fused with PVR-derived features while ensuring multi-view consistency, have demonstrated promising results in downstream robotic tasks [63, 64, 65, 66, 67]. This has sparked interest in leveraging such methods for conditioning diffusion policies [68, 51]. We argue that this research direction is still in its early stages, with open challenges in effectively integrating PVR features into imitation learning policies [69]. In future work, we aim to investigate how PVRs can enhance policy generalization, particularly in out-of-distribution scenarios.

B.4 Demonstration collection

MimicGen. For all MimicGen tasks we use the datasets provided by the authors of [32]. These datasets consist of 1000 demonstrations, synthetically generated from a total of 10 human demonstrations. We only keep the first 100 demonstrations to keep training times within reasonable limits. Synthetically generated datasets can have an immediate impact on the NFE needed to get the optimal performance, as demonstrations collected using deterministic policies lack multimodality. Lack of multimodality implies conditional distributions that lack variance and therefore few-step CFM and COT-CFM performance should be similar. However, we found that for the MimicGen tasks considered in this paper there was enough variation in the demonstrations. Ideally, human-collected demonstrations would have the maximum variation, however we avoided using the tasks and demonstrations used in Chi et al. [5], as most generative policies solve the majority of them with 100% success rate.

PushT and BallInCup. For the `push-t` task we used 90 demonstrations provided in Chi et al. [5]. For the `cup` task, we trained a PPO [70] agent and collected 100 demonstrations by executing the learned policy.

Maze2D. We used the policy that comes with D4RL [36] for collecting demonstrations in the Maze environments, implemented as a q-iteration policy combined with a waypoint controller. We modified the policy by adding noise in the q values in order to generate diverse demonstrations when the starting and goal positions are fixed. A total of 100 demonstrations were collected for each maze environment.

B.5 Hardware

Training and evaluation for the simulated tasks were performed using a workstation with (CPU, RAM, GPU): AMD Ryzen Threadripper PRO 5965WX 24-Cores, 128GB, 2× NVIDIA GeForce RTX 4090.

C Implementation Details: Real-robot tasks

C.1 Training details

For all real-robot task and for both CFM and COT policies we use the U-Net architecture for the flow as described in Appendix B.1. The visual encoders were also trained end-to-end and followed the same architecture described in Appendix B.3, with the only difference being the resolution of the images used. We decided to use higher resolution images for `push-T` (*e.g.*, 240x320 opposed to 96x96 (or 96x120) used in simulation and real-robot manipulation tasks) in order to accurately capture the pose of the T block and the target. For more details on the hyperparameters used for training on the real-robot tasks, see Table 4.

Hyperparameter	`push-T`, `cup-stacking`, `cup-in-drawer`	
	CFM Policy	COT Policy
Learning rate	1e-4	1e-4
Optimizer	AdamW	AdamW
Batch size	64	64
Weight decay	1e-6	1e-6
Epochs	1000	1000
Learning rate schedule	cosine	cosine
EMA decay rate	0.9999	0.9999
Warmup steps	1500	1500
Action prediction horizon	16	16
Observation horizon	2	2
Action execution horizon	8	8
Image size	240x320 (`push-T`) / 96x120 (others)	240x320 (`push-T`) / 96x120 (others)
Number of Clusters	–	64
PCA features	–	100

Table 4: Training hyperparameters for CFM Policy and COT Policy across real-robot tasks.

C.2 Demonstration collection

For each of the three tasks CFM and COT policies were tested on using real hardware, we collected a total of 30 demonstrations using a 3D spacemouse. The actions used for learning the two policies were end-effector twists that were commanded through the teleoperation device. For the `push-T` task we only allowed translational motion of the end-effector, making the actions 2-dimensional. For the rest of the tasks the actions were 7-dimensional (6 dimensions for controlling the twist of the end-effector and 1 for controlling the gripper).

Despite having less demonstrations for the real tasks than for the simulation tasks, the real datasets ended up highly multimodal and covered multiple initial configurations of the workspace and the robot.

C.3 Evaluation details

Test protocol We evaluated the performance of both CFM and COT policies on 5 random environment configurations for each task. The environment configurations were chosen a-priori and were recreated as accurately as possible after every rollout. We further made sure that none of these environment configurations appeared identical in our datasets.

For each task we tested both policies using the minimum amount of steps for each of the `euler` and `midpoint` solvers (*e.g.*, 1 and 2 respectively). The maximum allowed time before considering a rollout a failure was chosen to be 1 minute. For each of the unsuccessful rollouts reported in Table 2. their contribution to the average TTC was 60 seconds.

Discussion Throughout our experiments we have consistently found that COT policies outperformed CFM policies for low-NFE inference. Moreover, we empirically observed that CFM policies would often get stuck for extended periods of time or produce jerky motions. This was less frequent when the `midpoint` solver was used, however success rates were still low. On the contrary, COT Policy succeeded for most of the initial configurations in all tasks. We have observed that COT policy was able to reproduce most of the modes existing in the dataset, even with 1-step `euler` inference, as can be seen in Fig. 7, where the robot pushes the T block in the target with both clockwise and anti-clockwise motions.

Although performance is similar when using the `euler` and the `midpoint` solver, there is a big gap in the `cup-in-drawer` task. We suspect that this gap mainly stems from the intermittent motions induced by the 2-step `midpoint` on a task that is dynamic and has a high risk of collision. Low performance for CFM policy is consistent with the other tasks and stems from the inability of CFM to

generate high quality samples with few-step inference and the reasons described above when it comes to inference with the `midpoint` solver. Intuitively, intermittent motion for few-step inference could be mitigated by having a faster GPU on the robot's control computer, however this is not always possible.

C.4 Hardware

Training for the real tasks was performed using a workstation with (CPU, RAM, GPU): AMD Ryzen Threadripper PRO 5965WX 24-Cores, 128GB, 2× NVIDIA GeForce RTX 4090. Inference on the robot was performed using a desktop PC with (CPU, RAM, GPU): Intel(R) i9-9900KF, 64GB, NVIDIA GeForce RTX 2080.

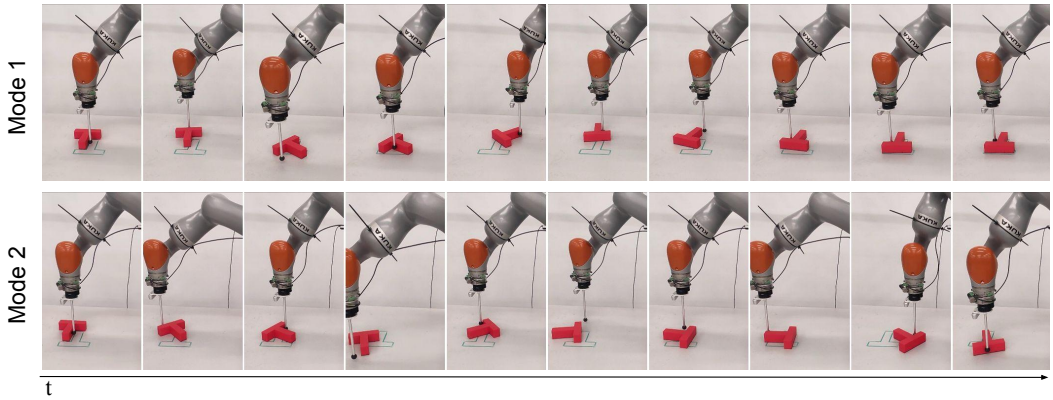

Figure 7: Two modes of solving the real `push-T` task (*e.g.*, by rotating the T clockwise and anti-clockwise) uncovered by COT Policy with NFE=1 using the `euler` solver.

D Additional evaluation results

D.1 Low-Dimensional Distribution Learning

We evaluate how well CFM, OT-CFM, and COT-CFM can recreate the 2D two-moon distribution shown in Fig. 1 starting from a ring of 8 Gaussian distributions and the 1D fork distribution adapted from [17], with a standard Gaussian noise distribution (see Fig. 8). Both distributions are conditional. In the moons distribution, 0 and 1 are condition values indicating the top or bottom moon, respectively. The fork distribution is a continuous distribution over y conditioned on x, defined as $y = 0$ if $x \leq 0$ and $y = \mathcal{U}(\{x, -x\})$ if $x > 0$. Since the condition variable is already low-dimensional, we cluster it into 2 clusters.

Task		CFM		OT-CFM		COT-CFM	
		$W_2^2 \downarrow$	NFE	$W_2^2 \downarrow$	NFE	$W_2^2 \downarrow$	NFE
Moons 2D	E	1.490	1	0.777	1	**0.345**	1
		1.094	2	0.709	2	**0.322**	2
	D	0.335	134	0.683	74	**0.253**	**62**
Fork 1D	E	1.134	1	2.678	1	**0.349**	1
		0.575	2	0.992	2	**0.125**	2
	D	0.157	74	0.981	**62**	**0.072**	74

Table 5: 2-Wasserstein distance on the moons and fork distributions evaluated using `euler` (E) and `dormand-prince` (D) solvers.

COT-CFM improves sample quality with fewer steps. We report the 2-Wasserstein distance between the recreated and target distribution in Table 5. For the same number of few `euler` steps, COT-CFM outperforms the other two methods. In both tasks, 2-step generation from COT-CFM with an `euler` solver outperforms > 60-step CFM and OT-CFM generation with the Dormand-Prince 5th-order (`dopri5`) solver, an adaptive solver commonly used for solving Neural ODEs. Furthermore, COT-CFM is the only method capable of successfully recreating the two moons for NFE = 1 (see Fig. 1). This is mainly due to the straight paths the COT couplings induce, which is also evident by the reduced number of steps taken by the `dopri5` solver. Interestingly, CFM evaluated with a 1-step

`euler` solver approximately generates the conditional mean of the two conditional distributions, while OT-CFM fails to model them accurately even with 100 steps. These results confirm the claims made in §2.2 that (I-)CFM gives unbiased conditional samples with curved integration paths, OT-CFM gives straight paths but biased samples, while the proposed COT-CFM achieves a balance between the two. See Fig. 8 for further visualizations.

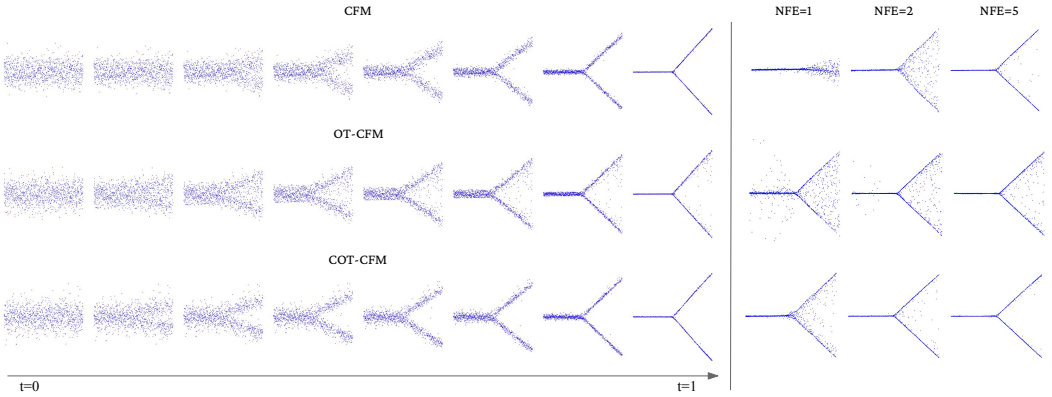

Figure 8: Qualitative performance of CFM, OT-CFM, and COT-CFM on generating the conditional Fork distribution from standard Gaussian noise. **Left:** The distribution as time progresses from 0 to 1 and **Right:** the generated distributions for different NFE using the `euler` solver.

D.2 Additional Trajectory Variance results

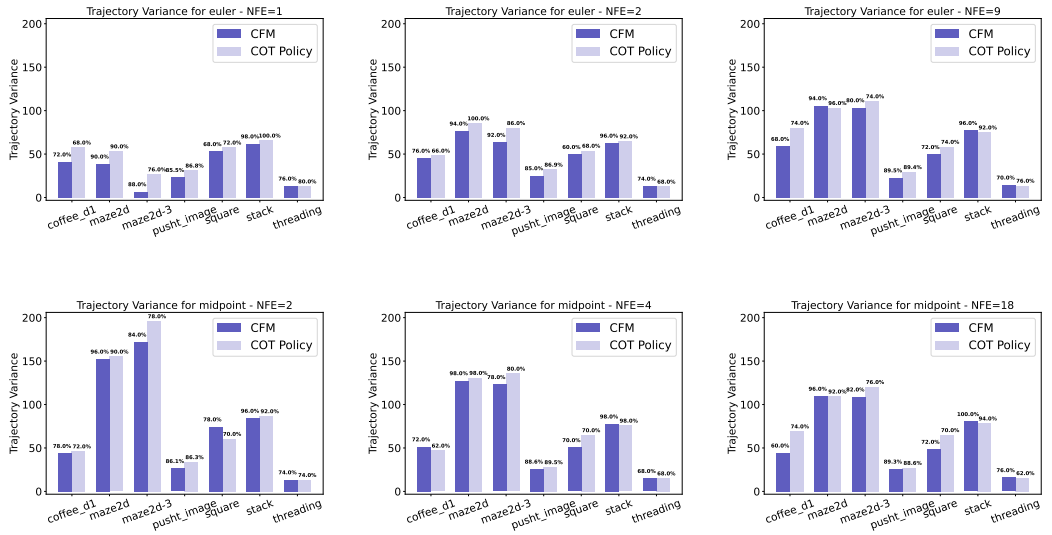

Figure 9: Trajectory variance comparison across six tasks using CFM and COT Policy **Top:** using the `euler` solver and **Bottom:** using the `midpoint` solver. Trajectory variance and success rates were computed over an average of 50 environemnt rollouts with random environment seeds (in contrast to Fig. 3 where enviornment seed was the same for all rollouts).

D.3 COT Policies without clustering

As we explain in Appendix E.2, clustering alleviates the need for tuning γ to a value that balances condition and action coupling. To demonstrate the need for fine-tuning γ we report the performance

Method	NFE	push-t	coffee_d1	Avg
COT Policy (w/o clustering)	2	0.858	0.690	0.774
COT Policy (with clustering)	2	**0.878**	**0.787**	**0.833**

Table 6: Performance comparison on push-t and coffee_d1 tasks.

of a variant of COT policy which uses Equation (6) for γ and no clustering for the push-t and coffee_d1 tasks. As per Table 6, clustering leads to increased success rates. Similar performance could potentially be achieved for COT without clustering, however this would require expensive tuning of γ for every dataset. In contrast, clustering allows us to keep the number of clusters and γ fixed throughout all tasks and still observe higher success rates and multimodal behaviors for low NFE.

E Hyperparameters of COT Policy

E.1 The effect of γ on the minibatch OT solution

The hyperparameter γ, used in (4), regulates the effect the conditions c_0, c_1 have in the coupling process using unconditional minibatch OT. In order to retrieve an empirical coupling $\pi(x_0, x_1)$ that approximates the conditional OT coupling between x_0 and x_1, the second term of the cost (4) needs to dominate (large γ) the first to prioritize pairing by condition. If the two terms have similar contributions to the cost (small γ), then neighboring noise x_0 and target samples x_1 might be coupled despite having different conditions. As already mentioned in §2.2, when $\gamma = 0$ we recover OT-CFM as noise and target samples are coupled solely based on their Euclidean distance.

To get the desired empirical coupling that prioritizes coupling by condition, we therefore need to set γ to a large value. However, it is possible that large values can lead to numerical instability. As can be seen by Fig. 10, the OT matrix that we get from performing minibatch OT on the tuples (x_0, c_0) and (x_1, c_1) is consistent for $\gamma = \{10, 100, 1000, 10000\}$. For larger values ($\gamma > 10000$) we observe that the diagonal dominates the top part of the OT matrix. This practically means that the i-th target point gets coupled with the $(|B_1| - i)$-th source point. This coupling is similar to the independent coupling (i-th target point get coupled with the i-th source point) and is the result of numerical errors encountered while solving the optimization problem of the Earth-Movers Distance (EMD) problem.

To avoid behavior similar to (I)-CFM, while still prioritizing condition coupling, we set γ for each batch based on the following formula:

$$\gamma = 10 \cdot \frac{\overbrace{\sum_{i=0}^{|B_1|} \|x_0^{(i)} - x_1^{(i)}\|^2}^{\text{Avg. sample distance}}}{\underbrace{\sum_{j=0}^{|B_1|} \|c_0^{(j)} - c_1^{(j)}\|^2}_{\text{Avg. condition distance}}} \tag{6}$$

With this choice of γ the second term of (4) is scaled to be on average 10 times larger than the first. We have empirically found that choosing γ based on (6), numerical errors in the minibatch OT calculations are avoided and in condition coupling is prioritized.

E.2 The effect of the number of clusters on COT Policy

Clustering the observations in COT Policy has a similar effect to setting γ to a value that balances sample and condition distance. This is because samples $x_0^{(i)}, x_1^{(i)}$ from the same clusters ($c_0^{(i)} = c_1^{(i)}$)

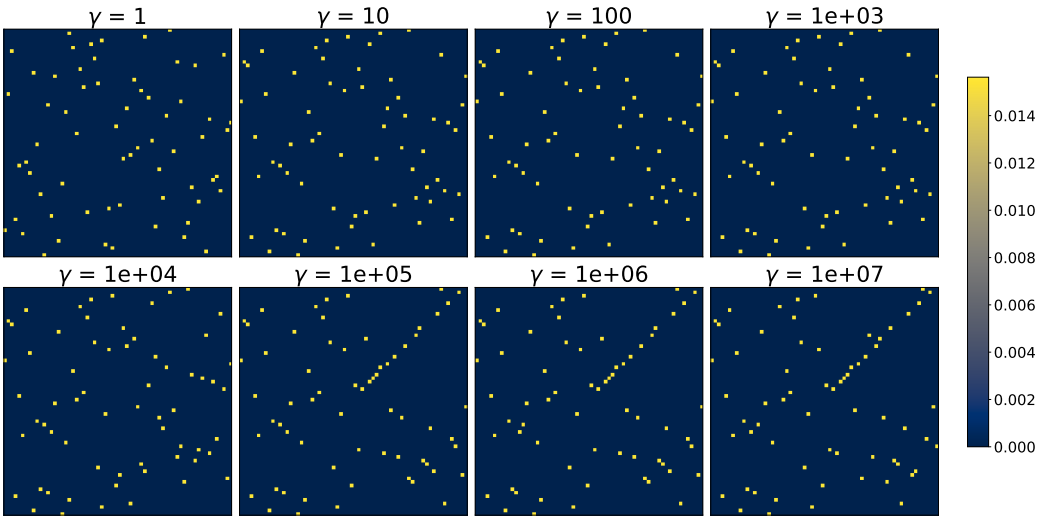

Figure 10: OT matrix for a range of condition scaling parameters γ. The OT matrix is consistent for medium to large values of γ ($[10, 10000]$), but numerical errors are observed for larger values of γ.

are coupled based on their Euclidean distance (according to Equation (4)), despite the fact that their corresponding actual observations might be different ($o_0^{(i)} \neq o_1^{(i)}$). Finding a suitable γ for every empirical distribution is not straightforward, but choosing a number of clusters is simpler based on the following observation:

Proposition. *In tasks with continuous conditions, like visuomotor control, given a large enough γ, COT Policy approximates OT-CFM as $K \to 0$ and (I)-CFM as $K \to |\mathcal{D}|$.*

Although we don't provide full proofs for the above proposition, we give some intuitions about the connection between CFM, OT-CFM, and COT-CFM in the extreme cases where $K = 1$ and $K = |\mathcal{D}|$.

Clustering with $K = 1$ cluster. When we have a single cluster, then $c_0^{(i)} = c_1^{(i)}, \forall i$. This reduces the cost of Equation (4) to:

$$c((x_0, c_0), (x_1, c_1)) = \|x_0 - x_1\|^2 \tag{7}$$

As already explained in Appendix E.1, this renders COT-CFM (and consequently COT Policy) equivalent to OT-CFM.

Clustering with $K = |\mathcal{D}|$ clusters. When the number of clusters is the same as the dataset size, then every observation $o^{(i)}$ acts as a separate cluster and therefore $c_1^{(i)} \neq c_1^{(j)}, i \neq j$. The noise conditions in B_0 are random permutations of those in B_1 and therefore $c_0^{(i)} \neq c_0^{(j)}, i \neq j$ and $c_0^{(i)} \neq c_1^{(j)}, i, j \neq i, p(i)$, where $p(i)$ is the index of the sample of B_1 that was permuted to sample i in B_0. This basically means that every noise condition has exactly one identical target condition. Assuming a large enough γ that renders the second term of Equation (4) dominant, a coupling that would make it equal 0 would be the global optimum of the EMD optimization. That coupling exists and is constructed by the pairs $i, p(i)$, for $i = 1, \ldots, |B_1|$ since $c_0^{(i)} = c_1^{(p(i))}$. This is equivalent to a random permutation and therefore there is no dependency between the coupled samples. For this reason, COT-CFM with $K = |\mathcal{D}|$ is equivalent to (I)-CFM.