Temporal Policy: History-Initialized Action Generation for Robotic Learning from Demonstration

by

Dylan Miller

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Generative modeling has emerged as a powerful paradigm for robotic imitation learning, enabling policies to capture complex, multimodal action distributions from expert demonstrations. However, predominant approaches such as Diffusion Policy typically rely on iterative denoising from uninformative Gaussian noise. This approach forces the network to bridge a significant geometric gap between a high-entropy prior and the physical configuration space, resulting in long and potentially high-curvature transport paths. This transport complexity acts as a fundamental bottleneck, imposing computational burdens and inference latency that limit deployment on real-time, resource-constrained robotic systems.

This thesis introduces **Temporal Policy**, an action generation framework formulated using stochastic interpolants. Our approach replaces the uninformative noise source distribution with the robot's recent state history. Mapping recent state history to future actions provides a structurally simpler regression objective, allowing the model to learn a direct, low transport cost vector field. This enables fast sampling in the robot's configuration space rather than an abstract noise space. Furthermore, the stochastic interpolant framework allows a single trained network to be deployed using either low-latency deterministic sampling or robust stochastic sampling.

Empirical evaluations on visuomotor simulation benchmarks and a physical 7-DoF WAM teleoperation platform validate this geometric efficiency. On

the Robomimic benchmark suite, Temporal Policy matches state of the art success rates, while reducing transport costs by nearly an order of magnitude compared to standard baselines. This is further demonstrated through single step inference, where our method outperforms noise initialized baselines. These attributes allow low-order ODE solvers to achieve high-fidelity action generation with a minimal number of integration steps, enabling higher frequency, real-time control. Furthermore, unlike methods operating in abstract noise spaces, the resulting vector fields act directly as physical kinematic velocities.

Preface

This thesis presents my own original research under the supervision of Dr. Jagersand. Guided by his emphasis on integrating robotics-specific properties into general learning approaches, this work originated from an investigation into the success of Diffusion Policies in robotic Learning from Demonstration. While diffusion models capture multimodal behavior, their use of an uninformative prior ignores important robotic context. This motivated a transition toward more general transport formulations that utilize the robot's previous state history as an informative prior. This work has been accepted for publication at IROS 2026.

In tandem with this theoretical development, significant effort was dedicated to the engineering of high-fidelity data collection infrastructure. The generative models proposed in this thesis are data-driven, therefore, the quality of the expert demonstrations is paramount. To achieve this, I helped develop a low-latency teleoperation system designed to capture the nuance of human manipulation. I was the primary lead on the software development, including low-latency video streaming, communication between bilateral arms, and interfacing with the Barrett Hand. Amir Noohian and Faezeh Haghverd primarily developed the control system, which has been accepted to IROS 2026 [35]. Justin Valentine developed the hardware for our custom haptic wrist.

Through these efforts we were able to travel to Ottawa to participate in the Robot Roundup at the IDEaS Marketplace and publish a conference paper [24].

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Martin Jagersand. His incredible wealth of knowledge, spanning hardware design, theory, and practical, hands-on robotics, has been an invaluable resource throughout my studies. I am deeply appreciative of his guidance and his ability to bridge the gap between abstract concepts and real-world systems.

I also extend my sincere thanks to my colleagues in the lab, in particular, Justin Valentine, Amir Noohian, and Faezeh Haghverd. Their collective effort in developing the teleoperation system was fundamental to this work, enabling my research in Learning from Demonstration.

I gratefully acknowledge the financial support provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Canada Graduate Scholarships Master's (CGS M) program and the Faculty of Graduate Studies and Research through the Alberta Graduate Excellence Scholarship (AGES) program, whose backing was instrumental to this project.

Most importantly, I thank my wife. Her endless support, patience, and understanding were the foundation that made the completion of this thesis possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advances in robot autonomy have the potential to revolutionize numerous domains, shifting focus from repetitive industrial automation to general-purpose robots capable of operating in unstructured environments. Traditional approaches struggle in these dynamic environments because manually encoding every physical edge case is impractical. Humans easily navigate unstructured tasks by relying on implicit physical intuition.

To operate in such environments, Learning from Demonstration (LfD) has emerged as a promising paradigm, allowing robots to acquire complex behaviors directly from expert demonstrations without the need for manual programming or reward engineering [43]. However, the effectiveness of LfD hinges on the expressivity of the underlying policy representation [5]. While early methods struggled with multi-modal behavior, modern generative models have unlocked new capabilities, making significant progress towards policies that capture the rich, multi-modal distributions of complex task execution. In particular, diffusion models and flow matching techniques have demonstrated remarkable capabilities in challenging manipulation tasks [8], [11], [41], [60].

Despite these advances, the standard formulation of these models typically generates data by initializing samples from an uninformative noise distribution. This approach forces the model to bridge a significant gap between the high-entropy prior and the physical configuration space, resulting in long and potentially high-curvature transport paths. This complexity acts as a fundamental bottleneck, imposing computational burdens and data requirements

that limit deployment in real-time, resource-constrained robotic systems.

This thesis seeks to address key limitations of current state-of-the-art generative models for robot LfD. We introduce **Temporal Policy**, a framework built on a simple but powerful concept yet to be exploited in robotics: initializing the generative process directly from the robot's recent state history to predict future action sequences. By reframing the generation problem as a bridge between the past and future state, rather than noise and data, Temporal Policy aims to bypass the computation barriers of current noise initialized generative models.

The primary contributions of this work are:

- **Temporal Coupling Formulation:** A data-dependent generative modeling policy that explicitly couples past state history to future actions. This proximity results in nearly an order-of-magnitude reduction in transport cost compared to standard noise-initialized baselines.
- **Computational Efficiency:** By simplifying the target vector field, the framework enables the use of low-order Ordinary Differential Equation (ODE) solvers requiring very few integration steps. This reduction in network evaluations enables higher-frequency real-time control.
- **Empirical Validation:** Evaluation across the Robomimic simulation benchmarks, demonstrating competitive performance and superior data efficiency, alongside successful real-world deployment on a physical 7-DoF teleoperation platform.

Beyond these empirical gains, the formulation provides inherent theoretical advantages for interpretability. Standard generative models suffer from a semantic gap, where intermediate integration steps lack physical meaning. By anchoring both the initialization and the target in the robot's state space, Temporal Policy generates intermediate steps that remain close to valid configurations throughout the generation process. This physically grounded formulation lays the foundation for future work in safety-critical, interpretable control strategies.

The overarching goal of this thesis is to advance robot autonomy by leveraging Learning from Demonstration (LfD) in a physically grounded framework. By bridging the expressivity of generative modeling with the structural advantages of state-based initialization, we contribute to the broader vision of autonomous systems that are efficient, safe, and capable of functioning seamlessly in the real world.

1.1 Contribution and Contents

The remainder of this thesis is organized as follows:

Chapter 2, Literature Review, provides a comprehensive review of Learning from Demonstrations (LfD). It traces the evolution of policy learning from classical behavior cloning to modern generative approaches. Special attention is given to the recent adoption of Diffusion Models, Flow Matching, and Stochastic Interpolants within the robotics domain, highlighting their advantages over previous methods in modeling multimodal action distributions. Finally, the chapter examines emerging research in data-dependent generative modeling and bridge matching, identifying the limitations of standard noise-based priors that motivate our proposed approach.

Chapter 3, Background, establishes the necessary mathematical background and theoretical foundations for the proposed work. It provides a brief introduction to Stochastic Differential Equations (SDEs) and details the formulations of Diffusion Models and Conditional Flow Matching, with a focus on Stochastic Interpolants as the unifying framework. Furthermore, the chapter examines the role of Optimal Transport in constructing efficient probability paths and covers the theoretical handling of paired data distributions, which serves as the basis for the methodology developed in subsequent chapters.

Chapter 4, Methodology, details the design and implementation of Temporal Policy. We introduce a novel generative formulation that initializes the generative process from the robot's state history, significantly reducing the transport distance compared to mapping from noise. The chapter details the

architecture, the stochastic interpolant training objective, and the unified inference mechanism that recovers the score function to enable both deterministic and stochastic sampling.

Chapter 5 provides an empirical evaluation of the proposed method. We report results from both simulation environments and real-world experiments. The analysis focuses on key performance metrics, including success rate and inference speed, demonstrating the efficacy of our approach compared to state-of-the-art baselines.

Finally, Chapter 6 concludes the thesis by summarizing the key findings and discussing potential avenues for future research in generative robot autonomy.

Chapter 2

Literature Review

In this chapter, we review the landscape of Learning from Demonstration, beginning with an overview of generative modeling approaches in robotics. We then narrow our focus on modern techniques, specifically diffusion models, flow matching and the unifying stochastic interpolants framework. Finally, we examine the critical role of coupling strategies within these models, discussing the limitations of current noise-based approaches and how they can be improved through structured, data-dependent formulations.

2.1 Foundations of Learning from Demonstrations

Learning from Demonstrations (LfD), also referred to as Imitation Learning, aims to acquire a control policy by mimicking the behavior of an expert, such as a human operator or an optimal control algorithm. Formally, we assume access to a dataset of expert trajectories $\mathcal{D} = \{(\mathbf{s}_0, \mathbf{a}_0), (\mathbf{s}_1, \mathbf{a}_1), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$, where $\mathbf{s}_t$ represents the robot state and $\mathbf{a}_t$ represents the corresponding expert action. The goal is to learn a policy π that maps states to actions such that the robot's behavior approximates the expert's strategy.

2.1.1 Behavior Cloning and Direct Regression

The most direct approach to LfD is Behavior Cloning (BC), which frames policy learning as a supervised regression problem. In this paradigm, the policy

π_θ is typically parameterized by a neural network and trained to minimize a loss function measuring the discrepancy between the predicted action and the ground truth expert action. The standard objective is the Mean Squared Error (MSE):

$$\mathcal{L}_{BC}(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\|\pi_\theta(\mathbf{s}) - \mathbf{a}\|_2^2\right] \tag{2.1}$$

While computationally efficient and simple to implement, naive Behavior Cloning suffers from mode-averaging. The MSE implicitly assumes the relationship between states and optimal actions is unimodal and deterministic. In reality, expert behavior is highly variable, and multiple valid strategies can achieve the same goal. When trained with a unimodal loss function on a multimodal distribution, the network minimizes error by predicting the arithmetic mean of the observed actions. As illustrated in Figure 2.1, averaging two valid trajectories can result in an invalid or dangerous prediction. This limitation is the primary driver for adopting generative models, which explicitly represent the underlying probability distribution $p(\mathbf{a}|\mathbf{s})$ to capture multimodal behavior.

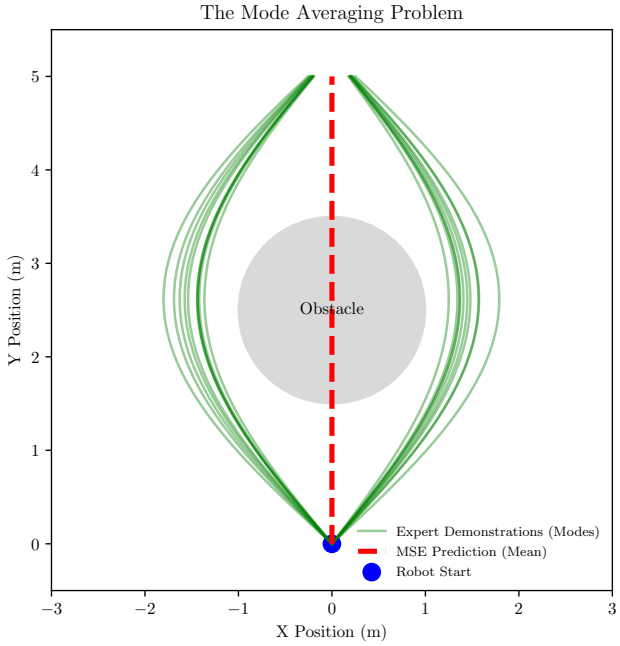


Figure 2.1: The mode averaging problem. When a unimodal regression model is trained on a multimodal distribution (e.g., go left or go right), the MSE loss minimizes error by predicting the mean (go straight), potentially leading to invalid or dangerous states.

2.2 Generative Models in Robotics

To overcome the mode-averaging limitations of deterministic regression, research in robot learning has increasingly adopted generative modeling. Rather than learning a function $f : \mathcal{S} \to \mathcal{A}$ that outputs a single action, generative approaches estimate the conditional probability distribution $p(\mathbf{a}|\mathbf{s})$. By sampling from this distribution, the policy can express multiple valid solutions for a given state, preserving the rich, multimodal nature of the expert's demonstrations.

Mixture Density Networks

One of the earliest approaches to modeling multimodal distributions in robotics was the Mixture Density Network (MDN) [7]. MDNs parameterize a Gaussian Mixture Model (GMM) using a neural network, outputting a set of mixture weights, means, and variances rather than a deterministic action.

While MDNs can theoretically capture multiple modes, they suffer from significant numerical instability during training and often struggle to scale to high-dimensional action spaces, such as the joint space of a 7-DoF arm. Furthermore, the number of mixture components K must be fixed *a priori*, limiting the model's flexibility in unstructured environments where the complexity of the solution space is hard to ascertain.

Variational Autoencoders

As deep generative models gained traction in robotics, Variational Autoencoders (VAEs) [26] became a standard tool for learning latent action representations [32], [34]. A prominent recent example is Action Chunking with Transformers (ACT) [63], which successfully combines a Conditional VAE with a Transformer backbone. While effective for specific fine-grained tasks, the standard VAE formulation typically formulates the decoder as a unimodal Gaussian, which remains susceptible to the mode averaging problems discussed above. This inherently biases the network toward over-smoothed trajectory predictions when faced with multiple valid strategies. In complex environments

where averaging two valid paths results in an invalid action, this mean-seeking behavior degrades performance.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) [19], [20] address VAE over-smoothing by training a generator to output actions that match expert demonstrations. Although this adversarial objective can theoretically capture the full data distribution, the optimization is notoriously difficult. GANs frequently experience *mode collapse* during training, collapsing to a single output rather than covering the entire multimodal distribution. This optimization challenge highlights the need for models that can provide both multimodal coverage and stable training.

Energy-Based Models

To model complex action distributions, Energy-Based Models (EBMs) [27], such as Implicit Behavioral Cloning (IBC) [17], learn a scalar energy function $E(\mathbf{s}, \mathbf{a})$ rather than an explicit policy. Expert demonstrations are assigned low energy, and inference requires finding the global minimum:

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} E(\mathbf{s}, \mathbf{a}) \tag{2.2}$$

While IBC effectively captures multimodal action distributions, pushing the computational burden to the inference phase creates a bottleneck for dynamic tasks. Solving this iterative optimization at every control step is computationally expensive and frequently traps the policy in local minima due to the rugged energy landscape.

2.3 Diffusion Models

Diffusion models [21, 46] are a class of generative models that learn to reverse a forward diffusion process which systematically degrades data with Gaussian noise. Rather than generating samples in a single step, these models perform

iterative denoising, enabling them to represent complex, multimodal distributions. In the context of LfD, this iterative refinement allows the policy to generate high-fidelity action sequences that capture diverse expert strategies.

Chi et al. [11] introduced **Diffusion Policy**, demonstrating the effectiveness of diffusion models for visuomotor control. Their approach conditions the denoising process on a history of observations, including images and robot states, and generates a sequence of future actions via an action chunking formulation. This enables the policy to produce temporally consistent action sequences while maintaining closed-loop feedback.

Empirically, Diffusion Policy achieves strong performance across both simulated and real-world manipulation tasks, outperforming prior behavior cloning methods such as implicit behavior cloning (IBC) and recurrent policies. A key advantage of this approach is its representational expressivity, as its ability to model complex, multimodal distributions directly from demonstrations allows it to capture multiple distinct ways to solve a task. This enables the agent to commit to a single successful trajectory rather than producing the mean-seeking behaviors characteristic of unimodal baselines.

2.3.1 Sensory and Task Conditioning

Several subsequent works built upon Diffusion Policy by incorporating additional input modalities. Conditioning on depth images or point clouds was shown to increase data efficiency while preserving high task performance [60]. This 3D representation was further enhanced by Wang et al. [55], who integrated semantic descriptors from foundational vision models such as DINOv2 [37].

Beyond visual data, researchers have leveraged natural language conditioning for flexible task specification. Frameworks like the Multimodal Diffusion Transformer (MDT) focus on handling goals specified through different modalities [41], while others utilize language to scale control policies across extended time horizons [61]. Furthermore, language conditioning is a key component in efforts to integrate diffusion models into Large Vision-Language-Action (VLA)

architectures [57].

To address tasks involving significant physical contact, recent works have incorporated force-torque or tactile feedback. Approaches like ForceMimic [29] and others [1] train policies conditioned on force data, enabling the execution of contact-rich tasks like insertion or wiping that are difficult to infer from vision alone.

2.3.2 Architectural Variations and Geometric Priors

Another significant line of research focuses on enhancing diffusion policies by refining the core network architecture, particularly through geometric inductive biases. Standard network architectures often require vast amounts of data to learn spatial symmetries. **Equivariant Diffusion Policies** [54] address this by building symmetries (such as $SE(3)$ invariance) directly into the network structure, improving sample efficiency.

Furthermore, because robot end-effector poses reside on curved manifolds like $SE(3)$, standard Euclidean noise injection can be problematic. Works such as **SPOT** [22] formulate the diffusion process directly on the manifold. While theoretically rigorous, these methods often introduce significant implementation complexity compared to standard diffusion.

2.3.3 Inference Efficiency

A primary limitation of diffusion based policies in high-frequency control is the computational cost of iterative denoising. To mitigate this, several works have adopted advanced numerical solvers [25], [31] to allow for high-quality sampling in significantly fewer iterations than standard approaches.

Alternatively, distillation can be used to compress the multi-step denoising process into a few-step or single-step student network. Consistency Models [47, 48] facilitate this by enforcing a self-consistency property that maps any point along a probability path directly to the data origin. This strategy was successfully applied to robotics by Prasad et al. [40] and Yan et al. [59], allowing policies to bypass iterative numerical integration. Similarly, Wang et al. [56]

introduced **One-Step Diffusion Policy**, which employs diffusion distillation to provide the low-latency inference required for real-time robotic control.

Despite these improvements, diffusion based methods remain fundamentally constrained by the curved, noise-dependent trajectories of the underlying process. Overcoming this structural bottleneck requires moving beyond standard diffusion to more flexible frameworks.

2.4 Stochastic Interpolants and Flow Matching

While diffusion models are highly expressive, their reliance on Gaussian priors and iterative denoising often leads to high computational costs and complex, curved trajectories. To address these limitations, this thesis utilizes the framework of **Stochastic Interpolants** [2], which provides a unified mathematical foundation for diffusion and flow-based models. By parameterizing the bridge between two distributions, this framework enables the construction of generative paths between arbitrary source and target distributions, rather than being restricted to isotropic noise.

A prominent realization of this framework is **Conditional Flow Matching (CFM)** [28]. CFM reframes generative modeling as a regression task on a probability flow's velocity field. Unlike diffusion, which learns to reverse a noise process, CFM learns to approximate the time-dependent vector field that transports probability mass along direct paths.

The primary advantage of this formulation is geometric. By formulating probability paths using linear interpolants, a predominant design choice within the CFM framework, CFM generates straighter velocity fields with significantly lower curvature than those found in diffusion-based approaches. This geometric regularity reduces the representational burden on the neural network and enables the use of efficient, large-step ODE solvers during inference. The resulting inference speed has made it an increasingly popular choice in robotics control [8], [15].

2.4.1 Generalized Couplings

A critical advantage of the Stochastic Interpolant framework is the flexibility of the joint distribution $q(x_0, x_1)$ used to pair training samples. While standard CFM typically assumes source and target are sampled independently, alternative couplings can alter the geometry of the velocity field. Specifically, couplings that reduce path crossings lead to straighter marginal flows.

Recent works have introduced specific coupling strategies to minimize trajectory curvature. Pooladian et al. [39] and Tong et al. [50] propose **Optimal Transport (OT-CFM)**, which pairs source samples x_0 with target data x_1 via Optimal Transport. While solving for the exact OT coupling over the entire dataset $\mathcal{D}$ is computationally prohibitive (generally $O(N^3)$), these works demonstrate that the coupling can be effectively approximated using **Minibatch OT**. During training, rather than pairing samples randomly within a minibatch, OT-CFM solve a discrete OT problem to pair samples $\mathbf{x}_0^{(i)}$ and $\mathbf{x}_1^{(j)}$ strictly within the batch. In the robotics domain, Sochopoulos et al. [45] demonstrated the practical value of this approach, achieving a $10\times$ inference speedup compared to Diffusion Policy on manipulation tasks.

Alternatively, Liu et al. [30] introduce **Rectified Flow**, an iterative procedure that progressively straightens probability paths to enable few-step generation. This approach refines the vector field by retraining the model on synthetic pairs $(\hat{x}_0, \hat{x}_1)$ generated by its own ODE trajectories. This process, known as **ReFlow**, effectively transforms an initial independent coupling into a deterministic one where paths are increasingly linear and non-crossing.

However, these approaches apply couplings between an uninformative Gaussian source distribution and the target data distribution. Using a Gaussian prior discards the rich contextual information available in robotic control tasks. This limitation motivates a fundamental shift in the generative framework, moving away from uninformative noise toward the targeted formulations discussed next.

2.5 Data-Dependent Couplings

The standard paradigm of mapping Gaussian noise to data is ill-suited for robot control tasks where the initial state is known. By exploiting this known starting point, we can define a generative process that is structurally simpler and computationally more efficient. This insight has led to the development of **Data-Dependent Couplings**, which replace the standard Gaussian prior with informative distributions.

Historically, the concept of learning a generative process between two arbitrary distributions is formally captured by **Schrödinger Bridges (SB)** [44]. Unlike standard diffusion, SB formulations seek the entropy-regularized optimal transport map between a source p_0 and target p_1. Iterative proportional fitting is a common approach to approximate the SB problem [12], [44]. This procedure alternates between training forward and backward SDEs, however, it requires numerical simulation (i.e., iteratively solving the SDE) during training to generate sample trajectories. This dependence on ODE/SDE simulation introduces significant computational bottlenecks and potential training instabilities.

To bypass this computational burden, recent works construct informative priors directly within the **Stochastic Interpolant** framework. Notably, these methods are simulation-free, meaning the transport can be learned without ever needing to integrate the generative path during training. Albergo et al. [3] demonstrate the utility of data-dependent couplings in the image domain. Specifically, they define the source distribution p_0 as a degraded version of the target, such as a masked image for in-painting or a downsampled image for super-resolution. Their experiments show that initializing the flow with these informative priors yields significantly higher generation quality compared to standard Gaussian baselines.

Building on this concept, Brinke et al. [9] apply "physics-informed" initialization to geometric time-series data, such as N-body simulations and molecular trajectories. Instead of initializing the flow from uninformed noise, they

construct the source samples x_0 by projecting the current state forward in time. Specifically, they use the average velocity from the observed history to generate a source trajectory via a biased random walk. This effectively treats the source sample x_0 as a linear inertial continuation of the system's past motion. By placing the source distribution geometrically closer to the target (x_1), this strategy simplifies the flow matching objective, requiring the model to learn only the residual correction rather than the full trajectory from scratch. Their ablation studies demonstrate that this informed initialization reduces prediction error compared to standard uninformed priors. However, an algorithmic limitation of this approach arises from its reliance on the CFM objective, which is restricted to deterministic sampling. This constraint is notable because stochastic sampling via SDEs has been generally observed to yield superior sample quality compared to deterministic transport [2], [47]. While the physics-informed prior simplifies the learning task, the framework remains unable to leverage the performance benefits associated with stochastic generation.

To address this limitation, Zhang et al. [62] propose **Trajectory Flow Matching (TFM)** for clinical time-series modeling. Observing that different patient outcomes often emerge from identical initial physiological states, they condition the vector field on a history window of past observations $x_{[t-h,t]}$. To capture the remaining uncertainty, they train a secondary network to explicitly parameterize the diffusion term alongside the drift. This dual-network approach was also used to independently estimate the drift and score fields [51]. While these strategies successfully enable stochastic generation, they introduce a significant trade-off in terms of architectural complexity and model scale. The requirement for auxiliary networks to facilitate stochasticity increases the total parameter count and complicates the training pipeline.

Chen et al. [10] resolves the complexity of this dual network design by utilizing the Stochastic Interpolant framework to transport a single point (a Dirac mass at the current state, δ_{x_0}) to an arbitrary target distribution. Crucially, they demonstrate that the drift for this process is non-singular and can be

learned efficiently via square loss regression. This formulation allows the score to be reconstructed directly from the drift without training a separate network. Because the diffusion noise schedule can be modified at inference time, it supports both deterministic ODE and stochastic SDE sampling. While this framework offers several desirable properties, it remains unexplored in the context of robotic LfD. Consequently, we adopt this framework as the primary theoretical basis for the method proposed in this thesis.

2.5.1 Non-Gaussian Initialization in Robotic LfD

Despite these theoretical advancements in distribution-to-distribution mapping, only a few recent works have attempted to utilize non-noise sources specifically within the robotics domain.

Gao et al. [18] propose **VITA (Vision-to-Action)**, a flow matching policy that proposes using latent visual representations as the generative source. Instead of starting from Gaussian noise, the flow is initialized at the latent visual representation z_{vision}, effectively bridging perception to action. However, this approach operates entirely within a learned latent space. Crucially, it requires backpropagating losses through the ODE solver to learn an effective decoder, significantly increasing the computational overhead and training complexity.

Alternatively, Jiang et al. [23] address the computational latency of diffusion models with their **Streaming Flow Policy (SFP)**. Standard diffusion policies require a costly denoising loop to generate a full trajectory plan before the first action can be executed, introducing significant inference lag. SFP eliminates this bottleneck by equating the flow integration time with the physical execution time. Instead of generating a plan *a priori*, the model streams actions on-the-fly by integrating a learned vector field initialized at the robot's previous action a_{t-1}. This reformulation effectively distributes the computational burden of generation over the execution horizon, enabling high-frequency control while retaining the ability to model multimodal velocity fields. However, this efficiency entails a theoretical trade-off. By sequentially pushing a_t to a_{t+1}, the method targets per-timestep marginal distributions

rather than the joint distribution over the action sequence $p(a_{1:T})$. This lack of temporal coherence can cause the policy to artificially mix distinct behaviors during execution.

In summary, the literature reveals distinct limitations in robotic LfD. Standard noise-initialized generative models successfully capture multimodality, but they are often characterized by high computational costs and inference latency. This is primarily because samples must traverse long transport distances from a Gaussian prior to the target data manifold. Attempts to address this in robotics either require numerical simulation during training or fail to capture the joint distribution of an action sequence.

This thesis addresses this gap by formalizing the **Temporal Policy** framework. By leveraging the Stochastic Interpolants framework, we construct a history-initialized generative process that bridges the robot's past state directly to a multimodal distribution of future actions. This formulation reduces the complex state-to-action transport problem to a stable, single-network regression task that is capable of deterministic (ODE) or stochastic (SDE) sampling. Operating directly in the interpretable physical configuration space, Temporal Policy enables efficient control and generates consistent action segments.

Chapter 3

Background Material

This chapter establishes the mathematical foundations for the generative modeling techniques utilized in this thesis. We formulate the Learning from Demonstration problem as learning a policy π that maps observations to action sequences. We start with the discrete-time diffusion framework commonly used in robotics, before advancing to the continuous-time formulations, Stochastic Differential Equations (SDEs), Flow Matching, and Stochastic Interpolants, that form the core of our proposed method.

3.1 Problem Formulation

We consider a robotic system with state space $\mathcal{S}$ and action space $\mathcal{A}$. The goal of LfD is to learn a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ from a dataset of expert demonstrations $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$, where each trajectory τ consists of state-action pairs. In the context of generative modeling, we treat the policy as a conditional distribution $p(\mathbf{a}_t|\mathbf{s}_t)$ and seek to learn a model that allows us to sample actions $\mathbf{a}_t$ that are consistent with the expert distribution.

3.2 Notation

Throughout this thesis, we adopt the following notational conventions to distinguish between scalars, vectors, and probabilistic quantities. We use bold lowercase letters for vectors (e.g., $\mathbf{x}, \mathbf{u} \in \mathbb{R}^d$) and bold uppercase letters for matrices (e.g., $\mathbf{I}, \mathbf{\Sigma}$). Standard unbolded letters denote scalars (e.g., t, β). Unless

otherwise stated, all vectors are assumed to be column vectors.

We consider continuous-time processes indexed by $t \in [0, 1]$. For brevity, we often use subscripts to denote time dependence, i.e., $\mathbf{x}_t \equiv \mathbf{x}(t)$. The time derivative is denoted by a dot, $\dot{\mathbf{x}}_t \equiv \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}$.

We denote the probability density function (PDF) of a random variable $\mathbf{x}$ as $p(\mathbf{x})$. For time-dependent processes, $p_t(\mathbf{x})$ denotes the marginal density of the random variable $\mathbf{x}_t$ at time t. We use $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ to denote the score function (the gradient of the log-density with respect to the state $\mathbf{x}$). The expectation $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$ denotes the expected value of $f(\mathbf{x})$ where $\mathbf{x}$ is drawn from $p(\mathbf{x})$.

3.3 Discrete Diffusion (DDPM)

Before defining the continuous-time models that form the core of this thesis, we briefly review Denoising Diffusion Probabilistic Models (DDPM) [21], as this discrete framework serves as the foundation for the popular Diffusion Policy utilized in robotic imitation learning [11].

DDPM defines a forward diffusion process as a fixed Markov chain that gradually adds Gaussian noise to the data $\mathbf{x}_0$ over N discrete steps. The transition from step $t-1$ to t is given by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{3.1}$$

where $\{\beta_t\}_{t=1}^N$ is a variance schedule. As N becomes large, $\mathbf{x}_N$ approaches an isotropic Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

A key property of this Gaussian process is that we can sample an arbitrary step $\mathbf{x}_t$ directly from $\mathbf{x}_0$ without iterative simulation. Defining $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$, the marginal distribution is:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{3.2}$$

Using the reparameterization trick, a sample at timestep t can be expressed as $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The generative process is the learnable reverse Markov chain, which removes noise to recover the data:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{3.3}$$

To train this model, we optimize the variational lower bound (ELBO) on the negative log-likelihood. Crucially, this optimization is tractable because the reverse process conditioned on the clean data, $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, is available in closed form. By matching $\boldsymbol{\mu}_\theta$ to the mean of this true posterior, Ho et al. [21] demonstrated that the mean can be parameterized as:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right). \tag{3.4}$$

This reveals that predicting the reverse step mean is equivalent to training a neural network $\boldsymbol{\epsilon}_\theta$ to predict the noise $\boldsymbol{\epsilon}$ injected at step t. Consequently, the model is trained using a simplified, unweighted mean squared error objective:

$$\mathcal{L}_{simple} = \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2]. \tag{3.5}$$

Once the noise-prediction network $\boldsymbol{\epsilon}_\theta$ is trained, generating new samples requires simulating the reverse Markov chain. Starting from pure Gaussian noise $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the model iteratively denoises the state down to $t = 0$. By substituting the learned noise predictor back into the tractable posterior mean, the sampling step from t to $t-1$ is defined as:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}, \tag{3.6}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $t > 1$, and $\mathbf{z} = \mathbf{0}$ for the final step $t = 1$. The variance term σ_t^2 is typically chosen as the forward schedule variance β_t.

This iterative denoising loop is one of the primary computational bottleneck of standard diffusion models. Generating a single sample requires N sequential network evaluations, introducing significant inference latency. To achieve greater flexibility, theoretical unification, and ultimately bypass this iterative bottleneck, modern approaches model diffusion as a continuous-time process.

3.4 Foundations of Stochastic Calculus

To understand the continuous-time generative models used in this work, we must formally define the noise process that drives them. Unlike deterministic systems governed by Ordinary Differential Equations (ODEs), our generative policies are modeled as SDEs driven by Brownian motion.

3.4.1 Wiener Process (Brownian Motion)

The fundamental building block of an SDE is the Wiener process (or Brownian motion), denoted as $\mathbf{w}_t$. Mathematically, the Wiener process is a continuous-time stochastic process with the following key properties:

1. **Initialization:** $\mathbf{w}_0 = 0$.
2. **Independent Increments:** For any $0 \leq s < t$, the increment $\mathbf{w}_t - \mathbf{w}_s$ is independent of the past values $\mathbf{w}_\tau$ for $\tau \leq s$.
3. **Gaussian Increments:** The increment follows a normal distribution:

$$\mathbf{w}_t - \mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, (t-s)\mathbf{I}). \tag{3.7}$$

Intuitively, $\mathbf{w}_t$ represents the cumulative effect of infinitely many random kicks. Crucially, while the path of $\mathbf{w}_t$ is continuous everywhere, it is nowhere differentiable. This "jagged" nature, illustrated in Figure 3.1, means we cannot use standard calculus (e.g., the chain rule) and must instead rely on **Itô's Lemma**, which introduces a second-order correction term.

3.4.2 Itô SDEs

An Itô Stochastic Differential Equation describes how a state $\mathbf{x}_t$ evolves under the influence of both deterministic forces (Drift) and random noise (Diffusion):

$$\mathrm{d}\mathbf{x}_t = \underbrace{\mathbf{f}(\mathbf{x}_t, t)\,\mathrm{d}t}_{\text{Drift}} + \underbrace{g(t)\,\mathrm{d}\mathbf{w}_t}_{\text{Diffusion}}. \tag{3.8}$$

- **Drift $\mathbf{f}(\mathbf{x}_t, t)$:** Defines the **deterministic vector field** of the system. It represents the instantaneous velocity of the state, governing the expected evolution of the trajectory in the absence of stochastic fluctuations.

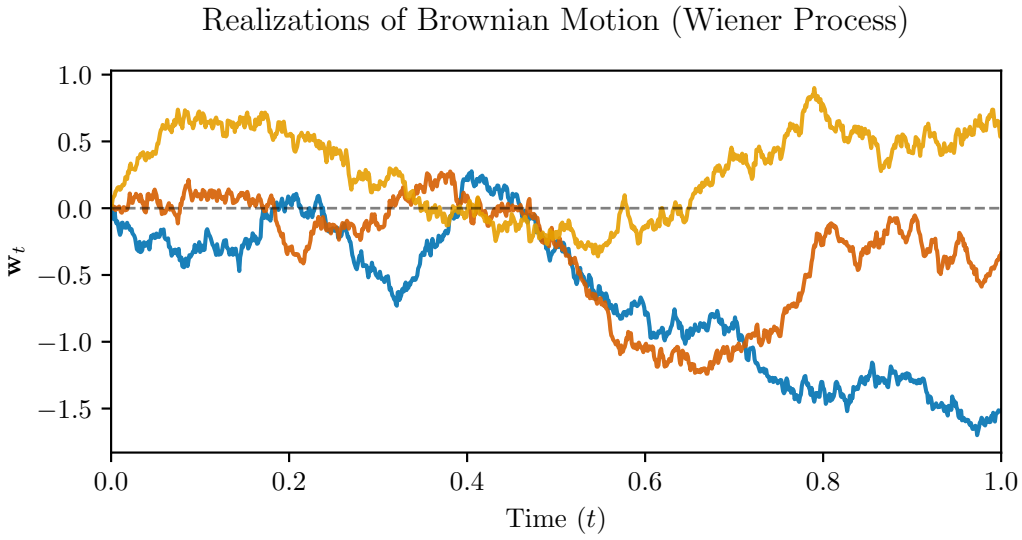


Figure 3.1: Realizations of the Wiener Process (Brownian Motion). Three sample paths of a one-dimensional Wiener process $\mathbf{w}_t$ starting from zero. Note the continuous yet jagged, nowhere-differentiable nature of the trajectories.

- **Diffusion** $g(t)$: Scales the magnitude of **continuous noise injection**. This stochastic forcing causes trajectories to diverge and branch out, enabling the model to generate samples from multimodal distributions even when initialized at a single deterministic state.

For a comprehensive background on SDEs, we refer the reader to Øksendal [36] and Särkkä and Solin [42].

3.4.3 The Fokker-Planck Equation

While the SDE in Eq. 3.8 describes the trajectory of a *single* sample, generative modeling is concerned with the evolution of the *entire probability density* $p_t(\mathbf{x})$. The bridge between the microscopic particle motion (SDE) and the macroscopic density evolution is the Fokker-Planck Equation (FPE), also known as the Kolmogorov Forward Equation.

For the SDE defined in Eq. 3.8, the time-evolution of the probability density $p_t(\mathbf{x})$ is governed by the partial differential equation:

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \underbrace{-\nabla \cdot (\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x}))}_{\text{Transport}} + \underbrace{\frac{1}{2} g(t)^2 \Delta p_t(\mathbf{x})}_{\text{Diffusion}}. \tag{3.9}$$

Here, $\nabla\cdot$ denotes the divergence operator and Δ denotes the Laplace operator. This equation reveals the two distinct mechanisms:

1. **Transport**: The divergence measures how the flow locally expands or compresses the density. It describes the **deterministic transport** of probability mass along the vector field $\mathbf{f}$.

2. **Diffusion**: The Laplacian term measures the local curvature of the density. It describes the **stochastic spreading** of the density due to the noise $g(t)$.

Intractability and Non-Uniqueness. Directly solving the FPE to compute $p_t(\mathbf{x})$ is computationally intractable for high-dimensional state spaces (e.g., robot configuration spaces), as grid-based PDE solvers suffer from the curse of dimensionality. Consequently, we do not solve for the density directly, instead, we learn vector fields that approximate the characteristics of the FPE.

Crucially, the mapping between the microscopic dynamics and the macroscopic FPE is not one-to-one. Multiple distinct stochastic processes can yield the exact same marginal density evolution $p_t(\mathbf{x})$. Most notably, for every SDE involving diffusion, there exists a corresponding deterministic Ordinary Differential Equation, termed the **Probability Flow ODE** [49], that transports the probability mass exactly according to Eq. 3.9 without injecting noise. This theoretical duality is what allows modern generative models to be trained using robust stochastic objectives (SDEs) while offering the option of efficient deterministic sampling (ODEs) during inference.

3.5 Continuous Diffusion Models

Song et al. [49] unified discrete diffusion into a continuous-time framework using SDEs. The forward process, illustrated in Figure 3.2, diffuses data $\mathbf{x}_0 \sim p_{data}$ to a prior $\mathbf{x}_T \sim p_{prior}$ via the Itô SDE:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \tag{3.10}$$

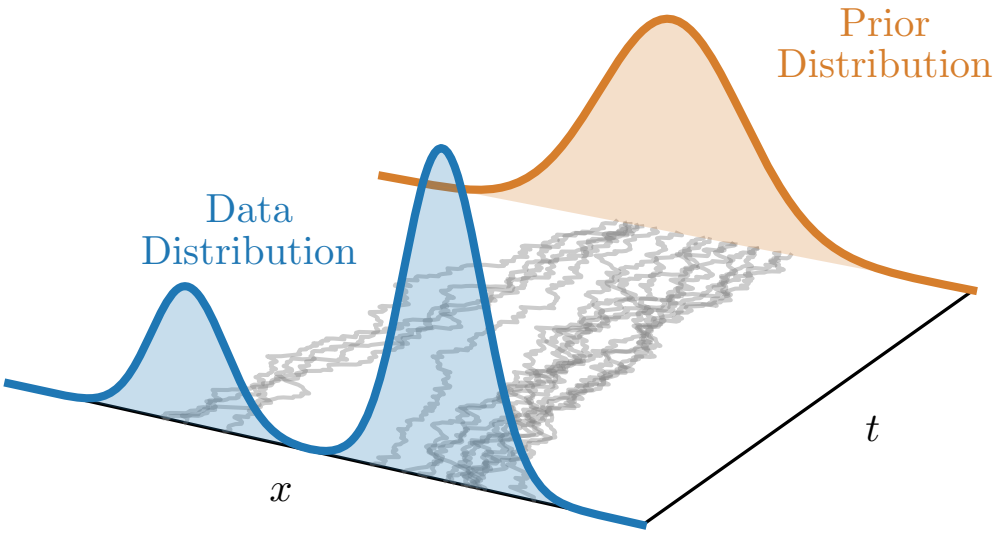


Figure 3.2: Visualization of score-based generative modeling. The diagram illustrates the diffusion process where a complex data distribution p_{data} (blue, $t = 0$) is gradually perturbed by a forward SDE (gray paths) into a simple prior distribution p_{prior} (orange, $t = T$).

where $\mathbf{f}(\cdot, t)$ is the drift coefficient, $g(t)$ is the diffusion coefficient, and $\mathbf{w}$ is a standard Wiener process. Remarkably, Anderson [4] showed that this process can be reversed by a corresponding reverse-time SDE:

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}\,, \tag{3.11}$$

where $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the **score function**, and $\bar{\mathbf{w}}$ denotes a reverse-time standard Wiener process.

Generative modeling thus reduces to *Score Matching*: training a neural network $s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ to estimate the gradient of the log-density. Once trained, samples are generated by numerically solving Eq. 3.11 using solvers like Euler-Maruyama.

3.5.1 Estimating the Score

To generate samples, we must numerically simulate Eq. 3.11 backwards from T to 0. However, the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is generally unknown analytically. While the probability density evolves according to the Fokker-Planck equation (Sec. 3.4.3), solving this PDE directly is intractable for high-dimensional data.

Instead, one can utilize **Denoising Score Matching**. If the transition kernel $p_t(\mathbf{x}_t|\mathbf{x}_0)$ is Gaussian (which is true for affine forward SDEs), the conditional score is analytically tractable. Assuming a transition kernel of the form $p_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \mu_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$, the conditional score is:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t - \mu_t\mathbf{x}_0}{\sigma_t^2}. \tag{3.12}$$

We can then train the network by minimizing the expected difference between the model output and this analytical conditional score:

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_t}\left[\|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0)\|^2\right] \tag{3.13}$$

Crucially, Vincent [53] proved that minimizing this denoising objective is equivalent to minimizing with the intractable marginal score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, up to a constant independent of θ. This conditioning technique, where conditioning on a data point turns an intractable loss into a tractable one is a recurring theme, reappearing later in Conditional Flow Matching.

The relationship between the score function and the clean data is formalized by **Tweedie's Formula** [14]. Using the same Gaussian notation as above (μ_t as the signal scaling factor and σ_t as the noise scale), the optimal estimate of the clean data (the posterior mean) is given by:

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}{\mu_t}. \tag{3.14}$$

This identity provides an interpretation of the score-based generative model, demonstrating that learning the score function $\nabla \log p_t$ is equivalent to learning a denoiser that predicts the clean action $\mathbf{x}_0$ from the noisy state $\mathbf{x}_t$ at every time step. Furthermore, this extends to the noise-prediction objective used in standard discrete diffusion models. By applying the reparameterization $\mathbf{x}_t = \mu_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$, the posterior mean can be rewritten in terms of the expected injected noise:

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{\mathbf{x}_t - \sigma_t \mathbb{E}[\boldsymbol{\epsilon}|\mathbf{x}_t]}{\mu_t}. \tag{3.15}$$

By training a neural network to approximate this noise, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx \mathbb{E}[\boldsymbol{\epsilon}|\mathbf{x}_t]$, and substituting this relation into Tweedie's formula, it follows that the predicted

noise is simply a scaled negative score function:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \approx -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sigma_t}. \tag{3.16}$$

While predicting the score, the clean state, or the injected noise are mathematically equivalent formulations, these choices produce different numerical properties during training, especially with regards to singularities.

3.6 Continuous Normalizing Flows and Flow Matching

While SDE-based diffusion is powerful, the trajectories it generates are stochastic and often "meandering," leading to slow sampling speeds. Furthermore, standard diffusion asymptotically maps the data distribution to an analytically tractable prior over an infinite time horizon, requiring artificial truncation in practice. **Flow Matching** [28] offers an alternative based on deterministic probability paths.

Instead of defining a rigid noise process, we define a probability path $p_t(\mathbf{x})$ that explicitly interpolates between two arbitrary distributions, a source p_0 and a target p_1, over a strictly finite time interval $t \in [0, 1]$. This path is generated by a time-dependent vector field $\mathbf{v}_t(\mathbf{x})$ via the Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\phi_t(\mathbf{x})}{\mathrm{d}t} = \mathbf{v}_t(\phi_t(\mathbf{x})), \quad \phi_0(\mathbf{x}) = \mathbf{x}. \tag{3.17}$$

This flow transforms the distribution according to the continuity equation:

$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathbf{v}_t) = 0. \tag{3.18}$$

3.6.1 Conditional Flow Matching (CFM)

Directly learning the marginal vector field $\mathbf{v}_t$ that satisfies the continuity equation is intractable. Traditional Continuous Normalizing Flows (CNFs) train this field by simulating the ODE forward to compute the likelihood, requiring expensive numerical integration at every training step. Lipman et al. [28] introduced **Conditional Flow Matching** to bypass this simulation bottleneck.

The core insight is that the intractable marginal vector field can be expressed as the exact conditional expectation of tractable, sample-specific conditional vector fields:

$$\mathbf{v}(\mathbf{x}, t) = \mathbb{E}_{p(\mathbf{x}_0, \mathbf{x}_1 | \mathbf{x}_t = \mathbf{x})} \left[\mathbf{u}_t(\mathbf{x} | \mathbf{x}_0, \mathbf{x}_1) \right]. \tag{3.19}$$

Given a source $\mathbf{x}_0 \sim p_0$ and target $\mathbf{x}_1 \sim p_1$, we define a conditional path $p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1)$. The most common choice is a linear interpolation scheme:

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1. \tag{3.20}$$

The corresponding conditional vector field is simply the constant velocity pointing from source to target:

$$\mathbf{u}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_1 - \mathbf{x}_0. \tag{3.21}$$

The objective is to regress a neural network $\mathbf{v}_\theta$ to this conditional field:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t, p(\mathbf{x}_0, \mathbf{x}_1), p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1)} \|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2. \tag{3.22}$$

This allows for simulation-free training. We simply sample a pair $(\mathbf{x}_0, \mathbf{x}_1)$, compute the intermediate point $\mathbf{x}_t$, and minimize the regression error against the straight-line velocity.

3.7 Stochastic Interpolants

The **Stochastic Interpolant** framework [2] unifies Diffusion (SDEs) and Flow Matching (ODEs) into a single mathematical formulation. This is the primary framework utilized in this thesis due to its flexibility in defining boundary conditions and dynamics.

A stochastic interpolant is a process $\mathbf{x}_t$ that interpolates between two arbitrary samples $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$:

$$\mathbf{x}_t = \alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{x}_1 + \gamma(t)\mathbf{z}, \quad t \in [0, 1], \tag{3.23}$$

where α, β control the signal mixing, γ controls the noise injection, and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is an auxiliary noise variable. To ensure the process strictly interpolates between the source $\mathbf{x}_0$ and target $\mathbf{x}_1$, these scalar functions must satisfy

the following boundary conditions:

$$\begin{aligned} &\alpha(0) = 1, && \beta(0) = 0, && \gamma(0) = 0 \\ &\alpha(1) = 0, && \beta(1) = 1, && \gamma(1) = 0 \end{aligned}$$

These constraints guarantee that $\mathbf{x}_{t=0} = \mathbf{x}_0$ and $\mathbf{x}_{t=1} = \mathbf{x}_1$ exactly, regardless of the intermediate noise. This framework is powerful because it allows us to derive both a deterministic ODE and a stochastic SDE that generate the same marginal probability path $p_t(\mathbf{x})$. The velocity field $\mathbf{v}(t, \mathbf{x})$ required for these equations is defined as the conditional expectation of the instantaneous velocity:

$$\mathbf{v}(t, \mathbf{x}) = \mathbb{E}[\dot{\mathbf{x}}_t | \mathbf{x}_t = \mathbf{x}]. \tag{3.24}$$

In practice, we learn this field by minimizing a square-loss objective similar to CFM:

$$\mathcal{L}(\theta) = \mathbb{E}[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \dot{\mathbf{x}}_t\|^2]. \tag{3.25}$$

The choice of the noise schedule $\gamma(t)$, modulates the "bridge" behavior of the process. For $\gamma(t) = 0$, we recover deterministic Flow Matching. For $\gamma(t) > 0$, we introduce process noise, which serves to spatially smooth the density p_t and the associated velocity field [2]. This smoothing expands the support of the probability path, enabling the network to learn a well-defined vector field over a wider region of the state space.

Once trained, the model can be sampled in several ways. The simplest approach is to solve the deterministic ODE:

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{v}_\theta(\mathbf{x}_t, t). \tag{3.26}$$

If stochastic sampling is required, one can also derive a corresponding SDE. When the source distribution is standard normal, learning the score $\nabla \log p_t$ is sufficient for both deterministic and stochastic sampling. However, for arbitrary source and target distributions, generally independent estimates for both the velocity $\mathbf{v}_\theta$ and score $\mathbf{s}_\theta$ are required. This leads to the following stochastic process:

$$\mathrm{d}\mathbf{x}_t = \left(\mathbf{v}_\theta(\mathbf{x}_t, t) + \frac{1}{2} g(t)^2 \mathbf{s}_\theta(\mathbf{x}_t, t)\right) \mathrm{d}t + g(t) \mathrm{d}\mathbf{w}_t \tag{3.27}$$

where $g(t)$ is the diffusion coefficient.

3.8 Inference: Numerical Integration

Once the time-dependent vector field $\mathbf{v}_\theta(\mathbf{x}, t)$ (and optionally the score field $\mathbf{s}_\theta(\mathbf{x}, t)$) has been learned, generating a sample from the target distribution p_1 reduces to an Initial Value Problem (IVP). Starting from the source sample $\mathbf{x}_0$, we simulate the dynamics forward in time from $t = 0$ to $t = 1$. Since analytical solutions are rarely available, we rely on numerical integration schemes that discretize the time interval $[0, 1]$ into N steps of size $\Delta t = 1/N$.

3.8.1 Deterministic Sampling (ODE)

When the process noise is zero ($\gamma(t) = 0$), the dynamics simplify to an Ordinary Differential Equation (ODE). The simplest method to solve this is the **Forward Euler** method. Given the state $\mathbf{x}_k$ at time step t_k, the update rule is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_\theta(\mathbf{x}_k, t_k)\Delta t. \tag{3.28}$$

While computationally efficient, Euler integration introduces discretization errors proportional to the curvature of the vector field. Higher-order solvers, such as **Runge-Kutta (RK4)** or **Dopri5** [13], can be used to improve accuracy at the cost of an increased number of function evaluations (NFE) per step, where each evaluation requires a full forward pass of the network. Crucially, because ODEs are deterministic, solving this equation from a fixed $\mathbf{x}_0$ will always yield the exact same trajectory $\mathbf{x}_{0\to1}$.

3.8.2 Stochastic Sampling (SDE)

To capture the multimodality required for robust robotic control, we often utilize the stochastic dynamics ($\gamma(t) > 0$) provided by the Stochastic Interpolant framework. This requires solving an SDE, for which standard ODE solvers are insufficient due to the non-differentiable nature of Brownian motion. The standard approach is the **Euler-Maruyama** scheme. Consistent with Eq.

3.27, we must include the score-based correction term to ensure the marginal distributions are preserved:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \underbrace{\left(\mathbf{v}_\theta(\mathbf{x}_k, t_k) + \frac{1}{2} g(t_k)^2 \mathbf{s}_\theta(\mathbf{x}_k, t_k)\right) \Delta t}_{\text{Corrected Drift}} + \underbrace{g(t_k)\sqrt{\Delta t}\, \mathbf{z}_k}_{\text{Noise}}, \tag{3.29}$$

where $\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard Gaussian noise term drawn at each step. The stochastic term actively injects noise throughout the integration. This ensures that even when initialized at a single deterministic point $\mathbf{x}_0$, the trajectory can "branch" into different modes of the target distribution, preventing the mode collapse typical of deterministic policies.

3.9 Couplings and Optimal Transport

The Stochastic Interpolant framework allows the source $\mathbf{x}_0$ and target $\mathbf{x}_1$ to be drawn from an arbitrary joint distribution, or *coupling*, denoted by $q(\mathbf{x}_0, \mathbf{x}_1)$. Understanding the geometry of this coupling is central to improving the efficiency of generative models.

3.9.1 The Geometry of Couplings

In standard diffusion models, the source and target are assumed to be independent. The coupling is simply the product of the marginals:

$$q_{\text{indep}}(\mathbf{x}_0, \mathbf{x}_1) = q(\mathbf{x}_0) q(\mathbf{x}_1). \tag{3.30}$$

This typically implies pairing a target sample $\mathbf{x}_1$ with an unrelated random noise sample $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Geometrically, as illustrated in Figure 3.3, this creates "tangled" trajectories. Because the source and target are uncorrelated, the sample paths cross over each other frequently, resulting in a learned marginal vector field $\mathbf{v}_t$ with high curvature. This highlights that linear conditional paths do not necessarily produce linear marginal paths.

High curvature is computationally expensive. Integrating a curved ODE requires many small steps (function evaluations) to avoid discretization error. To achieve fast generation, we desire straight trajectories, which are trivial to integrate numerically.

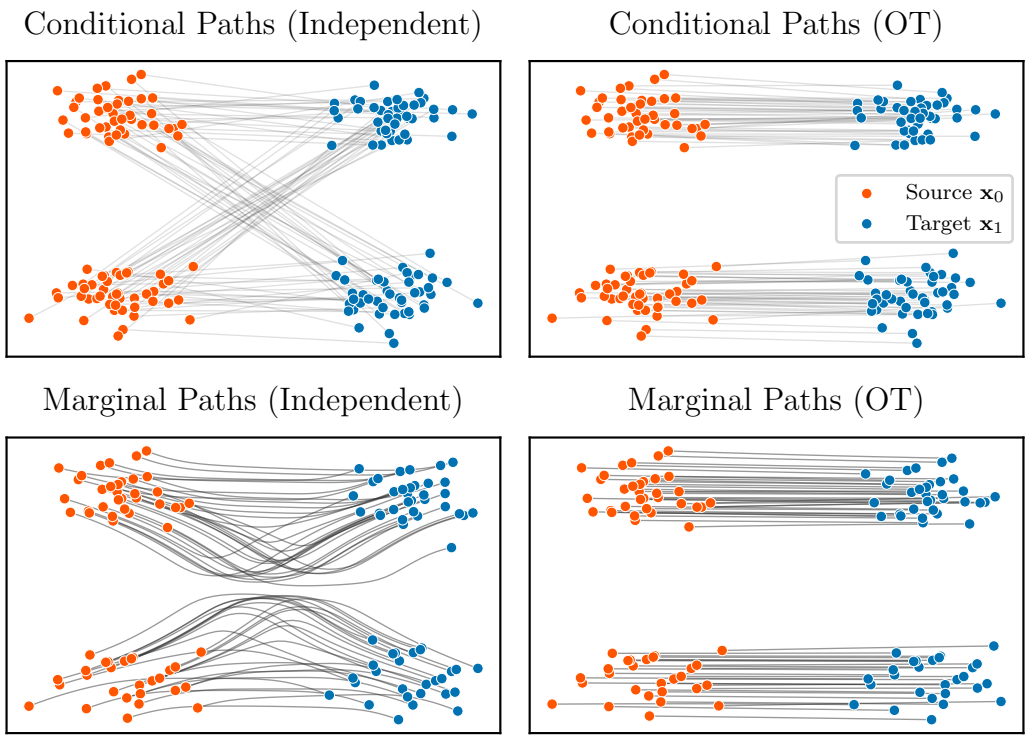


Figure 3.3: The geometry of couplings. Visualizing the trajectories connecting a source distribution (orange) to a target (blue). **Left Column: Independent Coupling.** Conditional samples are paired randomly (top left). While individual paths are linear, they intersect each other. The resulting marginal flows (bottom left) are highly curved, requiring many numerical integration steps to solve accurately. **Right Column: Optimal Transport Coupling.** Samples are paired to minimize the total transport distance (top right). The resulting flows are straight (bottom right), parallel and organized allowing for fast inference with minimal discretization error.

3.9.2 Bounding Vector Field Complexity

The efficiency of a coupling can be formalized using **Optimal Transport** (OT). Given a cost function $c(\mathbf{x}_0, \mathbf{x}_1)$ representing the "effort" to move mass from $\mathbf{x}_0$ to $\mathbf{x}_1$, the general Kantorovich OT problem seeks the coupling q^* that minimizes the total expected cost:

$$\mathcal{C}(q) = \int c(\mathbf{x}_0, \mathbf{x}_1) \mathrm{d}q(\mathbf{x}_0, \mathbf{x}_1). \tag{3.31}$$

In the context of generative modeling, a common choice is the quadratic cost $c(\mathbf{x}_0, \mathbf{x}_1) = \|\mathbf{x}_0 - \mathbf{x}_1\|^2$. The solution to this specific problem defines the 2-Wasserstein distance (W_2).

While the Kantorovich formulation (Eq. 3.31) describes the static pairing of mass, generative modeling requires the continuous time-evolution of the probability density p_t, which is established by the **Benamou-Brenier formulation** [6]. The goal is to find a pair $(p_t, \mathbf{v}_t)$ that minimizes

$$\inf_{p,\mathbf{v}} \int_0^1 \int_{\mathbb{R}^d} \|\mathbf{v}_t(\mathbf{x})\|^2 p_t(\mathbf{x}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}t, \tag{3.32}$$

subject to the continuity equation constraint $\partial_t p_t + \nabla \cdot (p_t \mathbf{v}_t) = 0$.

Because the kinetic energy integral heavily penalizes spatial deviations and high velocities, the energy-minimizing flows in the Euclidean metric are strictly straight lines traveled at a constant velocity.

To formally connect this physical property to the learning objective of flow matching and stochastic interpolants, we must relate the individual interpolant paths to the marginal vector field $\mathbf{v}_t(\mathbf{x})$. The marginal field is defined as the conditional expectation of the individual interpolant velocities passing through a point, $\mathbf{v}_t(\mathbf{x}) = \mathbb{E}[\dot{\mathbf{x}}_t \mid \mathbf{x}_t = \mathbf{x}]$. By applying Jensen's inequality to this conditional expectation, Albergo et al. [3] demonstrate that the kinetic energy of the learned flow is strictly bounded above by the expected kinetic energy of the interpolant:

$$\mathbb{E}[\|\mathbf{v}_t(\mathbf{x}_t)\|^2] = \mathbb{E}[\|\mathbb{E}[\dot{\mathbf{x}}_t \mid \mathbf{x}_t]\|^2] \leq \mathbb{E}[\mathbb{E}[\|\dot{\mathbf{x}}_t\|^2 \mid \mathbf{x}_t]] = \mathbb{E}[\|\dot{\mathbf{x}}_t\|^2]. \tag{3.33}$$

Thus, reducing the interpolant kinetic energy cost systematically bounds the kinetic energy and curvature of the marginal vector field. This theoretical property indicates that such couplings provide the neural network with a structurally simpler regression target. By mitigating unnecessary complexity in the vector field, these structured interpolants promote smoother learned dynamics, which facilitates faster numerical integration during inference.

Chapter 4

Methodology

This chapter details the methodology for Temporal Policy, our proposed framework for robot action generation. Unlike standard generative models that map uninformative noise to actions, Temporal Policy leverages a stochastic interpolant framework to formulate generation as a point-to-distribution transport problem. Its defining characteristic is the initialization of this generative process directly from the robot's recent state history. Figure 4.1 provides an overview of our method.

The chapter begins by establishing the structural equivalence between the state and action spaces (Section 4.1) before formally defining the temporal coupling that links the robot's history to its future trajectory (Section 4.2). We then formulate the stochastic interpolant required to learn this transition and derive the target vector field (Section 4.3). Finally, we detail the neural network architecture used to approximate this field (Section 4.4) and outline the inference procedures for flexible, real-time action execution (Section 4.5).

4.1 State-Action Representation

The primary objective of Temporal Policy is to map a robot's recent state history to a temporally coherent sequence of future actions. To achieve this, we enforce a strict structural equivalence between the robot's state space $\mathcal{S}$ and the action space $\mathcal{A}$. Specifically, both are defined by the robot's proprioceptive

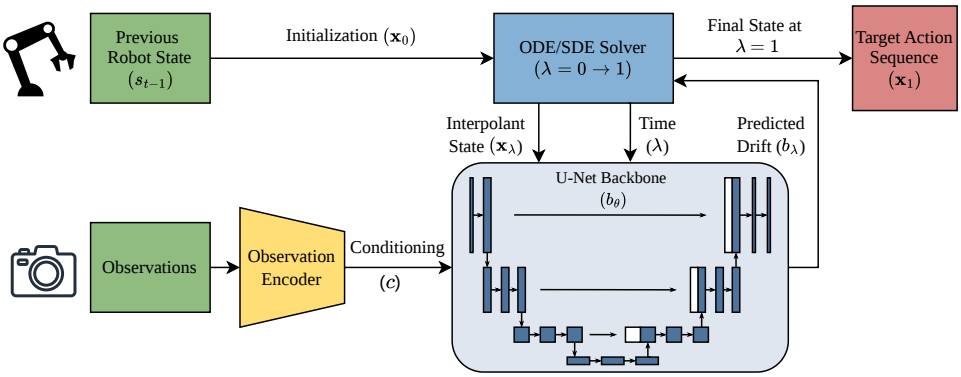


Figure 4.1: Overview of our Architecture. Unlike standard generative models that initialize from independent Gaussian noise ($\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$), our method explicitly initializes the generative process with the robot's recent state history. The architecture consists of: (1) An **Observation Encoder** that processes sensory observations into a conditioning vector c; (2) A **U-Net Backbone** that predicts the drift $b_\theta(\mathbf{x}_\lambda, \mathbf{x}_0, \lambda, c)$; and (3) An **ODE/SDE Solver** that integrates this vector field from $\lambda = 0$ to $\lambda = 1$ to generate the target action sequence.

configuration (e.g., joint positions or end effector pose)

$$\mathcal{S} \cong \mathcal{A} \cong \mathbb{R}^D \tag{4.1}$$

where D is the number of controlled degrees of freedom.

Our framework generates a sequence of H such actions, denoted as an action chunk $\mathbf{x}_1 = \{a_0, a_1, \ldots, a_{H-1}\}$. The length of the horizon H is a hyperparameter that is largely determined by the nature of the task and control frequency (H is usually 4-16). This action chunking strategy allows the model to generate more coordinated and extended behaviors compared to single-step action prediction.

4.2 Temporal Coupling Formulation

The foundation of our generative process relies on the specific construction of the source and target samples $\mathbf{x}_0$ and $\mathbf{x}_1$. In Section 3.9, we established that minimizing the transport cost between the source and target leads to straight, efficient probability flows. Rather than relying on algorithms like Minibatch OT to infer pairings between independent distributions, we exploit

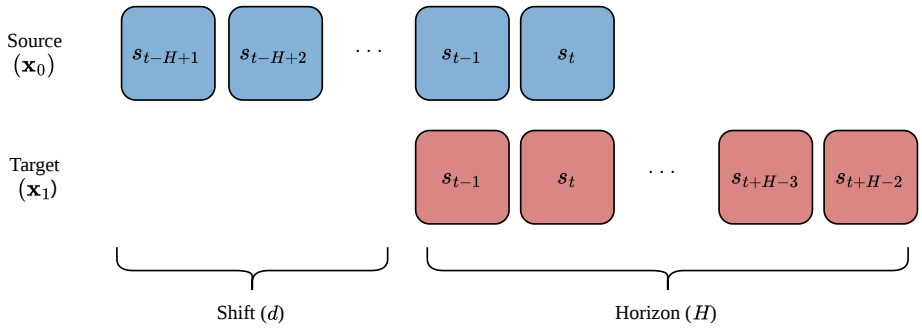


Figure 4.2: Diagram illustrating the temporal coupling of an action chunk with horizon length H and shift $d = H - 2$. An input history of states, labeled as the Source ($\mathbf{x}_0$) is coupled with a sequence of future states, labeled as the Target ($\mathbf{x}_1$).

the temporal structure of the demonstrations to derive a physically grounded coupling.

As shown in Figure 4.2, we explicitly define our source $\mathbf{x}_0$ as the robot's recent state history and the target $\mathbf{x}_1$ as the corresponding action sequence shifted forward by d discrete timesteps ($1 \leq d \leq H$):

- **Source ($\mathbf{x}_0$):** State history $s_{t-H+1:t}$.
- **Target ($\mathbf{x}_1$):** Future action chunk $s_{t-H+1+d:t+d}$.

Sequence overlap ($d < H$) between source and target helps reduce discontinuities between action chunks. This definition creates a data-dependent **temporal coupling** $q_{\text{data}}(\mathbf{x}_0, \mathbf{x}_1)$.

4.3 Stochastic Interpolant Framework

We leverage the stochastic interpolants framework [2], [10] to learn the transition from $\mathbf{x}_0$ to $\mathbf{x}_1$. Figure 4.3 highlights conceptual differences of our approach compared to standard noise initialized baselines. Following the framework established by Chen et al. [10], we define a probability path $p_\lambda(\mathbf{x}|\mathbf{x_0})$ that interpolates between the Dirac measure at the history $p_0(\mathbf{x}) = \delta_{\mathbf{x}_0}$ and the target

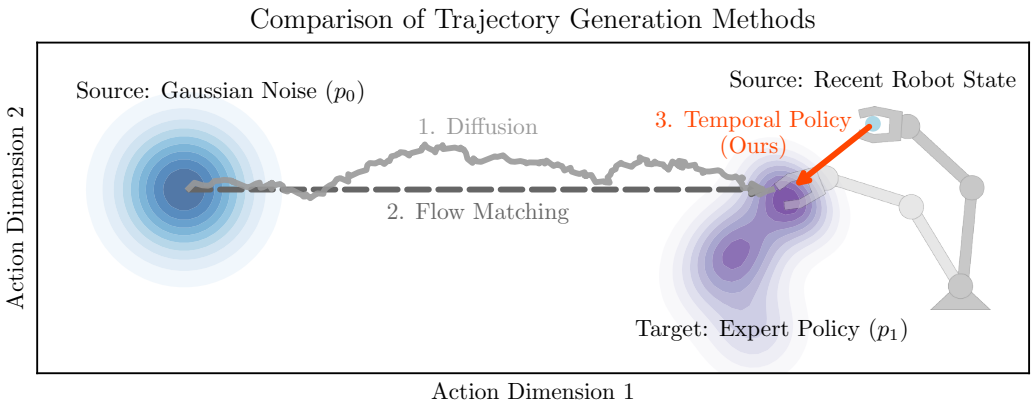


Figure 4.3: Conceptual comparison of generative modeling approaches. (1) **Diffusion Models** generate stochastic, curved paths from Gaussian noise (p_0). (2) **Standard Flow Matching** uses straight paths but still initializes from independent noise. (3) Our **Temporal Policy** initializes the generative process directly from the robot's history, reducing the transport distance.

distribution $p_1(\mathbf{x})$. To avoid confusion with physical time t, we denote the interpolation time variable as $\lambda \in [0, 1]$.

We construct a specific interpolant $\mathbf{x}_\lambda$ that mixes the source, target and a standard Wiener process $\mathbf{w}_\lambda \sim N(\mathbf{0}, \lambda\mathbf{I})$ over the interpolation time λ:

$$\mathbf{x}_\lambda = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1 + \varepsilon(1-\lambda)\mathbf{w}_\lambda. \tag{4.2}$$

The term $\varepsilon(1-\lambda)$ represents the noise schedule γ_λ where $\varepsilon \geq 0$ is a scalar hyperparameter controlling the noise scale. We choose a linear interpolation scheme between $\mathbf{x}_0$ and $\mathbf{x}_1$ to encourage straight vector fields for fast sampling. Unlike standard Stochastic Interpolants which use a static Gaussian variable (Eq. 3.23), this formulation embeds a non-stationary Wiener process $\mathbf{w}_\lambda$ directly into the interpolant. As detailed in Section 4.3.1, this ensures the regression target remains bounded and learnable at the initialization point ($\lambda = 0$).

Furthermore, this formulation enforces strict boundary conditions that anchor the generative trajectory directly to the physical data:

- **Start ($\lambda = 0$):** The noise variance is zero ($\mathbf{w}_0 = \mathbf{0}$), which forces the trajectory to begin exactly at the history $\mathbf{x}_0$.

- **End ($\lambda = 1$):** The multiplier $(1 - \lambda)$ becomes zero. This removes all noise and forces the trajectory to end exactly at the target action $\mathbf{x}_1$.

4.3.1 Derivation of the Vector Field Target

Having defined the stochastic interpolant $\mathbf{x}_\lambda$, we now formulate the regression objective required to train the neural network b_θ. The network's role is to approximate the drift vector field that transports samples along this probability path. To achieve this, we first derive the ground-truth target field, $u_\lambda(\mathbf{x}, \mathbf{x}_0)$.

Because the interpolant contains a non-differentiable Wiener process, we apply Itô's calculus to derive its governing stochastic differential equation:

$$\mathrm{d}\mathbf{x}_\lambda = \mathrm{d}\left((1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1\right) + \mathrm{d}\left(\gamma_\lambda \mathbf{w}_\lambda\right), \tag{4.3}$$

where $\gamma_\lambda = \varepsilon(1-\lambda)$. Itô's rule decomposes the stochastic term into separate drift and diffusion terms:

$$\mathrm{d}(\gamma_\lambda \mathbf{w}_\lambda) = \underbrace{\dot{\gamma}_\lambda \mathbf{w}_\lambda \mathrm{d}\lambda}_{\text{Drift}} + \underbrace{\gamma_\lambda \mathrm{d}\mathbf{w}_\lambda}_{\text{Diffusion}}. \tag{4.4}$$

Substituting this back into Equation 4.3 and collecting terms, the full differential is:

$$\mathrm{d}\mathbf{x}_\lambda = \underbrace{\left[(\mathbf{x}_1 - \mathbf{x}_0) + \dot{\gamma}_\lambda \mathbf{w}_\lambda\right]}_{\text{Trajectory Drift } r_\lambda} \mathrm{d}\lambda + \gamma_\lambda \mathrm{d}\mathbf{w}_\lambda. \tag{4.5}$$

For our specific linear schedule $\gamma_\lambda = \varepsilon(1-\lambda)$, the exact trajectory drift r_λ for a single sample path simplifies to:

$$r_\lambda = (\mathbf{x}_1 - \mathbf{x}_0) - \varepsilon \mathbf{w}_\lambda. \tag{4.6}$$

Because the neural network b_θ operates during inference without access to the future target state $\mathbf{x}_1$ or the specific continuous noise realization $\mathbf{w}_\lambda$, it cannot predict the exact trajectory drift r_λ directly. Instead, it is trained to approximate the ground-truth target vector field, $u_\lambda(\mathbf{x}, \mathbf{x}_0)$. Following the stochastic interpolant framework, this optimal target field is formally defined as the conditional expectation of the trajectory drift over all possible target states and noise paths that cross the current coordinate:

$$u_\lambda(\mathbf{x}, \mathbf{x}_0) = \mathbb{E}_{\mathbf{x}_1, \mathbf{w}_\lambda}\left[r_\lambda \mid \mathbf{x}_\lambda = \mathbf{x}, \mathbf{x}_0\right]. \tag{4.7}$$

To enable simulation-free training, we leverage the marginal equivalence of the Wiener process. At any fixed interpolation time λ, the marginal distribution of the noise path evaluates to $\mathbf{w}_\lambda \sim \mathcal{N}(\mathbf{0}, \lambda\mathbf{I})$. We can therefore reparameterize the noise as $\mathbf{w}_\lambda \overset{d}{=} \sqrt{\lambda}\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. By minimizing the Mean Squared Error over the empirical dataset and standard Gaussian samples, the network b_θ learns to approximate $u_\lambda(\mathbf{x}, \mathbf{x}_0)$ using the following regression objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\lambda,(\mathbf{x}_0,\mathbf{x}_1),\mathbf{z}}\left[\left\| b_\theta(\mathbf{x}_\lambda, \mathbf{x}_0, \lambda, \mathbf{c}) - \left(\mathbf{x}_1 - \mathbf{x}_0 - \varepsilon\sqrt{\lambda}\mathbf{z}\right)\right\|_2^2\right], \tag{4.8}$$

with $\lambda \sim \mathcal{U}(0,1)$, $(\mathbf{x}_0, \mathbf{x}_1) \sim q(p_0, p_1)$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The training procedure for learning this drift is detailed in Algorithm 1.

Algorithm 1 Temporal Policy Training

1: **Input:** Robot state-action dataset $\mathcal{D} = \{s_t, a_t\}$, Horizon H, Shift d, Noise scale ε
2: **repeat**
3: Sample $(\mathbf{x}_0, \mathbf{x}_1)$ where $\mathbf{x}_0 = s_{t-H+1:t}$ and $\mathbf{x}_1 = s_{t-H+1+d:t+d}$
4: Sample interpolation time $\lambda \sim \mathcal{U}(0,1)$ and noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Construct interpolant: $\mathbf{x}_\lambda = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1 + \varepsilon(1-\lambda)\sqrt{\lambda}\mathbf{z}$
6: Compute trajectory drift: $r_\lambda = (\mathbf{x}_1 - \mathbf{x}_0) - \varepsilon\sqrt{\lambda}\mathbf{z}$
7: Update θ by minimizing
$\mathcal{L}(\theta) = \|b_\theta(\mathbf{x}_\lambda, \mathbf{x}_0, \lambda, c) - r_\lambda\|^2$
8: **until** converged

This derivation highlights the benefit of formulating the interpolant using a continuous Wiener process. If we were to define the interpolant using a static Gaussian variable $\mathbf{z}$ to yield an identical marginal distribution, such that $\mathbf{x}_\lambda = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1 + \varepsilon(1-\lambda)\sqrt{\lambda}\mathbf{z}$, the standard time derivative would require evaluating $\frac{\mathrm{d}\sqrt{\lambda}}{\mathrm{d}\lambda}$. This introduces a $\frac{1}{2\sqrt{\lambda}}$ term, forcing the target vector field to diverge to infinity as $\lambda \to 0$ and creating numerical instability during training. By contrast, the Wiener process formulation absorbs the $\sqrt{\lambda}$ dynamic within the continuous increments $\mathrm{d}\mathbf{w}_\lambda$. Because $\mathbf{w}_\lambda = \sqrt{\lambda}\mathbf{z}$, our regression target for the drift remains strictly bounded, smoothly converging to the finite value $(\mathbf{x}_1 - \mathbf{x}_0)$ as $\lambda \to 0$.

4.3.2 Analysis of the Vector Field Target

We formally justify this design choice by analyzing the expected kinetic energy which bounds the learned marginal vector field (Section 3.9.2). We compare our stochastic interpolant against the standard noise-initialized baseline. Standard deterministic flow matching constructs an independent coupling where $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, resulting in the target vector field $r_\lambda^{indep} = \mathbf{x}_1 - \mathbf{x}_0$. Let the action chunk $\mathbf{x} \in \mathbb{R}^{HD}$ be a flattened sequence over horizon H. The total expected kinetic energy of this baseline is entirely defined by its static transport cost J_{indep}:

$$\begin{aligned}\int_0^1 \mathbb{E}[\|r_\lambda^{indep}\|^2]d\lambda &= \int_0^1 \mathbb{E}[\|\mathbf{x}_1 - \mathbf{x}_0\|^2]d\lambda \\ &= J_{indep}. \end{aligned} \tag{4.9}$$

In contrast, our formulation utilizes a stochastic conditional drift $r_\lambda^{temporal} = \mathbf{x}_1 - \mathbf{x}_0 - \varepsilon\sqrt{\lambda}\mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To evaluate the kinetic energy, we first expand the expected squared norm of this field:

$$\begin{aligned}\mathbb{E}[\|r_\lambda^{temporal}\|^2] &= \mathbb{E}[\|\mathbf{x}_1 - \mathbf{x}_0 - \varepsilon\sqrt{\lambda}\mathbf{z}\|^2] \\ &= \mathbb{E}[\|\mathbf{x}_1 - \mathbf{x}_0\|^2] - 2\varepsilon\sqrt{\lambda}\mathbb{E}[(\mathbf{x}_1 - \mathbf{x}_0)^\top \mathbf{z}] + \varepsilon^2\lambda\mathbb{E}[\|\mathbf{z}\|^2]. \end{aligned} \tag{4.10}$$

Because the standard Gaussian noise $\mathbf{z}$ is independent of the boundary states $\mathbf{x}_0$ and $\mathbf{x}_1$, the cross-term expectation evaluates to zero ($\mathbb{E}[\mathbf{z}] = \mathbf{0}$). Furthermore, the expected squared norm of the noise evaluates to the total dimension of the state-action space ($\mathbb{E}[\|\mathbf{z}\|^2] = HD$). This simplifies the instantaneous energy to:

$$\mathbb{E}[\|r_\lambda^{temporal}\|^2] = J_{temporal} + \varepsilon^2\lambda HD. \tag{4.11}$$

Integrating this expected squared norm over the full trajectory $\lambda \in [0, 1]$ yields:

$$\begin{aligned}\int_0^1 \mathbb{E}[\|r_\lambda^{temporal}\|^2]d\lambda &= \int_0^1 \left(J_{temporal} + \varepsilon^2\lambda HD\right) d\lambda \\ &= J_{temporal} + \frac{1}{2}\varepsilon^2 HD. \end{aligned} \tag{4.12}$$

For our method to provide a structurally simpler regression target, the total energy must remain strictly less than the baseline ($J_{temporal} + \frac{1}{2}\varepsilon^2 HD < J_{indep}$).

To evaluate this, we expand the static transport costs. Assuming the training data is standardized per degree of freedom, the expected squared L_2 norm evaluates to the trace of the covariance matrix, $\mathbb{E}[||\mathbf{x}||_2^2] = HD$. For any source $\mathbf{x}_0$ and target $\mathbf{x}_1$ sharing this marginal norm, the expected transport cost expands to:

$$J = \mathbb{E}[||\mathbf{x}_1 - \mathbf{x}_0||_2^2] = 2HD - 2\mathbb{E}[\mathbf{x}_1^\top \mathbf{x}_0] \tag{4.13}$$

For an independent Gaussian source, the cross-correlation term is zero ($\mathbb{E}[\mathbf{x}_1^\top \mathbf{x}_0] = 0$), yielding $J_{indep} = 2HD$. Conversely, our temporal coupling pairs the recent history $\mathbf{x}_0$ with the immediate future trajectory $\mathbf{x}_1$. Because expert demonstrations exhibit smooth motions, the state vectors remain highly collinear over short gaps, yielding a strong positive cross-correlation ($\mathbb{E}[\mathbf{x}_1^\top \mathbf{x}_0] > 0$). Substituting the expanded static costs into our energy condition reveals the relationship:

$$\begin{aligned} \int_0^1 \mathbb{E}[\|r_\lambda^{temporal}\|^2]d\lambda &< \int_0^1 \mathbb{E}[\|r_\lambda^{indep}\|^2]d\lambda \\ 2HD - 2\mathbb{E}[\mathbf{x}_1^\top \mathbf{x}_0] + \frac{1}{2}\varepsilon^2 HD &< 2HD \\ \frac{1}{4}\varepsilon^2 HD &< \mathbb{E}[\mathbf{x}_1^\top \mathbf{x}_0]. \end{aligned} \tag{4.14}$$

Given the collinearity of physical states, this condition is easily satisfied for reasonable choices of the noise scale ε. Following Jensen's inequality (Section 3.9.2), this limits the complexity of the learned vector field b_θ. Therefore we expect Temporal Policy to produce geometrically straighter and less entangled trajectories compared to noise initialized approaches.

4.4 Network Architecture

The vector field $b_\theta(\mathbf{x}_\lambda, \mathbf{x}_0, \lambda, c)$ is parameterized by a neural network. We employ a U-Net backbone architecture designed for sequence modeling. Unlike standard U-Nets used in image synthesis which operate on 2D spatial dimensions, our model utilizes 1D convolutions operating over the physical time dimension of the action chunk. To incorporate the source state $\mathbf{x}_0$, we concatenated it with the intermediate state $\mathbf{x}_\lambda$ along the feature dimension.

To guide the generation process, the network is conditioned on the interpolation time λ and an observation vector $\mathbf{c}$. The interpolation time $\lambda \in [0, 1]$ is first projected into a high-dimensional feature space using sinusoidal positional embeddings [52], allowing the network to distinguish different stages of the flow. Simultaneously, visual observations are processed using a ResNet encoder. To preserve spatial information crucial for manipulation tasks, we apply a spatial softmax layer [16] to the final feature map, converting the dense convolutional features into a compact set of 2D feature points. These visual embeddings are concatenated with the time embeddings to form the global conditioning vector $\mathbf{c}$. This vector is injected into the U-Net backbone via Feature-wise Linear Modulation (FiLM) layers [38], which perform learned affine transformations on the intermediate feature maps of the network. Dimensions and hyperparameters are detailed in Appendix A.1. While we parameterize the drift vector field b_θ using a standard 1D U-Net, the Temporal Policy formulation is architecture-agnostic, compatible with any backbone that supports the coupled state-action dimensionality.

4.5 Inference: Generating Action Chunks

The Temporal Policy framework allows for flexible action generation strategies. While the model is trained to predict a specific vector field b_θ corresponding to a fixed noise schedule γ_λ, we can adjust the dynamics at inference time without retraining. As established in Section 3.4.3, this adaptation is possible because multiple distinct stochastic processes can yield the exact same marginal density evolution.

Changing the Noise Schedule

By manipulating the Fokker-Planck equation, we can determine exactly how to adjust the drift term to compensate for a new inference noise level. Consider the general SDE formulation of our interpolant with a training noise coefficient γ_λ and drift u_λ:

$$\mathrm{d}\mathbf{x} = u_\lambda \mathrm{d}\lambda + \gamma_\lambda \mathrm{d}\mathbf{w}. \tag{4.15}$$

The evolution of the probability density $p_\lambda(\mathbf{x})$ is governed by the *Fokker-Planck equation*:

$$\frac{\partial p_\lambda}{\partial \lambda} = -\nabla \cdot (u_\lambda p_\lambda) + \frac{1}{2}\gamma_\lambda^2 \Delta p_\lambda. \tag{4.16}$$

Our goal is to construct a new SDE with an arbitrary inference noise schedule g_λ that yields this exact same density evolution. To do this, we rewrite the equation in terms of g_λ by trivially adding and subtracting $\frac{1}{2}g_\lambda^2 \Delta p_\lambda$:

$$\frac{\partial p_\lambda}{\partial \lambda} = -\nabla \cdot (u_\lambda p_\lambda) + \frac{1}{2}g_\lambda^2 \Delta p_\lambda - \frac{1}{2}(g_\lambda^2 - \gamma_\lambda^2)\Delta p_\lambda. \tag{4.17}$$

Using the identity $\Delta p = \nabla \cdot (p \nabla \log p)$, we substitute the final term:

$$\frac{\partial p_\lambda}{\partial \lambda} = -\nabla \cdot (u_\lambda p_\lambda) + \frac{1}{2}g_\lambda^2 \Delta p_\lambda - \frac{1}{2}(g_\lambda^2 - \gamma_\lambda^2)\nabla \cdot (p_\lambda \nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x})) \tag{4.18}$$

Factoring out the divergence operator $(-\nabla\cdot)$ and the density (p_λ) isolates our new modified drift term:

$$\frac{\partial p_\lambda}{\partial \lambda} = -\nabla \cdot \left[\left(u_\lambda + \frac{1}{2}(g_\lambda^2 - \gamma_\lambda^2)\nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}) \right) p_\lambda \right] + \frac{1}{2}g_\lambda^2 \Delta p_\lambda. \tag{4.19}$$

This derivation reveals that we can transition to the new inference schedule g_λ by replacing the original drift with the updated term in the inner parentheses. The resulting inference SDE becomes:

$$\mathrm{d}\mathbf{x} = \underbrace{\left[u_\lambda + \frac{g_\lambda^2 - \gamma_\lambda^2}{2} \nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}|\mathbf{x}_0) \right]}_{\tilde{b}} \mathrm{d}\lambda + g_\lambda \mathrm{d}\mathbf{w} \tag{4.20}$$

where $\tilde{b}$ is the modified drift.

To compute this modified drift in practice, we must evaluate the score function $\nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}|\mathbf{x}_0)$. Following the framework established by Chen et al. [10], the score $\nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}|\mathbf{x}_0)$ can be recovered analytically in closed form directly from the drift u_λ and the interpolant coefficients $\alpha_\lambda = 1 - \lambda$, $\beta_\lambda = \lambda$, and $\gamma_\lambda = \varepsilon(1 - \lambda)$. Substituting our learned network b_θ for the theoretical drift u_λ, the time-dependent score is:

$$\nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}|\mathbf{x}_0) = A_\lambda \left[\beta_\lambda b_\theta - \mathbf{c}_\lambda(\mathbf{x}, \mathbf{x}_0) \right], \tag{4.21}$$

where the scaling factor A_λ and correction term $\mathbf{c}_\lambda$ are determined analytically:

$$A_\lambda = \left[\lambda\gamma_\lambda(\dot{\beta}_\lambda\gamma_\lambda - \beta_\lambda\dot{\gamma}_\lambda)\right]^{-1}, \tag{4.22}$$

$$\mathbf{c}_\lambda(\mathbf{x}, \mathbf{x}_0) = \dot{\beta}_\lambda\mathbf{x} + (\beta_\lambda\dot{\alpha}_\lambda - \dot{\beta}_\lambda\alpha_\lambda)\mathbf{x}_0. \tag{4.23}$$

This result is significant because it grants access to the gradient of the log-density without requiring a separate score-matching training objective. By applying the specific linear coefficients of our interpolant defined in Equation 4.2, these terms simplify substantially, yielding the score:

$$\nabla_{\mathbf{x}} \log p_\lambda(\mathbf{x}|\mathbf{x}_0) = \frac{\lambda b_\theta - (\mathbf{x} - \mathbf{x}_0)}{\varepsilon^2\lambda(1-\lambda)} \tag{4.24}$$

This formulation provides two key capabilities:

1. **Deterministic Sampling ($g_\lambda = 0$):** By setting the inference noise to zero, we recover the Probability Flow ODE. This allows for fast, low-variance sampling using standard solvers (e.g., Euler) which is advantageous for real-time control loops. Crucially, deterministic sampling within the Temporal Policy framework is substantially more physically grounded than standard diffusion flows. Integrating a probability flow from $\mathcal{N}(0, \mathbf{I})$ requires the vector field to traverse vast, non-physical regions of the state space. By anchoring both the start and end points of the integration in valid physical configurations, our ODE solver generates probability paths that remain close to the robot's kinematic manifold throughout the generation process, yielding a highly interpretable path. For our specific interpolant, the deterministic drift $\tilde{b}$ simplifies to:

$$\tilde{b} = \frac{1+\lambda}{2} b_\theta + \frac{1-\lambda}{2\lambda}(\mathbf{x} - \mathbf{x}_0) \tag{4.25}$$

2. **Stochastic Sampling ($g_\lambda > 0$):** Conversely, injecting noise during inference allows the policy to explore multimodal action distributions. Moreover, the stochastic formulation suppresses the buildup of approximation errors over successive integration steps.

The choice between these solvers for physical deployment represents a fundamental trade-off between approximation error and discretization error. As

established in the literature, the SDE is theoretically superior when the number of function evaluations (N) is large because its stochastic error contraction dominates [58]. However, the stochastic Euler-Maruyama method possesses a convergence order of 0.5 (error scales with $\sqrt{\Delta\lambda}$). Because real-time robotic control requires a small N (and therefore a large step size $\Delta\lambda$) to minimize latency, discretization artifacts heavily dominate the system. In this low-N regime, the deterministic Euler method's convergence order of 1.0 (error scales with $\Delta\lambda$) is expected to produce better generated actions.

Numerical challenges arise at the boundaries $\lambda = 0$ and $\lambda = 1$, where the denominators in Eq. 4.24 and Eq. 4.25 are zero. For the initial step $\lambda = 0$, we use the raw learned drift b_θ without the score-dependent correction since the theoretical drift at $\lambda = 0$ is:

$$b = \mathbb{E}[x_1|x_0] - x_0 \tag{4.26}$$

At the target boundary $\lambda = 1$, the PF-ODE drift is well behaved. For stochastic sampling, similar stability can be achieved by selecting an inference noise schedule g_λ that includes a $1-\lambda$ factor. This effectively cancels out the singularity in the denominator of the score (Eq. 4.24). Alternatively, one can also use the unmodified drift b_θ and noise schedule γ_λ after some threshold value of λ in the neighborhood of the target boundary.

Upon recovering the final action chunk $\mathbf{x}_1$ (Algorithm 2), we employ receding horizon control, executing only the first K steps ($K < H$) before querying the network for a new trajectory. This strategy balances long-term planning with responsive execution.

Algorithm 2 Temporal Policy Inference

Input: State history $\mathbf{x}_0$, learned drift b_θ, noise schedule g_λ
Initialize: $\mathbf{x}_{\lambda=0} = \mathbf{x}_0$
for $\lambda = 0$ **to** 1 **step** $\Delta\lambda$ **do**
  Compute score: $\nabla \log p_\lambda = \frac{1}{\varepsilon^2 \lambda(1-\lambda)}[\lambda b_\theta - (\mathbf{x}_\lambda - \mathbf{x}_0)]$
  Compute modified drift: $\tilde{b} = b_\theta + \frac{g_\lambda^2 - \gamma_\lambda^2}{2} \nabla \log p_\lambda$
  if $g_\lambda = 0$ **then**
    // ODE
    Update state: $\mathbf{x}_{\lambda+\Delta\lambda} = \mathbf{x}_\lambda + \tilde{b}\Delta\lambda$
  else
    // SDE
    Sample $z \sim \mathcal{N}(0, \mathbf{I})$
    Update state: $\mathbf{x}_{\lambda+\Delta\lambda} = \mathbf{x}_\lambda + \tilde{b}\Delta\lambda + g_\lambda \sqrt{\Delta\lambda} z$
  end if
end for
Output: Action chunk $\mathbf{x}_{\lambda=1}$

Chapter 5

Experiments

In this chapter, we evaluate the efficacy of the proposed Temporal Policy framework against state-of-the-art baselines. We begin with a comprehensive evaluation in simulation using the Robomimic suite to analyze success rates, inference speed, and trajectory quality. Next, we present ablation studies investigating the effects of sampling steps, noise scale, and limited training data on performance. Finally, we validate Temporal Policy on a real world manipulation task.

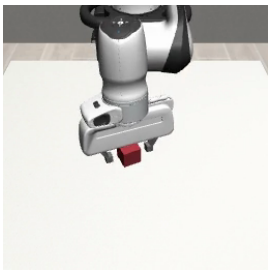
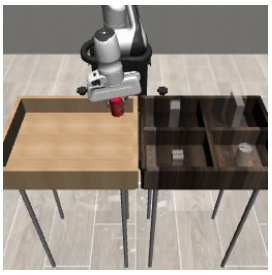
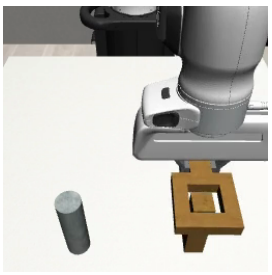
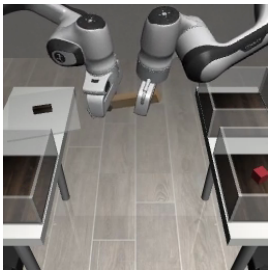
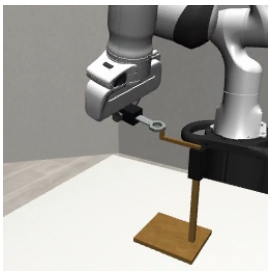
5.1 Simulation Benchmarks

We conduct our primary evaluation on the Robomimic benchmark suite [33], which comprises five manipulation tasks of varying complexity. Detailed task descriptions are provided in Table 5.1.

For each task, we utilize the *Proficient Human* (ph) dataset, consisting of 200 high-quality demonstrations collected by a single expert operator. Additionally, for the Lift, Can, Square, and Transport tasks, we evaluate on the *Multi-Human* (mh) dataset, which contains 300 demonstrations collected from varied operators, introducing significantly higher variance in completion strategy and data quality. For all tasks, the model input includes raw RGB image observations and robot proprioception. Further details are provided in Appendix A.1.

We compare Temporal Policy against two established generative imitation

Table 5.1: Simulation task descriptions from the Robomimic suite.

Image	Task Description
	**Lift** - A fundamental task where the robot must grasp a small cube and lift it to a target height.
	**Can** - The robot must pick a can from a source bin and place it in a specific smaller target bin.
	**Square** - A precise task where the robot must pick up a square nut and place it on a square rod.
	**Transport** - A longer-horizon multi-step bimanual task. One arm retrieves a hammer from a closed container, while the other clears trash from a bin. The hammer is then handed over between arms for final placement.
	**Tool Hang** - A multi-step task where the robot assembles a frame through a precise insertion task and subsequently hangs a hook tool onto the completed structure.

learning baselines:

1. **Diffusion Policy** [11]: A DDPM-based approach that iteratively denoises action sequences starting from an uninformative Gaussian prior.

2. **Conditional Flow Matching (CFM)** [28]: An ODE-based generative model that utilizes a linear interpolation scheme to define a probability flow between a standard Gaussian prior and the data distribution.

To ensure a rigorous comparison, we adopt the baseline training hyperparameters established by Diffusion Policy [11]. Training details are outlined in Appendix A.1. For our formulation, the training noise coefficient is fixed at $\varepsilon = 1.0$. We train each policy on 3 separate seeds, and evaluate the policy every 50 epochs using 50 distinct test seeds, reporting the maximum and average success rates of the final 10 checkpoints. Reported inference times were measured on a consumer NVIDIA RTX 4080 with 16GB of VRAM.

5.1.1 Quantitative Performance and Efficiency

Table 5.2 summarizes the performance of all methods. All three approaches achieve comparable high success rates across the benchmark tasks. Like CFM, our method utilizes only 10 function evaluations (NFE) to generate action sequences. While the baseline Diffusion Policy and CFM models in our evaluation employ 255M parameters, we implement Temporal Policy using a compact 17M parameter architecture. We found that the full 255M parameter led to overfitting for our Temporal Policy. We suspect that the geometrically simpler regression target can lead to memorization for very high parameter networks. However, a more compact network may also be sufficient for other baselines. We trained a scaled-down CFM baseline restricted to the same 17M parameter architecture on the transport task and did not observe degraded performance. For Temporal Policy, the more compact network and low NFE together result in a reduced inference time of **19.1 ms**. Notably, this latency is lower than the typical camera acquisition period (33.3 ms at 30 FPS), ensuring that control actions are computed within a single sensor cycle.

Table 5.2: Performance on simulation tasks. We report *Max/Average* success rates. *Max* is the maximum rate over all training, and *Average* is the mean of the last 10 evaluation checkpoints. Each checkpoint includes 50 task evaluations and all values are averaged across 3 seeds. 'Ph' refers to *proficient-human*, and 'mh' refers to *multi-human*. Bold indicates best results.

Task		Diffusion Policy	CFM	Temporal Policy
Lift	ph	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
	mh	**1.00/1.00**	**1.00**/0.99	**1.00**/0.99
Can	ph	**1.00**/0.97	**1.00/0.98**	**1.00/0.98**
	mh	0.99/0.96	0.99/0.95	**1.00/0.98**
Square	ph	0.98/0.91	0.97/0.91	**0.99/0.96**
	mh	**0.93/0.87**	0.85/0.78	0.91/0.83
Transport	ph	0.97/0.89	**0.98/0.91**	0.97/**0.91**
	mh	0.82/0.73	0.67/0.54	**0.85/0.76**
Tool Hang	ph	**0.87/0.71**	0.79/0.70	0.81/0.69
Parameters		255M	255M	**17M**
NFE		100	**10**	**10**
Inference Time (ms)		615.6	63.5	**19.1**

We investigate the impact of sampling steps on task performance in Figure 5.1 (Left). Standard diffusion fails to produce valid actions below 32 NFEs, requiring at least 64 steps to reach high performance. Conversely, Temporal Policy achieves relatively high success rates with only a single sampling step. The single step performance on all tasks is show in Table 5.3. Here, Temporal Policy either matches or outperforms CFM on all tasks. This fast convergence empirically validates that the data-dependent coupling simplifies the target vector field, allowing low-order ODE solvers to take large step sizes without accumulating fatal discretization errors.

To determine the optimal inference strategy, we evaluate the practical trade-off between computational latency (number of integration steps) and task performance. We compare the deterministic Probability Flow ODE ($\varepsilon = 0$) against an SDE with varying noise scales ($\varepsilon > 0$) on the Transport task. As shown in Figure 5.1 (Right), the dominant effect of increasing the diffusion coefficient is a rightward shift in success rates, demonstrating a penalty on required sampling steps. The deterministic ODE ($\varepsilon = 0$) achieves a relatively

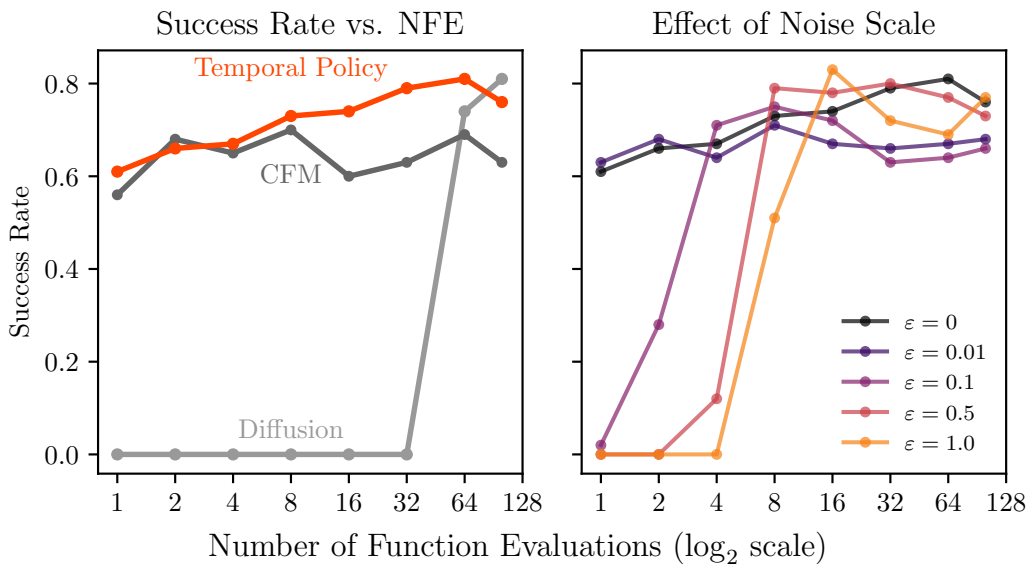


Figure 5.1: Comparison of model performance on the Tool Hang task. (Left) Success rate vs. NFE for Diffusion, CFM, and our Temporal Policy. (Right) An ablation study showing the effect of the inference time SDE coefficient (ε) on success rate.

high success rate in as few as 2 to 4 integration steps. Conversely, introducing stochasticity severely degrades low-step performance and fully stochastic sampling ($\varepsilon = 1.0$) requires at least 8 to 16 steps to reach parity.

This rightward shift reflects the weaker convergence order of the Euler-Maruyama integration required for SDEs compared to the deterministic ODE solver. While prior work in domains such as image synthesis has shown stochastic sampling produces superior generation quality compared to deterministic ODEs [47], our evaluation demonstrates no clear advantage here. Since stochastic sampling does not yield higher success rates to justify its computation overhead, the deterministic ODE is the most efficient choice. Importantly, the adoption of deterministic sampling does not imply a collapse in the model's underlying distributional capacity. A supplemental qualitative analysis of the policy's ability to resolve distinct expert modes is provided in Appendix A.2.

Table 5.3: Single step performance. Success rates are computed over 100 episodes. Bold indicates best results.

Task		Diffusion Policy	CFM	Temporal Policy
Lift	ph	0.00	**1.00**	**1.00**
	mh	0.13	**1.00**	**1.00**
Can	ph	0.00	**0.98**	**0.98**
	mh	0.00	0.94	**0.97**
Square	ph	0.00	0.84	**0.90**
	mh	0.00	0.70	**0.88**
Transport	ph	0.00	0.91	**0.92**
	mh	0.00	0.63	**0.67**
Tool Hang	ph	0.00	0.56	**0.61**

Transport Cost and Flow Linearity

To quantify the geometric efficiency of the generative process, we compute the total transport cost $\mathcal{T}$, defined as the path integral of the generated sample's evolution in the configuration space. For a discrete integration trajectory $\tau = \{x_0, \ldots, x_K\}$ where $x_k \in SE(3)$ represents the sample state at step k, the transport cost is:

$$\mathcal{T}(\tau) = \sum_{k=0}^{K-1} \left(\|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2 + r \cdot \theta(R_k, R_{k+1}) \right) \tag{5.1}$$

Here, $\mathbf{p}$ denotes translation, R denotes the rotation matrix, and $\theta(R_a, R_b)$ represents the geodesic distance between rotations. The characteristic length r is set to 0.1 m to balance units. As detailed in Table 5.4, our Temporal Policy exhibits a near magnitude reduction in transport compared to CFM (1.21 vs. 9.99) and a more than 150× reduction compared to Diffusion Policy (190.02).

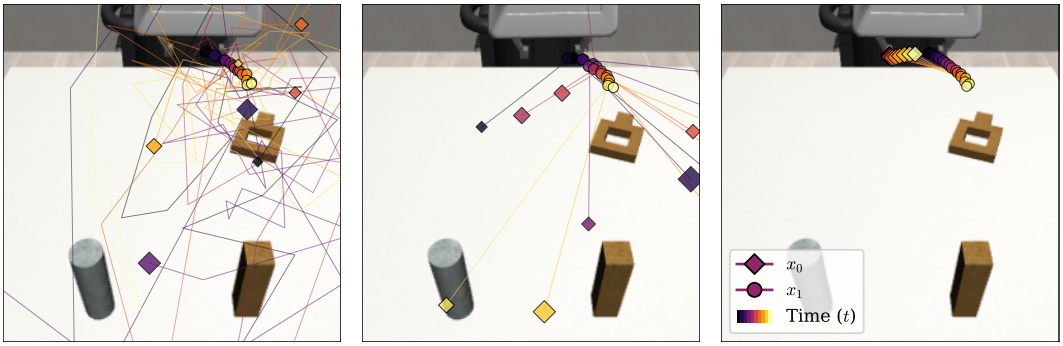
Table 5.4: Transport metrics on the square task. Results are reported as mean ± standard deviation over 100 episodes.

Model	Transport	Straightness
Diffusion Policy	190.02±3.94	20.61±1.67
CFM	9.99±0.82	1.09±0.03
Temporal Policy	**1.21**±0.90	**1.02**±0.03

We attribute this reduction to two factors: (1) the **proximity of initialization**, where starting from the robot's past history places the source distribution significantly closer to the target action than standard Gaussian noise; and (2) the **linearity of the flow**, induced by the linear interpolant objective. To disentangle these effects, we define the straightness ratio $\mathcal{S}$ to quantify the deviation of the learned flow from a straight-line path:

$$\mathcal{S}(\tau) = \frac{\mathcal{T}(\tau)}{\|\mathbf{p}_K - \mathbf{p}_0\|_2 + r \cdot \theta(R_0, R_K)} \tag{5.2}$$

A ratio $\mathcal{S} \approx 1$ implies the model moves particles along straight lines in the configuration manifold. Our method achieves a ratio of **1.02**, significantly outperforming Diffusion Policy ($\mathcal{S} = 20.61$), with only slight improvements over CFM ($\mathcal{S} = 1.09$). This stark contrast in geometric efficiency is visualized in Figure 5.2, which illustrates how noise-initialized models rely on long, and sometimes highly nonlinear paths compared to the direct point-to-distribution transport of Temporal Policy. This confirms that utilizing the robot's history as the flow source not only shortens the generation distance but also induces straight paths, directly enabling the efficient, low-NFE sampling observed in Figure 5.1 and Table 5.3.



(a) Diffusion Policy (b) CFM (c) Temporal Policy

Figure 5.2: Visualization of the generative flow paths on the Square task. (a) Diffusion Policy relies on long, highly nonlinear trajectories to denoise the state from a Gaussian prior. (b) CFM utilizes a linear interpolant, but the random noise initialization still forces the particles to cover significant geometric distance. (c) By anchoring the flow to the observed state history, Temporal Policy produces highly efficient, near-linear trajectories that require minimal transport cost.

Comparison with CFM, which exhibits mostly straight paths, reveals our method's lower transport cost mostly stems from starting closer to the target manifold, rather than from straightening the flow. This proximity provides an inherent numerical advantage. While standard CFM requires high directional precision to navigate the long distance between noise and data without accumulating integration drift, our Temporal Policy exhibits a much higher tolerance to directional error. This likely explains the superior single step performance observed in Table 5.3.

For Temporal Policy, the transport cost is a direct function of the expert demonstrations, where high-velocity motions in the demonstration data will naturally exhibit higher transport costs than slower movements. Notably, since our approach initializes from past history, it allows the vector field to be interpreted as a velocity in physical configuration space.

Ablation Study: Impact of Training Noise Scale

To evaluate the effect of the injected noise during training, we perform an ablation over the parameter ε. The policy success rates are shown in Figure 5.3, evaluated using a deterministic ODE sampler with 10 steps. During training, the standard deviation of the injected noise $\varepsilon(1-\lambda)\sqrt{\lambda}\mathbf{z}$ observes a maximum of 0.38ϵ at $\lambda = \frac{1}{3}$, where the value $\varepsilon = 2.6$ scales the peak standard deviation to approximately 1.0. For small ε (≤ 0.1), performance is low and takes longer to converge to peak performance. Conversely, for $\varepsilon \geq 1.0$, convergence is quick and nearly saturates task performance. The single step performance provides additional insight because evaluating $b_\theta(x_0, 0)$ reduces to pure velocity prediction without noise. The $\varepsilon = 0.1$ model achieves a success rate of 0.34, while the $\varepsilon = 1.0$ model achieves 0.89. This suggests that for low noise scales, models fail to capture a robust velocity field, rather than solely compounding integration errors during sampling. While models with large noise scales achieve high performance, they require the network to expend additional capacity on denoising, therefore, we utilize $\varepsilon = 1.0$ to balance this tradeoff.

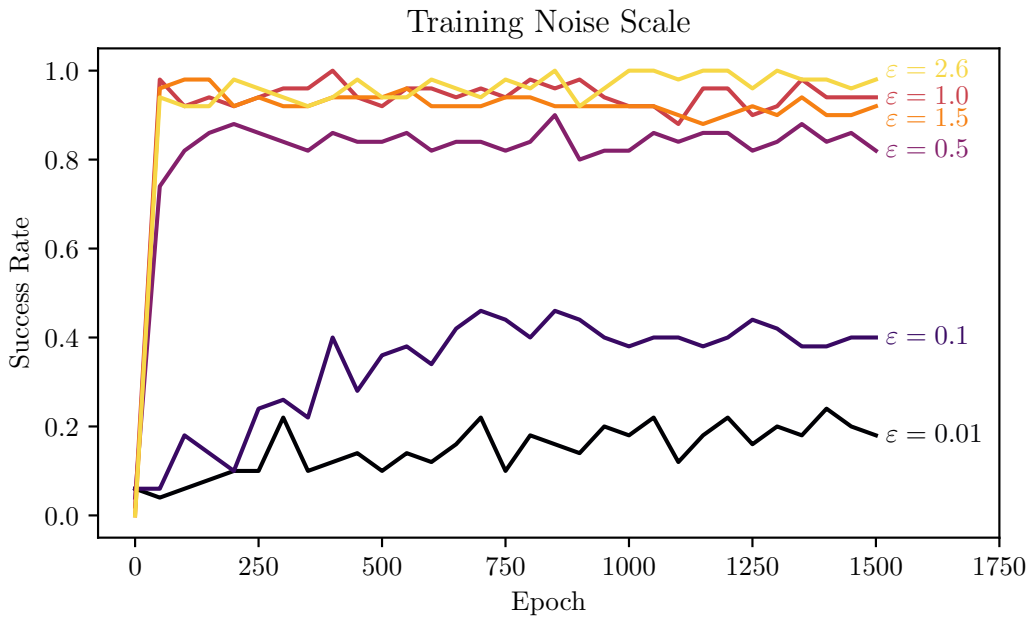


Figure 5.3: Ablation for the training noise scale (ε) on the success rate for the square task.

Dataset Size and Model Performance

Figure 5.4 illustrates success rates as a function of training dataset size. Temporal Policy consistently outperforms Diffusion Policy and CFM across all data regimes. We hypothesize that the Temporal Policy formulation simplifies the learning problem by reducing the discrepancy between the source and target distributions, allowing the model to generalize effectively with fewer demonstrations. This efficiency has important practical implications, as the high cost of data collection remains a significant barrier to the adoption of LfD methods.

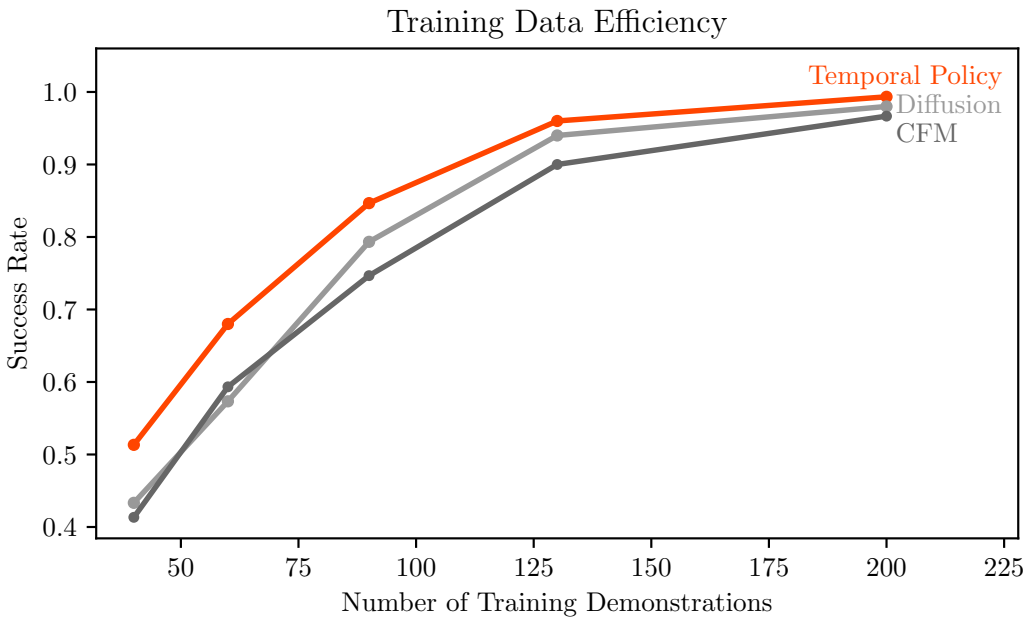


Figure 5.4: Data Efficiency on Square Task.

5.2 Hardware Deployment and Physical Validation

Our experimental validation is conducted on a teleoperation platform comprising two Barrett WAM manipulators configured in a bilateral leader-follower architecture.

The leader device is a 4-DoF Barrett WAM equipped with a custom-built 3-DoF wrist exoskeleton, providing a full 7-DoF interface. This hybrid configuration allows the human operator to control the full pose (position and orientation) of the end-effector. The follower is a standard 7-DoF Barrett WAM manipulator mounted with a BarrettHand (BH8-262) end-effector for grasping tasks.

The system operates on a position-position control loop running at 500 Hz. The bilateral coupling ensures that interaction forces experienced by the follower, such as contact with objects, are reflected back to the leader, providing haptic feedback to the operator. This transparency is critical for collecting high-quality demonstrations with minimal operator mental load. While the arm trajectory is driven by the leader manipulator, the follower's grasping ac-

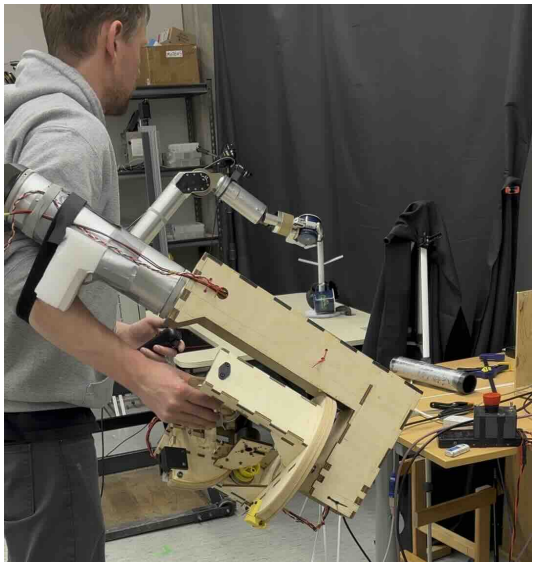


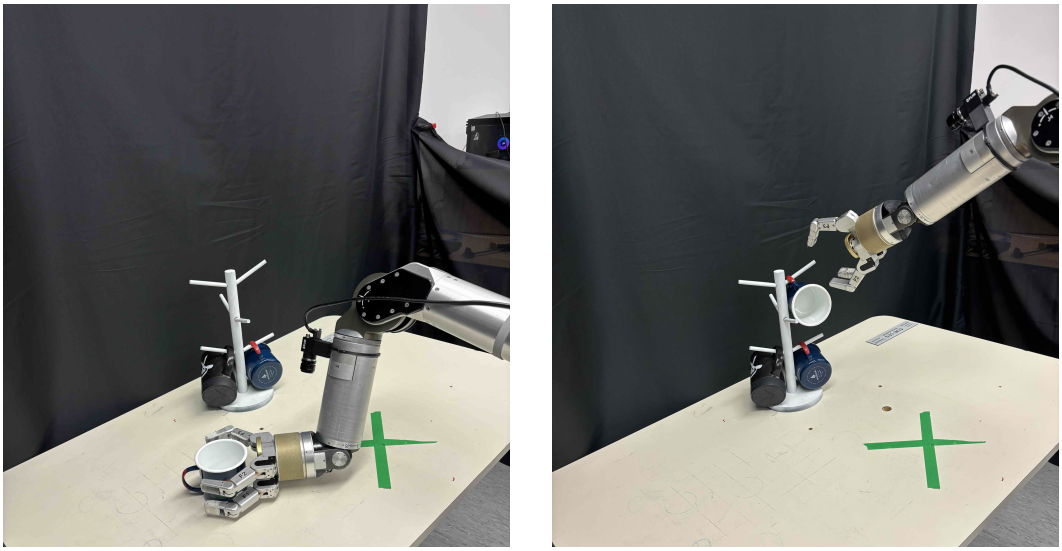
Figure 5.5: Teleoperation setup for data collection. A human operator uses a custom wrist exoskeleton to pilot the follower arm.

tions are commanded with a Bluetooth-enabled PlayStation Navigation controller to send velocity open/close commands to the BarrettHand (BH8-282).

Visual observations are captured using two FLIR Blackfly S USB3 cameras operating at 30 FPS. The camera configuration provides complementary viewpoints including an egocentric camera mounted on the follower robot's wrist to provide a moving, close-up view of object interactions, while an exocentric camera is fixed externally in the workspace to provide a static, global view of the scene.

The image streams are captured asynchronously from the robot state. To construct the training dataset, we subsample the real-time stream and record data at a frequency of 10 Hz. For each recorded timestep t, we pair the current robot state with the most recently captured image frame from each camera. The policy input at each step consists of the tuple $\mathcal{O}_t = \{I_t^{(wrist)}, I_t^{(fixed)}, q_t, g_t\}$, comprising the RGB images from both views, the follower's 7-DoF joint positions q_t, and the current gripper state g_t.

We evaluated the policy on a household-inspired manipulation task, hanging a mug on a tree (Figure 5.6). The robot must first grasp an unconstrained mug such that the handle faces away from the gripper palm, orient the mug,



(a) Start State (b) End State

Figure 5.6: Overview of the mug-hang task. (a) The initial configuration. (b) The successful terminal state.

and align the handle to hang it from a fixed tree peg. This task specifically evaluates the policy's ability to handle unsupported free-space alignment, where slight depth or angular deviations result in missed connections.

To train the policy, we collected 150 expert demonstrations of the hanging task using the bilateral teleoperation interface. The state-action temporal coupling consisted of follower joints positions and gripper velocities, utilizing an overlap of 2 between source and target pairs $(\mathbf{x}_0, \mathbf{x}_1)$. To provide necessary context, the model was additionally conditioned on the gripper joint positions and the RGB images with an observation window of 2. During autonomous physical deployment, the policy operates at a control frequency of 10 Hz. For fast inference, we used deterministic sampling with 10 Euler steps. At each inference step, the model predicts a future action trajectory of 16 steps while only executing 8. These generated waypoints are upsampled to 500 Hz using monotonic cubic interpolation to match the WAM's low-level control loop.

(a) Wrist-mounted camera (b) External camera

Figure 5.7: Camera perspectives during the mug-hanging alignment phase.

5.2.1 Performance on the Physical Mug-Hanging Task

To validate real-world applicability, we deployed Temporal Policy on the physical hardware for 20 evaluation trials of the mug-hanging task. Across the 20 trials with randomized initial mug configurations, Temporal Policy achieved a 50% (10/20) overall success rate. The policy demonstrated high robustness in the initial acquisition phase, successfully grasping the mug in 95% (19/20) episodes. The primary failure mode occurred during the alignment phase, where the end-effector systematically undershot the depth of the tree peg. We hypothesize this is caused by the grasped mug occluding the target peg from the wrist camera as shown in Figure 5.7a.

Despite the alignment failures, the deployment validated the core theoretical claims regarding inference efficiency. Operating on the consumer NVIDIA RTX 4080 hardware, the policy produced an inference latency of 19.62 ms, well below the camera frame rate of 33.3 ms. A breakdown of the compute reveals that the visual encoder contributes 1.86 ms of the total overhead. The majority of the computation is dedicated to the 10-step ODE solver, which averaged 1.78 ms per step for a total of 17.78 ms. To address the depth undershooting in future iterations, better camera placement can be used to limit occlusions during critical phases.

Chapter 6

Conclusion

In this thesis, we presented Temporal Policy, a novel generative framework for robotic imitation learning designed to overcome the structural inefficiencies of standard noise initialized generative models. While previous approaches like Diffusion Policy have demonstrated success in capturing multimodal behaviors, they rely on iterative denoising from uninformative Gaussian noise, a process that is computationally expensive and conceptually disconnected from the robot's physical configuration space.

Our work reframes action generation as a point-to-distribution transport problem via the theory of stochastic interpolants. This formulation enables either fast deterministic sampling or robust stochastic sampling all within a single, unified network that remains stable during training. By initializing the generative flow at the observed state history rather than arbitrary noise, Temporal Policy defines a direct, low-cost mapping from the current physical configuration to the target action distribution.

This theoretical advantage was empirically validated across our experiments. Temporal Policy achieved a near-magnitude reduction in transport cost compared to the noise initialized Conditional Flow Matching and greater than 150× reduction compared to Diffusion Policy. This geometric efficiency is further underscored by a straightness ratio of 1.02, indicating that the learned flow is nearly linear. These properties confer two critical benefits. First, by dramatically shortening the transport distance, our method increases the system's tolerance for directional error. Because the path to the target is brief,

small angular deviations in the predicted velocity vectors have a minimal impact on the final state. Second, the high linearity of the learned flow allows for stable, high-performance sampling with minimal neural network evaluations. Together, these factors explain our method's superior single-step performance.

These geometric gains translate into computational and practical advantages. By grounding the flow in the robot's state space, the learned velocity vectors become inherently interpretable as physical displacements, providing a level of transparency absent in abstract denoising processes. Critically, we demonstrate that Temporal Policy remains capable of capturing complex multimodal distributions, matching state-of-the-art success rates on visuomotor simulation benchmarks. We further validated our approach on physical hardware, demonstrating that the low inference latency (19.62 ms) practically supports the real-time execution speeds required for physical control.

6.1 Limitations

The theoretical efficacy of any imitation learning algorithm is strictly upper-bounded by the quality of the provided expert dataset. Because generative models are optimized to match the empirical distribution of the training data, they will faithfully reproduce sub-optimal, inconsistent, or noisy behaviors present in the demonstrations rather than discovering an optimal policy.

Beyond these paradigm-level constraints, the specific point-to-distribution formulations employed in our methodology introduce structural sensitivities regarding state initialization. The efficacy of the transport relies on the assumption that the current observation is a reliable anchor for the generative process. In scenarios with significant perception noise or latency, the data-dependent initialization may be displaced, potentially biasing the entire generation trajectory. Unlike global diffusion models that generate from pure noise, our method's strong conditioning on the start point means that state estimation errors can propagate more directly into the action sequence.

6.2 Future Directions

Building upon these insights, several avenues exist to extend this work. For instance, using the robot's current velocity to project the initial state forward could initialize the flow closer to the target manifold, further reducing transport cost. Because Temporal Policy modifies the data-dependent coupling, our framework is orthogonal to many concurrent advancements in generative modeling and robot learning. Future iterations can leverage consistency distillation [48] or higher-order numerical solvers to further reduce the required NFE. Expanding the scope of the policy's inputs also presents a significant opportunity for growth. Future research should investigate more capable network architectures and the integration of additional modalities, such as depth, force sensing, and language. Further validating the approach on more diverse and complex datasets would help determine the framework's scaling laws and its capacity to generalize across a wider range of unstructured environments.

Another promising direction lies in exploiting the geometric interpretability of the temporal flow. Physical constraints like collision avoidance or strict kinematic limits can be actively applied to correct the trajectory during the inference process. Finally, while we favored deterministic sampling for its efficiency, future work could explore a hybrid strategy that employs fast ODE solvers for routine motions and switches to theoretically more robust SDE sampling during high-uncertainty or contact-rich task phases.

6.3 Final Remarks

Ultimately, Temporal Policy shifts generative imitation learning from abstract denoising towards a physically grounded transport process. By reconciling the high expressivity of flow-based models with the real-time requirements of robotic control, this framework addresses the core computational and conceptual barriers that have historically hindered the deployment of diffusion-based policies. In doing so, this work facilitates the broader adoption of Learning from Demonstration in physical robotic systems.

References

[1] M. Aburub, C. C. Beltran-Hernandez, T. Kamijo, and M. Hamaya, *Learning Diffusion Policies from Demonstrations For Compliant Contact-rich Manipulation*, Oct. 2024. DOI: 10.48550/arXiv.2410.19235. arXiv: 2410.19235 [cs]. Accessed: May 5, 2025.

[2] M. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Stochastic Interpolants: A Unifying Framework for Flows and Diffusions," *Journal of Machine Learning Research*, vol. 26, no. 209, pp. 1–80, 2025, ISSN: 1533-7928. Accessed: Jun. 29, 2026.

[3] M. S. Albergo, M. Goldstein, N. M. Boffi, R. Ranganath, and E. Vanden-Eijnden, "Stochastic Interpolants with Data-Dependent Couplings," in *Proceedings of the 41st International Conference on Machine Learning*, PMLR, Jul. 2024, pp. 921–937. Accessed: Jun. 29, 2026.

[4] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, May 1982, ISSN: 03044149. DOI: 10.1016/0304-4149(82)90051-5. Accessed: Apr. 18, 2025.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009, ISSN: 09218890. DOI: 10.1016/j.robot.2008.10.024. Accessed: Oct. 3, 2022.

[6] J.-D. Benamou and Y. Brenier, "A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem," *Numerische Mathematik*, vol. 84, no. 3, pp. 375–393, Jan. 2000, ISSN: 0029-599X, 0945-3245. DOI: 10.1007/s002110050002. Accessed: Dec. 23, 2025.

[7] C. M. Bishop, "Mixture density networks," Aston University, Tech. Rep., 1994.

[8] K. Black et al., "$π_0$: A Vision-Language-Action Flow Model for General Robot Control," in *Robotics: Science and Systems*, 2025. arXiv: 2410.24164 [cs]. Accessed: Dec. 5, 2024.

[9] K. B. ten Brinke, K. Minartz, and V. Menkovski, *STFlow: Data-Coupled Flow Matching for Geometric Trajectory Simulation*, May 2026. DOI: 10.48550/arXiv.2505.18647. arXiv: 2505.18647 [cs.LG]. Accessed: Jun. 29, 2026.

[10] Y. Chen, M. Goldstein, M. Hua, M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Probabilistic Forecasting with Stochastic Interpolants and Föllmer Processes," in *Proceedings of the 41st International Conference on Machine Learning*, PMLR, Jul. 2024, pp. 6728–6756. Accessed: Jun. 29, 2026.

[11] C. Chi et al., *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion*, Jun. 2023. arXiv: 2303.04137 [cs]. Accessed: Sep. 20, 2023.

[12] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet, "Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 17 695–17 709. Accessed: Jun. 29, 2026.

[13] J. Dormand and P. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, Mar. 1980, ISSN: 03770427. DOI: 10.1016/0771-050X(80)90013-3. Accessed: Jan. 1, 2026.

[14] B. Efron, "Tweedie's Formula and Selection Bias," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1602–1614, Dec. 2011, ISSN: 0162-1459, 1537-274X. DOI: 10.1198/jasa.2011.tm11181. Accessed: Dec. 23, 2025.

[15] Y. Fang, X. Zhang, H. Cheng, X. Zang, R. Song, and J. Zhao, "Flow Policy: Generalizable Visuomotor Policy Learning via Flow Matching," *IEEE/ASME Transactions on Mechatronics*, vol. 31, no. 1, pp. 140–150, Feb. 2026, ISSN: 1941-014X. DOI: 10.1109/TMECH.2025.3586019. Accessed: Jun. 29, 2026.

[16] C. Finn, Xin Yu Tan, Yan Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden: IEEE, May 2016, pp. 512–519, ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487173. Accessed: Dec. 19, 2025.

[17] P. Florence et al., *Implicit Behavioral Cloning*, Sep. 2021. DOI: 10.48550/arXiv.2109.00137. arXiv: 2109.00137 [cs]. Accessed: Dec. 18, 2025.

[18] D. Gao et al., *VITA: Vision-to-Action Flow Matching Policy*, Mar. 2026. DOI: 10.48550/arXiv.2507.13231. arXiv: 2507.13231 [cs.CV]. Accessed: Jun. 29, 2026.

[19] I. J. Goodfellow et al., *Generative Adversarial Networks*, Jun. 2014. DOI: 10.48550/arXiv.1406.2661. arXiv: 1406.2661 [stat]. Accessed: Dec. 18, 2025.

[20] J. Ho and S. Ermon, *Generative Adversarial Imitation Learning*, Jun. 2016. DOI: 10.48550/arXiv.1606.03476. arXiv: 1606.03476 [cs]. Accessed: Dec. 18, 2025.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851. Accessed: Jun. 29, 2026.

[22] C.-C. Hsu et al., *SPOT: SE(3) Pose Trajectory Diffusion for Object-Centric Manipulation*, Nov. 2024. DOI: 10.48550/arXiv.2411.00965. arXiv: 2411.00965 [cs]. Accessed: May 5, 2025.

[23] S. Jiang, X. Fang, N. Roy, T. Lozano-Pérez, L. P. Kaelbling, and S. Ancha, "Streaming Flow Policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories," in *Proceedings of The 9th Conference on Robot Learning*, PMLR, Oct. 2025, pp. 238–257. Accessed: Jun. 29, 2026.

[24] J. Justin Valentine, D. Miller, F. Haghverd, J. Piwowarczykand, and M. Jagersand, "Teleoperation for Mobile Manipulation in Unstructured Environments," in *Alberta Robotics & Intelligent Systems Expo*, Nov. 2024.

[25] T. Karras, M. Aittala, T. Aila, and S. Laine, *Elucidating the Design Space of Diffusion-Based Generative Models*, Oct. 2022. arXiv: 2206.00364 [cs, stat]. Accessed: Jan. 12, 2024.

[26] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, Dec. 2022. DOI: 10.48550/arXiv.1312.6114. arXiv: 1312.6114 [stat]. Accessed: Dec. 18, 2025.

[27] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang, "A Tutorial on Energy-Based Learning," *MIT Press*, 2006.

[28] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow Matching for Generative Modeling*, Feb. 2023. arXiv: 2210.02747 [cs, stat]. Accessed: Oct. 4, 2024.

[29] W. Liu, J. Wang, Y. Wang, W. Wang, and C. Lu, *ForceMimic: Force-Centric Imitation Learning with Force-Motion Capture System for Contact-Rich Manipulation*, Mar. 2025. DOI: 10.48550/arXiv.2410.07554. arXiv: 2410.07554 [cs]. Accessed: May 5, 2025.

[30] X. Liu, C. Gong, and Q. Liu, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow," in *The Eleventh International Conference on Learning Representations*, Sep. 2022. Accessed: Jun. 29, 2026.

[31] C. Lu, Y. Zhou, J. Chen, C. Li, and J. Zhu, "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps," in *36th Conference on Neural Information Processing Systems*, 2022.

[32] C. Lynch et al., *Learning Latent Plans from Play*, Dec. 2019. DOI: 10.48550/arXiv.1903.01973. arXiv: 1903.01973 [cs]. Accessed: Dec. 18, 2025.

[33] A. Mandlekar et al., "What Matters in Learning from Offline Human Demonstrations for Robot Manipulation," in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 1678–1690. Accessed: Jun. 29, 2026.

[34] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 2901–2910, ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00299. Accessed: Dec. 18, 2025.

[35] A. Noohian, D. Miller, J. Valentine, A. Lynch, and M. Jagersand, "Sensorless Four-Channel Control Architecture on the WAM Teleoperation System," in *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026, in press.

[36] B. Øksendal, *Stochastic Differential Equations* (Universitext). Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ISBN: 978-3-540-04758-2 978-3-642-14394-6. DOI: 10.1007/978-3-642-14394-6. Accessed: Dec. 11, 2024.

[37] M. Oquab et al., *DINOv2: Learning Robust Visual Features without Supervision*, Feb. 2024. DOI: 10.48550/arXiv.2304.07193. arXiv: 2304.07193 [cs]. Accessed: May 5, 2025.

[38] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, ISSN: 2374-3468. DOI: 10.1609/aaai.v32i1.11671. Accessed: Jun. 29, 2026.

[39] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Q. Chen, "Multisample Flow Matching: Straightening Flows with Minibatch Couplings," in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 2023, pp. 28 100–28 127. Accessed: Jun. 29, 2026.

[40] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, *Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation*, Jun. 2024. DOI: 10.48550/arXiv.2405.07503. arXiv: 2405.07503 [cs]. Accessed: Mar. 18, 2026.

[41] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov, *Multimodal Diffusion Transformer: Learning Versatile Behavior from Multimodal Goals*, Jul. 2024. DOI: 10.48550/arXiv.2407.05996. arXiv: 2407.05996 [cs]. Accessed: May 5, 2025.

[42] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*, 1st ed. Cambridge University Press, Apr. 2019, ISBN: 978-1-108-18673-5 978-1-316-51008-7 978-1-316-64946-6. DOI: 10.1017/9781108186735. Accessed: Apr. 22, 2025.

[43] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, Jun. 1999, ISSN: 13646613. DOI: 10.1016/S1364-6613(99)01327-3. Accessed: Dec. 20, 2025.

[44] Y. Shi, V. D. Bortoli, A. Campbell, and A. Doucet, *Diffusion Schrödinger Bridge Matching*, Dec. 2023. DOI: 10.48550/arXiv.2303.16852. arXiv: 2303.16852 [stat]. Accessed: Apr. 24, 2025.

[45] A. Sochopoulos, N. Malkin, N. Tsagkas, J. Moura, M. Gienger, and S. Vijayakumar, "Fast Flow-based Visuomotor Policies via Conditional Optimal Transport Couplings," in *Proceedings of The 9th Conference on Robot Learning*, PMLR, Oct. 2025, pp. 3357–3377. Accessed: Jun. 29, 2026.

[46] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*, Nov. 2015. DOI: 10.48550/arXiv.1503.03585. arXiv: 1503.03585 [cs]. Accessed: Apr. 28, 2025.

[47] Y. Song and P. Dhariwal, *Improved Techniques for Training Consistency Models*, Oct. 2023. DOI: 10.48550/arXiv.2310.14189. arXiv: 2310.14189 [cs]. Accessed: May 28, 2025.

[48] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, vol. 202, Honolulu, Hawaii, USA: JMLR.org, Jul. 2023, pp. 32 211–32 252. Accessed: Jun. 29, 2026.

[49] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-Based Generative Modeling through Stochastic Differential Equations*, Feb. 2021. arXiv: 2011.13456 [cs, stat]. Accessed: Jun. 5, 2023.

[50] A. Tong et al., "Improving and generalizing flow-based generative models with minibatch optimal transport," *Transactions on Machine Learning Research*, p. 34, Mar. 2024, ISSN: 2835-8856. Accessed: Jun. 29, 2026.

[51] A. Y. Tong et al., "Simulation-Free Schrödinger Bridges via Score and Flow Matching," in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, PMLR, Apr. 2024, pp. 1279–1287. Accessed: Jun. 29, 2026.

[52] A. Vaswani et al., *Attention Is All You Need*, Dec. 2017. arXiv: 1706.03762 [cs]. Accessed: Jun. 14, 2023.

[53] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, vol. 23, no. 7, pp. 1661–1674, 2011. DOI: 10.1162/NECO_a_00142.

[54] D. Wang et al., *Equivariant Diffusion Policy*, Oct. 2024. DOI: 10.48550/arXiv.2407.01812. arXiv: 2407.01812 [cs]. Accessed: May 5, 2025.

[55] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li, *GenDP: 3D Semantic Fields for Category-Level Generalizable Diffusion Policy*, Oct. 2024. DOI: 10.48550/arXiv.2410.17488. arXiv: 2410.17488 [cs]. Accessed: May 5, 2025.

[56] Z. Wang et al., *One-Step Diffusion Policy: Fast Visuomotor Policies via Diffusion Distillation*, Oct. 2024. DOI: 10.48550/arXiv.2410.21257. arXiv: 2410.21257 [cs]. Accessed: Mar. 18, 2026.

[57] J. Wen et al., *Diffusion-VLA: Scaling Robot Foundation Models via Unified Diffusion and Autoregression*, Dec. 2024. DOI: 10.48550/arXiv.2412.03293. arXiv: 2412.03293 [cs]. Accessed: May 5, 2025.

[58] Y. Xu, M. Deng, X. Cheng, Y. Tian, Z. Liu, and T. Jaakkola, *Restart Sampling for Improving Generative Processes*, Nov. 2023. DOI: 10.48550/arXiv.2306.14878. arXiv: 2306.14878 [cs]. Accessed: Mar. 28, 2026.

[59] G. Yan et al., "ManiFlow: A General Robot Manipulation Policy via Consistency Flow Training," in *Proceedings of The 9th Conference on Robot Learning*, PMLR, Oct. 2025, pp. 2268–2293. Accessed: Jun. 29, 2026.

[60] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, *3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations*, Sep. 2024. arXiv: 2403.03954 [cs]. Accessed: Nov. 14, 2024.

[61] E. Zhang, Y. Lu, S. Huang, W. Wang, and A. Zhang, *Language Control Diffusion: Efficiently Scaling through Space, Time, and Tasks*, Dec. 2024. DOI: 10.48550/arXiv.2210.15629. arXiv: 2210.15629 [cs]. Accessed: May 5, 2025.

[62] X. Zhang et al., "Trajectory Flow Matching with Applications to Clinical Time Series Modelling," *Advances in Neural Information Processing Systems*, vol. 37, pp. 107 198–107 224, Dec. 2024. DOI: 10.52202/079017-3404. Accessed: Jun. 29, 2026.

[63] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*, Apr. 2023. arXiv: 2304.13705 [cs]. Accessed: Jan. 5, 2024.

[64] 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)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 5738–5746, ISBN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00589. Accessed: Dec. 9, 2023.

Appendix A

A.1 Simulation Experiment Details

All tasks and models use action chunk horizon $H = 16$, an execution horizon of 8, and an observation history of $n_{obs} = 2$. For Temporal Policy, an equivalent shift between source and target is used $d = 14$ such that there is an overlap of 2 steps. Additional task-specific environment parameters, including image resolutions and episode lengths, are summarized in Table A.1.

Table A.1: Robomimic simulation environment parameters. Image resolution is reported as Cameras×W×H. Max steps are reported for proficient-human (ph) and mixed-human (mh) datasets respectively.

Task	Image Resolution	Crop Size	Max Steps (ph, mh)
Lift	2×84×84	76×76	400, 500
Can	2×84×84	76×76	400, 500
Square	2×84×84	76×76	400, 500
Transport	4×84×84	76×76	700
Tool Hang	2×240×240	216×216	700

Representation and Normalization

Actions are defined in the absolute $SE(3)$ pose space of the robot's end-effector. We represent rotations using a 6D representation [64] (the first two columns of the rotation matrix), resulting in a 10-dimensional action vector (3 for position, 6 for rotation, 1 for gripper). All action dimensions are normalized to the $[-1, 1]$ range. For visual observations, raw pixel values are scaled from $[0, 255]$ to the $[0, 1]$ range. During training we use a random crop with

sizes specified in Table A.1 as a form of data augmentation. For inference these crops are centered to ensure deterministic policy behavior.

Network Architecture Specifications

Our Temporal Policy utilizes a U-Net backbone with two down-sampling, and two up-sampling stages with hidden dimensions $[256, 512]$. This is in contrast with the baselines Diffusion Policy and CFM which utilize $[512, 1024, 2048]$. All convolution layers use a kernel size of 5, and the networks employ GroupNorm (`n_groups=8`) and Mish activation functions throughout. Visual features are extracted via a ResNet-18 encoder that yields 32 spatial feature points. The interpolation time is represented using 128-dimensional sinusoidal embeddings.

Training

Models are optimized via AdamW with a base learning rate of 1×10^{-4} and a weight decay of 1×10^{-6}. We employ a cosine learning rate schedule with a linear warm-up for the first 500 steps. To improve inference stability, we maintain an exponential moving average of the model weights. We utilize a dynamic decay schedule that follows an asymptotic ramp-up (with a power of 0.75) until it reaches a maximum decay rate of 0.9999. All models were trained with a batch size of 128 for 1500 epochs on a single NVIDIA A100 GPU. Total training time ranged from approximately 2.5 hours to 53 hours, scaling with the size of the specific task dataset.

A.2 Supplemental Analysis: Multimodal Expressivity

While the primary evaluations in Chapter 5 focus on the efficiency of deterministic ODE sampling, we utilize an SDE sampler here to explicitly visualize the policy's ability to represent and resolve multimodal expert behaviors. In this context, the stochastic noise ($\varepsilon = 1.0$) enables the generation process to explore different high-density regions of the learned action distribution.

(a) Start (b) Counter-clockwise (c) Clockwise

Figure A.1: Visualization of multimodal trajectory generation using a stochastic SDE sampler. From a shared initial configuration (a), the Temporal Policy successfully resolves distinct expert modes, such as counter-clockwise (b) and clockwise (c) maneuvers.

As illustrated in Figure A.1, Temporal Policy can resolve distinct solutions for a single task, specifically, the clockwise and counter-clockwise approaches to grasping a cube. This indicates that the point-to-distribution transport preserves the diversity of the expert demonstrations rather than collapsing to a single mean trajectory. These qualitative results suggest that the learned vector field maintains the capacity for the complex, one-to-many mappings required for multimodal imitation learning.