DIAMOND MAPS: EFFICIENT REWARD ALIGNMENT VIA STOCHASTIC FLOW MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow and diffusion models produce high-quality samples, but adapting them to user preferences or constraints post-training remains costly and brittle, a challenge commonly called reward alignment. We argue that efficient reward alignment should be a property of the generative model itself, not an afterthought, and redesign the model for adaptability. We propose *Diamond Maps*, stochastic flow map models that enable efficient and accurate alignment to arbitrary rewards at inference time. Diamond Maps amortize many simulation steps into a single-step sampler, like flow maps, while preserving the stochasticity required for optimal reward alignment. This design makes search, sequential Monte Carlo, and guidance scalable by enabling efficient and consistent estimation of the value function. Our experiments show that Diamond Maps can be learned efficiently via distillation from GLASS Flows, achieve stronger reward alignment performance, and scale better than existing methods. Our results point toward a practical route to generative models that can be rapidly adapted to arbitrary preferences and constraints.

1 INTRODUCTION

Diffusion models (Song et al., 2020b; Ho et al., 2020) and flow matching (Lipman et al., 2022; Albergo et al., 2023; Liu et al., 2022) are the state-of-the-art generative models for image, video, audio, and molecules. These models are trained to return samples from a *data distribution* $p_{\text{data}}(z)$ over $z \in \mathbb{R}^d$. In many applications, however, we want more than faithful samples: we want to steer them to better satisfy a task objective, e.g. improved text–image alignment. Given a reward function $r(z)$, we seek samples that achieve high reward while staying close to p_{data}. This is commonly referred to as **reward alignment** (Uehara et al., 2024; 2025).

Existing reward alignment methods fall into two broad categories: (1) *Reward fine-tuning* adds an extra training stage to specialize a generator to $r(z)$ (Fan et al., 2023; Wallace et al., 2024; Uehara et al., 2024; Domingo-Enrich et al., 2024). (2) *Inference-time reward alignment* or *guidance* modifies the sampling procedure while leaving the pretrained model unchanged (Uehara et al., 2025; Chung et al., 2022; He et al., 2023; Skreta et al., 2025; Singhal et al., 2025). Both approaches have drawbacks: reward fine-tuning is complex and costly, and must be repeated for each new reward, limiting flexibility. Guidance avoids retraining, but typically relies on approximations that are inaccurate and often increases sampling cost substantially. It remains an open question how to combine the merits of both approaches.

In this work, we choose a different approach, proposing instead to re-design models towards adaptability to arbitrary rewards. The central question we address is:

How to build a generative model that enables fast accurate reward alignment at inference time?

It is known that exact guidance relies on estimation of the **value function** associated with a reward (Uehara et al., 2025). Intuitively, the value function measures how much reward a given state will return in the future on average. However, estimation of the value function is thought to be intractable with flow and diffusion models, as it requires many simulation steps with an SDE or ODE model (Song et al., 2020b; Holderrieth et al., 2025). We challenge this assumption in this work.

To this end, we propose **Diamond Maps**. Diamond Maps leverages recent advances in accelerated sampling methods (Song et al., 2023c; Frans et al., 2024), namely flow maps (Boffi et al., 2025a;b; Geng et al., 2025), which allow us to amortize many inference-steps of flow and diffusion models into a single neural network evaluation. Therefore, in this work we investigate how to **leverage flow maps to accurately and efficiently estimate the value function** and thereby enable exact guidance.

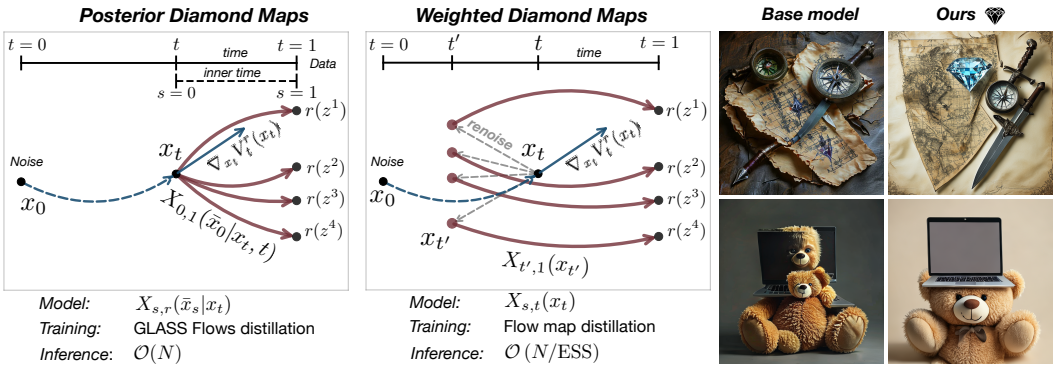

Figure 1: **Overview of Diamond Maps.** Diamond Maps are stochastic flow maps that allow to perform one-step "look-aheads" of a flow trajectory (blue) to potential endpoints at time 1 to evaluate a reward r. This allows for efficient exploration, search, and guidance. We propose 2 Diamond Map Designs. *Posterior Diamond Maps* (left) distill GLASS Flows into a flow map $X_{s,r}(\bar{x}|x_t, t)$ designed to sample exact samples from the posterior. *Weighted Diamond Maps* (middle) allow to use standard flow maps by making them stochastic via a simple renoising procedure with. ESS: effective sample size. Right: Improved image alignment with Diamond Maps. Prompts: "A diamond, a folded treasure map, a compass, and a dagger". "A laptop on top of a teddy bear".

Our key technical observation is that this can be achieved via ***stochastic* flow maps**. While standard flow maps are deterministic, recently **GLASS Flows** (Holderrieth et al., 2025) demonstrated that one can easily obtain stochastic transitions from a pre-trained flow model. This opens up the possibility of *flow map distillation*, which we leverage in this work. In addition, we also present a method to make standard flow maps stochastic at inference time, thereby enabling efficient value function estimation.

Specifically, in this work, we introduce **Diamond Maps**, stochastic flow maps for scalable reward alignment. We propose 2 designs and make the following contributions (see Figure 1):

1. First, we present **Posterior Diamond Maps**, one-step posterior samplers distilled from GLASS Flows. These offer optimal accuracy and efficiency at inference time.
2. Second, we present **Weighted Diamond Maps**, an algorithm to turn standard flow maps into consistent value function estimators. This method allows to use off-the-shelf distilled models.
3. We show that Posterior Diamond Maps - despite only trained to sample from posteriors - are also **one-step samplers of the time-reversal SDE** (Song et al., 2020b) enabling scalable proposals for search/SMC.
4. We empirically demonstrate that Diamond Maps provide fast and high-quality reward alignment.

2 Background

2.1 Flow Matching

We follow the flow matching framework (Lipman et al., 2022; Albergo et al., 2023; Liu et al., 2022), though we remark that everything applies similarly to score-based diffusion models (Song & Ermon, 2019; Song et al., 2021). We denote data points with vectors $z \in \mathbb{R}^d$ and the *data distribution* with p_{data}. Here, $t = 0$ corresponds to noise ($\mathcal{N}(0, I_d)$) and $t = 1$ corresponds to data (p_{data}). To noise data $z \in \mathbb{R}^d$, we use a *Gaussian conditional probability path* $p_t(x_t|z)$:

$$x_t = \alpha_t z + \sigma_t \epsilon \quad (\epsilon \sim \mathcal{N}(0, I_d)) \quad \Leftrightarrow \quad p_t(x_t|z) = \mathcal{N}(x_t; \alpha_t z, \sigma_t^2 I_d), \tag{1}$$

where $\alpha_t, \sigma_t \geq 0$ are *schedulers* with $\alpha_0 = \sigma_1 = 0$ and $\alpha_1 = \sigma_0 = 1$ and α_t (resp. σ_t) strictly monotonically increasing (resp. decreasing). Making $z \sim p_{\text{data}}$ random, this induces a corresponding *marginal probability path* $p_t(x_t) = \mathbb{E}_{z \sim p_{\text{data}}}[p_t(x_t|z)]$ which interpolates Gaussian noise $p_0 = \mathcal{N}(0, I_d)$ and data $p_1 = p_{\text{data}}$. The **flow matching posterior** is defined as

$$p_{1|t}(z|x_t) = \frac{p_t(x_t|z)p_{\text{data}}(z)}{p_t(x_t)} \tag{2}$$

and describes the distribution of clean data z given noisy points x_t. Flow matching models learn the *marginal vector field* $u_t(x_t) = \mathbb{E}_{z\sim p_{1|t}(\cdot|x)}[u_t(x|z)]$ where $u_t(x_t|z)$ is the conditional vector field (see Section A.1 for formula). Simulating an ODE with the marginal vector field from initial Gaussian noise leads to a trajectory whose marginals are p_t:

$$x_0 \sim p_0, \quad \frac{\mathrm{d}}{\mathrm{d}t} x_t = u_t(x_t) \quad \Rightarrow \quad x_t \sim p_t \tag{3}$$

In particular, $X_1 \sim p_{\text{data}}$ returns a sample from the desired distribution. In this work, we also use the alternative parameterization of the vector field $u_t(x)$ given by the *denoiser* D_t, i.e. the expectation of the posterior: $D_t(x) = \mathbb{E}_{z\sim p_{1|t}(\cdot|x)}[z]$.

2.2 Flow Maps

Flow matching requires the step-wise simulation of an ODE, a computationally expensive procedure. To address this, flow maps (Boffi et al., 2025a;b; Geng et al., 2025) can be used that aim to amortize many simulation steps into a single neural network evaluation. Specifically, the **flow map** $X_{t,r}(x_t)$ ($x_t \in \mathbb{R}^d, 0 \leq t \leq r \leq 1$) is defined as the function that maps from time t to time r along ODE trajectories, i.e. if $(x_t)_{0\leq t\leq 1}$ is an ODE trajectory in (3), then $X_{t,r}(x_t) = x_r$. Hence, learning $X_{t,r}$ directly can significantly save inference-time compute. One can then sample from a flow map in a single step by drawing $x_0 \sim \mathcal{N}(0, I_d)$ and setting $x_1 = X_{0,1}(x_0)$, which by construction satisfies $x_1 \sim p_{\text{data}}$.

A diversity of methods have been proposed to learn flow maps either by distillation from existing flow and diffusion models or from scratch via *self-distillation* (Song et al., 2023c; Boffi et al., 2025a;b; Geng et al., 2025). We note that **our method can be used for any distillation method**. For example, the *Lagrangian* loss (Boffi et al., 2025a;b) given by:

$$\mathcal{L}_{\text{Lag}}(\theta) = \mathbb{E}\left[\|\partial_r X^\theta_{t,r}(x_t) - u_r(X_{t,r}(x_t))\|^2\right] \tag{4}$$

where the expectation is over $z \sim p_{\text{data}}, x_t \sim p_t(\cdot|z)$.

3 Value function estimation via Stochastic Flow Maps

In many applications, the distribution p_{data} that a flow matching model aims to sample from serves only as a *prior* distribution. Often, we are given an objective function $r : \mathbb{R}^d \to \mathbb{R}$ called the **reward function** that we aim to maximize. This could be human preferences, constraints, inpainting, and many other tasks. In this setting, the pre-trained generative model allows us to "regularize" our search space. While some works keep the goal vague as "maximizing r while staying on the data manifold", it is commonly formalized as sampling from the **reward-tilted distribution**

$$p^r(z) = \frac{1}{Z} p_{\text{data}}(z) \exp(r(z)). \tag{5}$$

A central object for reward alignment is the **value function** (Uehara et al., 2024; 2025):

$$V^r_t(x_t) = \log \mathbb{E}_{z\sim p_{1|t}(\cdot|x_t)}[\exp(r(z))]. \tag{6}$$

Intuitively, the value function allows to *evaluate* states x_t based on the expected reward in the future. The form $\log \mathbb{E}[\exp(\cdot)]$, is not arbitrary but is inherent to the construction of flow and diffusion models as denoising a Gaussian kernel $p_t(x|z)$: the value function satisfies (see Section A.2) $\exp(V_t(x_t)) \propto p^r_t(x_t)/p_t(x_t)$ where $p^r_t(x_t) = \mathbb{E}_{z\sim p^r}[p_t(x_t|z)]$ is the noised reward-tilted distribution. This expression measures how much more likely a *noisy* state x_t is under p^r compared to p_{data} after noising. The value function $V^r_t(x_t)$ is the core object that needs to be estimated for most reward alignment methods that target to sample from p^r, as we illustrate next.

Guidance. A natural approach to sample from $p^r(z)$ is to estimate the corresponding flow matching model $u^r_t(x)$. This is called **(gradient) guidance**. One can show that u^r_t has the following formula (see Section A.2 for a proof):

$$u^r_t(x) = u_t(x) + b_t \nabla_{x_t} V^r_t(x_t), \tag{7}$$

where $u_t(x)$ is the marginal vector field of the original distribution (non-tilted), and where $b_t = \sigma_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\sigma}_t \sigma_t$. As we know $u_t(x)$, accurate guidance is equivalent to accurate estimation of the gradient $\nabla_{x_t} V^r_t(x_t)$ of the value function.

Sequential Monte Carlo (SMC) and search methods evolve a population of particles $x_t^1, \cdots, x_t^K$ from $t = 0$ to $t = 1$ and assign each of them a weight $w_t^1, \cdots, w_t^K$. If $x_t^i \sim p_t$, then by equation (20) setting $w_t^i = \exp(V_t(x_t^i))$ converts a population of particles from p_t to a population of particles from p_t^r. Therefore, accurate and efficient estimation of the value function $V_t(x_t)$ enables optimal filtering of particles, facilitating search over the support of the prior.

Denoiser approximation. For standard flow and diffusion models the value function is unfortunately *intractable*. The reason for this is that sampling from the posterior $p_{1|t}$ requires simulating a trajectory of an SDE (Song et al., 2020b; Ho et al., 2020) or ODE (Holderrieth et al., 2025), which is too expensive. For this reason, most methods revert to a simple approximation of the posterior $p_{1|t}$ via the denoiser

$$V_t^r(x_t) \approx r(D_t(x_t)) \tag{8}$$

However, it is well-known that this approximation is highly biased, e.g. leading to the so-called *Jensen gap* in the context of linear inverse problems (Chung et al., 2022). This makes many inference-time adaptation methods inaccurate, requires elaborate manual hyperparameter tuning, and limits applicability to simple rewards.

Our approach. In this work, we revisit the value function estimation problem. Instead of reverting to approximations, our goal is to find both (1) consistent (i.e. statistically accurate) and (2) efficient estimators using flow maps. Current sampling with flow maps is *deterministic*, the next point $x_r = X_{t,r}(x_t)$ is completely determined by x_r. However, to truly estimate the value function, we need to take into account *all* possible futures. In other words, we **require stochasticity for exploration**. This motivates our approach:

Goal 1. Estimate the value function $V_t^r(x_t)$ and its gradient $\nabla_{x_t} V_t^r(x_t)$ efficiently at inference-time for arbitrary rewards by constructing stochastic flow maps.

Further, we also construct stochastic flow maps for another reason: SMC and search methods construct search trees by stochastic transitions allowing to explore different branches. We want to construct search trees more efficiently:

Goal 2. Construct stochastic flow maps that enable fast sampling of stochastic Markov transitions for efficient search and SMC.

Next, we present 2 stochastic flow map designs: (1) *Posterior Diamond Maps* and (2) *Weighted Diamond Maps*.

4 Posterior Diamond Maps

In this section, we present *Posterior Diamond Maps*, a stochastic flow map model designed to be efficiently adaptable to a wide range of rewards at inference time. A **Posterior Diamond Map** model $X_{s,r}(\bar{x}_s|x_t, t)$ is a flow map designed to sample from the posterior distribution $p_{1|t}(\cdot|x_t)$ given x_t. Note that x_t is fixed for this flow map. Instead, the times $0 \leq s \leq r \leq 1$ refer to an "inner time axis" that describes the flow evolving from initial Gaussian noise to a sample from $p_{1|t}(\cdot|x_t)$. We sample from this model via

$$\bar{x}_0 \sim \mathcal{N}(0, I_d) \quad \Rightarrow \quad X_{0,1}(\bar{x}_0|x_t, t) \sim p_{1|t}(\cdot|x_t) \tag{9}$$

This enables efficient value function estimation:

Proposition 4.1. *For $\bar{x}_0^k \sim \mathcal{N}(0, I_d)$ and $z^k = X_{0,1}(\bar{x}_0^k|x_t, t)$, the following are consistent estimators of the value function and its gradient:*

$$V_t^r(x_t) \approx \log \frac{1}{N} \sum_{k=1}^{N} \exp(r(z^k)), \quad \nabla_{x_t} V_t(x_t) \approx \sum_{k=1}^{K} \frac{\exp(r(z^k))}{\sum_{j=1}^{K} \exp(r(z^j))} \nabla_{x_t} r(z^k) \tag{10}$$

The proof is simple and follows and the reparameterization trick (Kingma & Welling, 2013) (see Section A.3). In Algorithm 1, we present how to perform guidance with Posterior Diamond Maps and

in Algorithm 2 we present SMC. In the remainder of this section, we describe training of Posterior Diamond Maps in Section 4.1 and sampling in Section 4.2.

4.1 Training Posterior Diamond Maps via Distillation

We next discuss training of Posterior Diamond Maps. Flow maps are distilled ODE trajectories. Therefore, we first express sampling from $p_{1|t}$ via an ODE leveraging the recently introduced GLASS Flows (Holderrieth et al., 2025). Here, the **GLASS velocity field** is given by $\bar{u}_s(\bar{x}_s|x_t, t)$ ($\bar{x}_s, x_t \in \mathbb{R}^d, 0 \leq s, t \leq 1$) for which the sampling procedure is

$$X_0 \sim \mathcal{N}(0, I_d), \quad \frac{\mathrm{d}}{\mathrm{d}s} X_s = \bar{u}_s(\bar{x}_s|x_t, t) \quad \Rightarrow \quad X_1 \sim p_{1|t}(\cdot|X_t = x_t), \tag{11}$$

which generates samples from the posterior $p_{1|t}$. Holderrieth et al. (2025) show how to obtain $\bar{u}_s(\bar{x}_s|x_t, t)$ from a pre-trained flow-matching model $u_t(x_t)$ via simple linear transformations of inputs and outputs. Specifically, define the **sufficient statistic** $S_{s,t}$ as the weighted average and the **time reparameterization** t^* as:

$$S_{s,t}(\bar{x}_s, x_t) = \frac{\alpha_s \sigma_t^2 \bar{x}_s + \alpha_t \sigma_s^2 x_t}{\sigma_t^2 \alpha_s^2 + \alpha_t^2 \sigma_s^2}, \tag{12}$$

$$t^*(s, t) = g^{-1}\left(\frac{\sigma_t^2 \sigma_s^2}{\sigma_t^2 \alpha_s^2 + \alpha_t^2 \sigma_s^2}\right), \quad g(t) = \frac{\sigma_t^2}{\alpha_t^2} \tag{13}$$

(see Section A.6 for analytical formula for g^{-1}). Then the GLASS velocity field is given by

$$\bar{u}_s(\bar{x}_s|x_t, t) = a_{s,t}\bar{x}_s + b_{s,t} D_{t^*}(\alpha_{t^*} S_{s,t}(\bar{x}_s, x_t)) \tag{14}$$

where D_t is the denoiser (see Section 2.1) and $a_{s,t}, b_{s,t} \in \mathbb{R}$ are weight coefficients (see Section A.4 for derivations and explicit formulae). Therefore, the above vector field is a linear reparameterization of a pre-trained model in the denoiser parameterization. The intuition for the above formula is as follows: the sufficient statistics "summarizes" two states $\bar{x}_s, x_t$ into one state by taking their average. This "summary" is effectively a new state at a different time t^* that we can denoise with the "old" denoiser (Holderrieth et al., 2025).

Training. To train a Posterior Diamond Flow Map model $X^\theta_{s,r}(\bar{x}_s|x_t, t)$ with parameters θ, one can now take an existing flow model $u_t(x)$, reparameterize to a GLASS velocity field $\bar{u}_s(\bar{x}_s|x_t, t)$, and then distill it with a distillation method of choice. For example, Lagrangian distillation would minimize the following loss (see equation (4)):

$$\mathbb{E}\left[\|\partial_r X^\theta_{s,r}(\bar{x}_s|x_t, t) - u_r(X_{s,r}(\bar{x}_s|x_t, t)|x_t, t))\|^2\right] \tag{15}$$

where the expectation is over $z \sim p_{\text{data}}$, the times s, r, t are sampled uniformly, and $x_t \sim p_t(\cdot|z), \bar{x}_s \sim p_s(\cdot|z)$.

4.2 One-Step DDPM samples with Diamond Maps

Next, we discuss how one can *iteratively* sample from Posterior Diamond Maps. So far, we only discussed how to obtain terminal samples at time 1 via sampling from the posterior. However, to perform guidance, search, or SMC, we perform *iterative* step-wise sampling, i.e. we need a way to sample a state $x_{t'}$ given x_t for $t' < 1$ that is not a terminal state.

A naive way of sampling from Posterior Diamond Maps would be via **iterative denoising and noising**, i.e. to sample from the posterior and then noise again

$$x_1 \leftarrow X^\theta_{0,1}(\bar{x}_0|x_t, t) \quad (\bar{x}_0 \sim \mathcal{N}(0, I_d)) \qquad \text{(denoise)}$$

$$x_{t'} \leftarrow \alpha_{t'} x_1 + \beta_t \epsilon \quad (\epsilon \sim \mathcal{N}(0, I_d)) \qquad \text{(noise)}$$

For a perfectly trained Posterior Diamond Map, this would be exact, i.e. $x_{t'} \sim p_{t'}$. However, in practice, flow maps have an error that increases with the simulation steps that are distilled in a flow map. Therefore, it seems suboptimal that the above sampling procedure goes from $t \to 1$ and back to t' (e.g. set $t = 0.2, t' = 0.25$). Such an iterative denoising and noising scheme has already been shown to lead to error accumulation for flow maps and consistency models (Sabour et al., 2025). As we demonstrate, a significantly better sampling procedure can be derived.

Intuition. The fundamental idea to improve this scheme relies on the following intuition: the Posterior Diamond Map has an inner time s and an outer time t. By definition, these are two *separate* time axis. However, intuitively $s = 0$ should correspond to outer time t and $s = 1$ to outer time 1. But recall that our goal is to go to a time t' in between t and 1 ($t < t' < 1$)? Our approach is that we can find an inner time s that corresponds to this t' and thereby find a direct one step transition from x_t to $x_{t'}$ in this way by mapping inner states $\bar{x}_s$ to outer states x_t via the sufficient statistic.

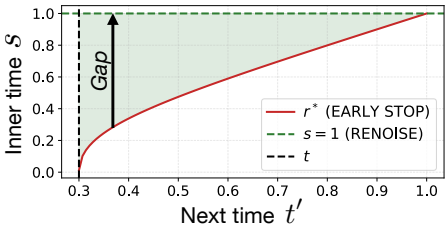

Figure 2: Effective time $0 \to r^*$ amortized in the flow map is significantly smaller for Diamond Early Stop DDPM sampling ("Early stop") than for iterative denoising and noising ("Renoise") leading to reduced error accumulation (see Figure 4). Here, we plot r^* as a function t, t' (see Section A.7).

To make this precise, we define **Diamond Early Stop DDPM sampling** as the sample that we obtain by stopping the Diamond flow "early" and re-transforming with the sufficient statistic $S_{s,t}$, i.e. for $\bar{x}_0 \sim \mathcal{N}(0, I_d)$ set

$$\begin{aligned} \bar{x}_{r^*} &= X^{\theta}_{0,r^*}(\bar{x}_0 | x_t, t) && \text{(flow stopped early)} \\ x_{t'} &= \alpha_{t'} S_{r^*,t}(\bar{x}_{r^*}, x_t) && \text{(map inner to outer state)} \end{aligned}$$

where we set $r^* = g^{-1}(g(t)g(t')/(g(t) - g(t')))$ with g as in (13). The choice of r^* is such that $t^*(r^*, t) = t'$ (see Section A.6). In the following, we call the **DDPM transitions** to be the transition probabilities $p^{\text{DDPM}}_{t'|t}(x_{t'}|x_t)$ from the time-reversal SDE (Ho et al., 2020; Song et al., 2020a) (see Section A.8). The DDPM transitions are a standard choice for reward alignment, in particular as a *proposal distribution* for SMC and search (Li et al., 2025b; Skreta et al., 2025; Singhal et al., 2025).

Proposition 4.2 (Diamond DDPM sampling)**.** *Let a time $t' > t$ and $p^{\text{DDPM}}_{t'|t}$ be the DDPM transition kernel. Then for $x_{t'}$ obtained via Diamond DDPM sampling: $x_{t'} \sim p^{\text{DDPM}}_{t'|t}(\cdot|x_t)$.*

A proof can be found in Section A.8. The above proposition is remarkable, as it shows that the DDPM flow map is contained in the Posterior Diamond Map, although the model was not trained for it. Therefore, Posterior Diamond Map not only accelerates value function estimation but also unlocks more efficient proposal distribution for SMC and search via Diamond DDPM Sampling. As we will demonstrate in Section 7.1, Diamond Early Stop DDPM sampling leads to significantly improved performance compared to iterative denoising and noising.

5 Weighted Diamond Maps

In this section, we propose a different approach to value function estimation via flow maps. We ask: can we estimate the value function by making an existing "standard" flow map $X_{t,t'}$ stochastic at inference time? As we show in this section, this is indeed possible.

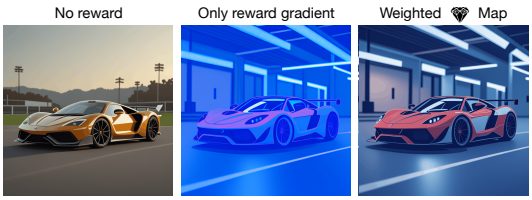

Figure 3: Illustration of Theorem 5.1 with a blueness reward (i.e. reward is maximized for full blue image). Left: Sampling from SANA-Sprint model with no reward. Middle: Naive re-noising and reward gradient with no weighting ((17)). The entire image becomes blue close to collapse. Right: Corrected gradient value function via Theorem 5.1: image remains more realistic due to the added regularization preventing drift off the data manifold.

Let $X_{t,r}$ be a normal flow map. For a given $x_t \in \mathbb{R}^d$, the standard flow map $X_{t,r}(x_t)$ is deterministic. However, a natural idea is to **renoise** x_t back to an earlier time $x_{t'}$ for $t' < t$ and use the flow map to map it to time 1. Specifically, define the **renoising map** for $t' < t$ as

$$x_{t'}(x_t, \epsilon) = \frac{\alpha_{t'}}{\alpha_t} x_t + \left(\sqrt{\sigma^2_{t'} - \frac{\alpha^2_{t'}}{\alpha^2_t} \sigma^2_t} \right) \epsilon$$

The renoising map is constructed such that if $\epsilon \sim \mathcal{N}(0, I_d)$, then $x_{t'}(x_t, \epsilon)$ corresponds to simulation of the diffusion forward process from t to t' (Song et al., 2020b; Sohl-Dickstein et al., 2015). We now use this renoising map to make the flow

map stochastic. For $\epsilon \sim \mathcal{N}(0, I_d)$, we set

$$x_{t'} = x_{t'}(x_t, \epsilon) \qquad \text{(renoise)} \tag{16}$$

$$z(x_t, \epsilon) = X_{t',1}(x_{t'}) \qquad \text{(map to data)} \tag{17}$$

This results in a distribution $z(x_t, \epsilon) \sim q_{1|t}(\cdot|x_t)$ over "clean" data that intuitively should sample "locally" around x_t. However, the resulting distribution is *not* the posterior distribution, i.e. $q_{1|t}(\cdot|x_t) \neq p_{1|t}(\cdot|x_t)$. Therefore, unlike the direct distillation approach in the previous section, samples from this distribution would *not* lead to a correct estimator of the value function. This is illustrated in Figure 3.

However, our fundamental idea is that one correct for it by a *recovery reward*. Specifically, define the *local reward* as

$$r_{\text{local}}(x_t, \epsilon) := r(z(x_t, \epsilon)) - \frac{\|x_t - \alpha_t z(x_t, \epsilon)\|^2}{2\sigma_t^2} \tag{18}$$

The second summand is a recovery reward that is high if x_t is a plausible noisy version of the suggested clean data point $z(x_t, \epsilon_i)$. It equals the NLL $-\log p_t(x|z)$ of the probability path. Refining this intuition, we establish:

Proposition 5.1 (Weighted Diamond Map)**.** *For every $i = 1, \cdots, N$ let $\epsilon_i \sim \mathcal{N}(0, I_d)$. Then the following **weighted Diamond guidance estimator** is a consistent estimator of the gradient of the value function*

$$\nabla_{x_t} V_t^r(x_t) \approx \sum_{i=1}^{N} w_i \left[\nabla_{x_t} r_{\text{local}}(x_t, \epsilon_i) + \delta^i_{\text{score}}\right],$$

where $\delta^i_{\text{score}} = \nabla_{x'_t} \log p_{t'}(x_{t'}) \frac{\alpha_{t'}}{\alpha_t} - \nabla_{x_t} \log p_t(x_t)$ is defined as a score-based correction and the weight coefficients w_i are given by $w_i = \text{softmax}(v_1, \cdots, v_N)$ and

$$v_i = r_{\text{local}}(x_t, \epsilon_i) + \gamma_i + \frac{1}{2}\|\epsilon_i\|^2, \quad \gamma_i \quad = (x_{t'} - x_t)^T \int_0^1 \nabla \log p_{t'}(x_t + u(x_{t'} - x_t))^T \mathrm{d}u$$

We present a proof in Section A.9 and the method in Algorithm 3. The estimator is tractable as we can compute δ_{score} because the score function $\nabla \log p_t(x)$ is a reparameterization of the velocity field $u_t(x)$. As t' is close to t, the γ-coefficients can be approximated efficiently with a trapezoidal quadrature $\gamma_i \approx \frac{1}{2}(\nabla \log p_{t'}(x_t) + \nabla \log p_{t'}(x^i_{t'}))^T (x^i_{t'} - x_t)$. Therefore, the above estimator can be computed in $\mathcal{O}(N)$ function evaluations. We show in Figure 3 that all of these terms are crucial for exact guidance estimation. Finally, we note that the Weighted Diamond Map can also be used to estimate the value function itself (and not only its gradient), see Section A.10.

Posterior Diamond Map vs. Weighted Diamond Map. We finally discuss the advantages and disadvantages of the distilled vs. weighted Diamond Map. The choice is effectively determined by a trade-off between training compute and inference-time compute. The Posterior Diamond Map requires distillation of a GLASS Flows, and thus requires more training time, while the Weighted Diamond Map requires only to distill a standard flow map. While we found that distillation of GLASS Flows to have not significantly more compute costs than standard distillation, model availability might also restrict choice. On the other hand, the Posterior Diamond Map has a sample efficiency $\mathcal{O}(N)$ as the samples are unbiased samples of the posterior $p_{1|t}$, whereas the sample efficiency of the weighted Diamond Map decreases to $\mathcal{O}(N/\text{ESS})$ where $0 \leq \text{ESS} \leq 1$ is the *effective sample size* (Casella & Berger, 2024). Therefore, the weighted Diamond Map may require more compute at inference time. The methods' relative advantages should be weighed depending on application and model availability.

6 Related Work

There has been recent interest in using flow maps and distilled diffusion models for reward alignment. Eyring et al. (2024) leverage a one-step model to optimize the initial noise sample x_0 with gradient descent methods (Ben-Hamu et al., 2024; Wang et al., 2025). Sabour et al. (2025) use flow maps

to obtain efficient one-step look-aheads and replace the denoiser approximation (see (8)) with a flow map approximation that can be corrected for via importance sampling. The LATINO sampler (Spagnoletti et al., 2025) iteratively noises a samples z and then denoises them with a one-step sampler. Our focus here is on estimating the value function, an arguably harder estimation task.

Several recent works have explored learning approximations of the posterior $p_{1|t}$ beyond just learning the mean via the denoiser (De Bortoli et al., 2025; Elata et al., 2024; Chen et al., 2025a). For example, De Bortoli et al. (2025) explore learning the posterior posterior $p_{1|t}$ via scoring rules allowing it to sample discrete-time transitions. Gaussian mixture flow matching (Chen et al., 2025a) approximates the posterior $p_{1|t}$ via Gaussian mixtures. We note that our method is not approximate but exact. We refer to Section C for an extended discussion of related work.

7 Experiments

7.1 Posterior Diamond Maps

We present experiments for Posterior Diamond Maps. Note that we cannot use large pre-trained flow/diffusion models or flow maps for this part and - due to limited compute budget - restrict ourselves to small-scale experiments.

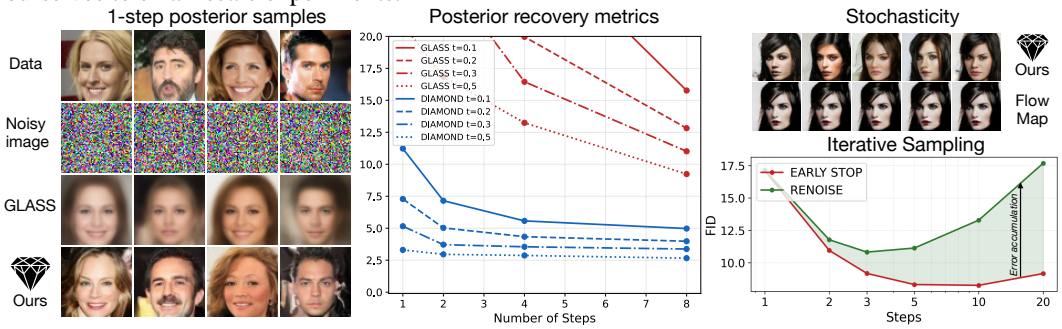

Figure 4: **Training and Sampling Posterior Diamond Maps.** Left: Example of one-step posterior samples from Posterior Diamond Maps. The one-step samplers are faithful and of high quality. Middle: Quantitative results for posterior sampling for various times t. Posterior Diamond Maps outperforms GLASS Flows and therefore have successively distilled them. Top right: In contrast to flow maps that always return the same sample, Posterior Diamond Maps are stochastic and allow to explore future possibilities (sample from posterior). Bottom right: Iterative sampling leads to error accumulation via a iterative denoising and noising scheme (Section 4.2), while improves for Diamond Early Stop DDPM sampling.

We begin by distilling Posterior Diamond Flow models from pre-trained flow models on CIFAR10 and CelebA. To test the ability to sample from the posterior $p_{1|t}$, we follow a similar setup as in (Holderrieth et al., 2025). In Figure 4, we plot qualitative and quantitative results. As one can see, Diamond Maps allows us to sample from the posterior in one step, distilling GLASS Flows effectively into a one-step sampler. We also test Diamond DDPM sampling (see Theorem 4.2) vs. standard iterative noising and renoising (see Section 4.2). Diamond Early Stop DDPM sampling outperforms iterative denoising and noising significantly, making it the method of choice for iterative sampling. We also apply guidance with Posterior Diamond Maps to common linear inverse problems (see Section D.1). For various reward scales and NFEs, we observe that Posterior Diamond Maps has a **better Pareto frontier** (see Figure 6). We find Posterior Diamond Maps is more robust towards high reward scales, caused by stochasticity, compared to guidance with a naive flow map approximation. We also present experiments with SMC by using CLIP (Radford et al., 2021) as a reward model adapting the CelebA model to simple text prompts (see Section D.1).

While at relatively small scale, these results show that Posterior Diamond Maps can be efficiently distilled and that efficient value function estimation leads to improved reward alignment, an observation we will test at scale next.

7.2 Weighted Diamond Maps

Setup high-resolution T2I. We next aim to evaluate the effectiveness of *Weighted Diamond Maps* applied to a pre-trained Flow Map *without* any fine-tuning. We focus on the Text-to-Image setting

Table 1: **Results on GenEval (SANA-Sprint 0.6B)**. Comparison of existing inference-time reward-alignment methods with Weighted Diamond Maps. Weighted Diamond Maps outperform baseline methods, while being more efficient.

Model	Time (s) ↓	Mean ↑	Single↑	Two↑	Counting↑	Colors↑	Position↑	Attribution↑
Flux-dev	23.0	0.68	0.99	0.85	0.74	0.79	0.21	0.48
SD3-Medium (Esser et al., 2024)	4.4	0.70	1.00	0.90	0.72	0.87	0.31	0.66
Qwen-Image (Wu et al., 2025)	35.0	0.87	0.99	0.92	0.89	0.88	0.76	0.77
SANA-Sprint 0.6B Chen et al. (2025b)	0.2	0.70	1.00	0.80	0.64	0.86	0.41	0.51
+ Prompt Optimization (Mañas et al., 2024)	95.0	0.75	0.99	0.91	0.82	0.89	0.36	0.56
+ Best-of-N (Karthik et al., 2023)	15.0	0.79	0.99	0.92	0.72	0.91	0.53	0.65
+ ReNO (Eyring et al., 2024)	30.0	0.81	0.99	0.93	0.74	0.92	0.60	**0.67**
+ Weighted Diamond Maps	<u>10.0</u>	**0.83**	**1.00**	**0.95**	**0.84**	**0.94**	**0.61**	0.65

with human-preference reward alignment. We work with high-resolution images of size 1024×1024, a challenging task. To this end, we leverage SANA-Sprint 0.6B (Chen et al., 2025b).

Rewards. We follow Eyring et al. (2024) and leverage a linear combination of four pre-trained human-preference reward models, ImageReward (Xu et al., 2023), HPSv2 (Wu et al., 2023b), PickScore (Kirstain et al., 2023), and CLIP as our total reward. As a disentangled evaluation from these rewards, we benchmark improvement on GenEval (Ghosh et al., 2023) which better measures genuine improvements in the generated images, and is more robust to reward hacking. We provide the full implementation details in Appendix D.2. We show example trajectories in Figure 5.

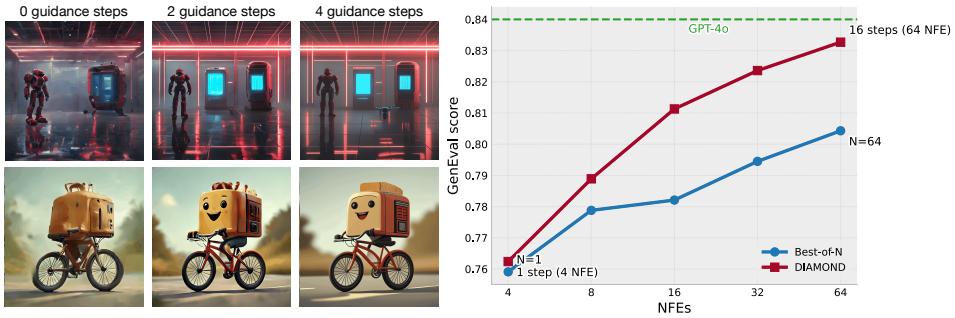

Figure 5: Guidance with Weighted Diamond Maps. Left: Trajectories of guidance (see Algorithm 3) are plotted after N steps (x_1-prediction). Top: Prompt is "a matte red mech suit under dramatic key lights, a sleek electric blue drone hovering nearby, and a cylindrical maintenance pod with glowing panels". Note: The drone only appears after 4 guidance steps. Middle: "A toaster riding a bike". Guidance removed artifacts and increases adherence to prompt. Right: Pareto frontier for guidance with Weighted Diamond Maps vs. Best-of-N. Number of guidance steps is increased with $N = 4$ Monte Carlo samples in Theorem 5.1. NFEs: number of function evaluations. Weighted Diamond Maps have significantly better scaling than Best-of-N.

Comparison against previous methods. We compare Weighted Diamond Maps with $N = 4$ particles against inference-time reward-alignment methods, primarily Best-of-N (Karthik et al., 2023; Ma et al., 2025), Reward-based Noise Optimization (Eyring et al., 2024), and Prompt Optimization (Mañas et al., 2024). We report our primary results in Table 1, where we can observe that not only does **Weighted Diamond Maps outperform competing approaches** but importantly it **does so in more efficient manner**. Note that this is exactly fulfilling the promise of Diamond Maps, *efficient* reward alignment at test-time.

Pareto-frontier. We further study the scaling factor of Diamond Maps across various NFEs, particularly against Best-of-N sampling. We find that the efficacy of Diamond Map Guidance is consistent across the low-NFE regime ($N \in [1, 64]$) as visualized in Figure 5. The performance and scaling is significantly better for Diamond Maps than for Best-of-N. In fact, for $N = 64$ NFEs it reaches a performance close to current state-of-the-art frontier models such as GPT-4o. Overall, this shows that Weighted Diamond Maps is a simple highly effective plug-in estimator for efficient alignment of distilled flow and diffusion models.

REFERENCES

Abbas Abdolmaleki, Bilal Piot, Bobak Shahriari, Jost Tobias Springenberg, Tim Hertweck, Michael Bloesch, Rishabh Joshi, Thomas Lampe, Junhyuk Oh, Nicolas Heess, et al. Learning from negative feedback, or positive feedback or both. In *The Thirteenth International Conference on Learning Representations*.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852, 2023.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025a. URL https://arxiv.org/abs/2406.07507.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025b. URL https://arxiv.org/abs/2505.18825.

George Casella and Roger Berger. *Statistical inference*. Chapman and Hall/CRC, 2024.

Hansheng Chen, Kai Zhang, Hao Tan, Zexiang Xu, Fujun Luan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. Gaussian mixture flow matching models, 2025a. URL https://arxiv.org/abs/2504.05304.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Enze Xie, and Song Han. Sana-sprint: One-step diffusion with continuous-time consistency distillation. *arXiv preprint arXiv:2503.09641*, 2025b.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Valentin De Bortoli, Alexandre Galashov, J Swaroop Guntupalli, Guangyao Zhou, Kevin Murphy, Arthur Gretton, and Arnaud Doucet. Distributional diffusion models with scoring rules. *arXiv preprint arXiv:2502.02483*, 2025.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

Noam Elata, Bahjat Kawar, Tomer Michaeli, and Michael Elad. Nested diffusion processes for anytime image generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5018–5027, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37:125487–125519, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

Ruiqi Feng, Tailin Wu, Chenglei Yu, Wenhao Deng, and Peiyan Hu. On the guidance of flow matching. *arXiv preprint arXiv:2502.02150*, 2025.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

Jiajun He, José Miguel Hernández-Lobato, Yuanqi Du, and Francisco Vargas. Rne: a plug-and-play framework for diffusion density estimation and inference-time control. *arXiv preprint arXiv:2506.05668*, 2025.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Peter Holderrieth, Uriel Singer, Tommi Jaakkola, Ricky TQ Chen, Yaron Lipman, and Brian Karrer. Glass flows: Transition sampling for alignment of flow and diffusion models. *arXiv preprint arXiv:2509.25170*, 2025.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. *arXiv preprint arXiv:2312.02696*, 2023.

Shyamgopal Karthik, Karsten Roth, Massimiliano Mancini, and Zeynep Akata. If at first you don't succeed, try, try again: Faithful diffusion-based text-to-image generation by selection. *arXiv preprint arXiv:2305.13308*, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *International Conference on Machine Learning*, pp. 17842–17857. PMLR, 2023.

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025a.

Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scalia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl, 2025. *URL https://arxiv. org/abs/2505.05470*.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025. URL `https://arxiv.org/abs/2501.09732`.

Konstantin Mark, Leonard Galustian, Maximilian P-P Kovar, and Esther Heid. Feynman-kac-flow: Inference steering of conditional flow matching to an energy-tilted posterior. *arXiv preprint arXiv:2509.01543*, 2025.

Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzal. Improving text-to-image consistency via automatic prompt optimization, 2024. URL `https://arxiv.org/abs/2403.17804`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*, 2025.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alán Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. *arXiv preprint arXiv:2503.02819*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023a.

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pp. 32483–32498. PMLR, 2023b.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023c.

Alessio Spagnoletti, Jean Prost, Andrés Almansa, Nicolas Papadakis, and Marcelo Pereyra. Latinopro: Latent consistency inverse solver with prompt optimization. *arXiv preprint arXiv:2503.12615*, 2025.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review, 2025. URL https://arxiv.org/abs/2501.09685.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Zifan Wang, Alice Harting, Matthieu Barreau, Michael M. Zavlanos, and Karl H. Johansson. Source-guided flow matching, 2025. URL https://arxiv.org/abs/2508.14807.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023a.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023b.

Zihui Wu, Yu Sun, Yifan Chen, Bingliang Zhang, Yisong Yue, and Katherine Bouman. Principled probabilistic imaging using diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 37:118389–118427, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Y Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024.

Po-Hung Yeh, Kuang-Huei Lee, and Jun-Cheng Chen. Training-free diffusion model alignment with sampling demons. *arXiv preprint arXiv:2410.05760*, 2024.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184, 2023.

Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search, 2025. URL https://arxiv.org/abs/2505.23614.

A Mathematical Details and Proofs

A.1 Details for background on flow matching

The conditional vector field is given by (Lipman et al., 2022; 2024):

$$u_t(x|z) = \left(\dot{\alpha} - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x \tag{19}$$

A.2 Derivation of Value Function and Proof of (7)

$$\exp(V_t(x_t)) \propto \frac{p_t^r(x_t)}{p_t(x_t)} \tag{20}$$

Proof. We can derive:

$$p_t^r(x) = \frac{1}{Z}\int p_t(x|z)\exp(r(z))p_{\text{data}}(z)\mathrm{d}z \tag{21}$$

$$\Rightarrow \quad \frac{p_t^r(x)}{p_t(x)} = \frac{1}{Z}\int \exp(r(z))\underbrace{\frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)}}_{\text{posterior}}\mathrm{d}z = \frac{1}{Z}\mathbb{E}[\exp(r(z)|x_t = x] = \frac{1}{Z}\exp(V_t^r(x)) \tag{22}$$

This shows (20). We can express the conditional and marginal vector field via the conditional and marginal score function (Lipman et al., 2024):

$$\begin{aligned} u_t(x|z) &= a_t x + b_t \nabla \log p_t(x|z) \\ u_t(x) &= a_t x + b_t \nabla \log p_t(x) \\ \text{where} \quad a_t &= \frac{\dot{\alpha}_t}{\alpha_t}, b_t = \beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t \end{aligned}$$

As this holds for any data distribution, it also holds for the reward-tilted distribution. Therefore, the tilted marginal vector field u_t^r is given by

$$u_t^r(x) = a_t x + b_t \nabla \log p_t^r(x) = u_t(x) + b_t \nabla \log \frac{p_t^r(x)}{p_t(x)} \tag{23}$$

Taking the gradient of the $\log$ and inserting it into (23), we obtain:

$$u_t^r(x) = u_t(x) + b_t \nabla_{x_t} V_t^r(x_t)$$

This finishes the proof. □

A.3 Proof of Theorem 4.1

Proof. Recall that the value function is given by

$$V_t^r(x_t) = \log \mathbb{E}_{z\sim p_{1|t}(\cdot|x_t)}[\exp(r(z))] \tag{24}$$

Therefore, for $z^1, \cdots, z^N \sim p_{1|t}(\cdot|x_t)$:

$$V_t^r(x_t) \approx \log \frac{1}{N}\sum_{i=1}^{N}\exp(r(z^i)) \tag{25}$$

is a consistent estimator of the value function. Setting $z^i = X_{0,1}(\bar{x}_0|x_t, t)$ proves that (10) is a consistent estimator.

Further, by differentiating (10) by x_t we obtain:

$$\nabla_{x_t} V_t^r(x_t) \approx \frac{1}{\frac{1}{N}\sum_{i=1}^{N}\exp(r(z^i))} \nabla_{x_t} \frac{1}{N}\sum_{i=1}^{N}\exp(r(z^i)) \tag{26}$$

$$= \frac{1}{\sum_{i=1}^{N}\exp(r(z^i))} \sum_{i=1}^{N} \nabla_{x_t} \exp(r(z^i)) \tag{27}$$

$$= \frac{1}{\sum_{i=1}^{N}\exp(r(z^i))} \sum_{i=1}^{N} \exp(r(z^i)) \nabla_{x_t} r(z^i) \tag{28}$$

$$= \sum_{i=1}^{N} \frac{\exp(r(z^i))}{\sum_{i=1}^{N}\exp(r(z^i))} \nabla_{x_t} r(z^i) \tag{29}$$

This shows (10). □

A.4 GLASS VELOCITY FIELD FOR POSTERIOR

The derivation follows (Holderrieth et al., 2025) for the special case of transition being equal to the posterior $p_{1|t}$. First, we set the following parameters in (Holderrieth et al., 2025, Theorem 1)

$$t' = 1, \rho = 0, \bar{\alpha}_s = \alpha_s, \bar{\sigma}_s = \sigma_s$$

and obtain - using their notation

$$\bar{\gamma} = 0, \mu(s) = \begin{pmatrix} \alpha_t \\ \bar{\alpha}_s \end{pmatrix}, \Sigma(s) = \begin{pmatrix} \sigma_t^2 & 0 \\ 0 & \bar{\sigma}_s^2 \end{pmatrix}$$
$$w_1(s) = \dot{\sigma}_s/\sigma_s, w_2(s) = \dot{\alpha}_s - \alpha_s\dot{\sigma}_s/\sigma_s, w_3(s) = 0$$

The sufficient statistics $S_{s,t}(\bar{x}_s, x_t)$ is then given by

$$S_{s,t}(\bar{x}_s, x_t) \tag{30}$$

$$= \frac{\mu^T\Sigma^{-1}[x_t, \bar{x}_s]}{\mu^T\Sigma^{-1}\mu} \tag{31}$$

$$= \frac{\frac{\alpha_s}{\sigma_s^2}\bar{x}_s + \frac{\alpha_t}{\sigma_t^2}x_t}{\frac{\alpha_s^2}{\sigma_s^2} + \frac{\alpha_t^2}{\sigma_t^2}} \tag{32}$$

$$= \frac{\alpha_s\sigma_t^2\bar{x}_s + \alpha_t\sigma_s^2 x_t}{\sigma_t^2\alpha_s^2 + \alpha_t^2\sigma_s^2}, \tag{33}$$

and the **time reparameterization** as:

$$t^*(s,t) = g^{-1}\left(\frac{1}{\mu^T\Sigma^{-1}\mu}\right) = g^{-1}\left(\frac{1}{\frac{\alpha_s^2}{\sigma_s^2} + \frac{\alpha_t^2}{\sigma_t^2}}\right) = g^{-1}\left(\frac{\sigma_t^2\sigma_s^2}{\sigma_t^2\alpha_s^2 + \alpha_t^2\sigma_s^2}\right), \quad g(t) = \frac{\sigma_t^2}{\alpha_t^2} \tag{34}$$

Therefore,

$$D_{\mu(s),\Sigma(s)}(x_t, \bar{x}_s) = D_{t^*}(\alpha_{t^*}S_{s,t}(\bar{x}_s, x_t))$$

Hence, for these choices (Holderrieth et al., 2025, Equation 14) implies that

$$\bar{u}_s(\bar{x}_s|x_t, t) = w_1(s)\bar{x}_s + w_2(s)D_{t^*}(\alpha_{t^*}S_{s,t}(\bar{x}_s, x_t))$$

which coincides with (14) after setting $a_{s,t} = w_1(s) = \dot{\sigma}_s/\sigma_s$ and $b_{s,t} = w_2(s) = \dot{\alpha}_s - \alpha_s\dot{\sigma}_s/\sigma_s$.

A.5 Training Posterior Diamond Maps from scratch

In the same way as for distillation from GLASS Flows, any self-distillation loss could be used for training Posterior Diamond Maps from scratch. To illustrate this, let $X^\theta_{s,r}(\bar{x}_s|x_t,t) = \bar{x}_s + (r-s)v^\theta_{s,r}(x|x_t,t)$ be a flow map parameterized by an average velocity field $v^\theta_{s,t}(x)$. Then we can train it via Lagrangian or Eulerian self-distillation (Boffi et al., 2025b)

$$L_{\text{Lag}}(\theta) = \mathbb{E}\left[\|v^\theta_{s,s}(\bar{x}_s|x_t,t) - u_s(\bar{x}_s|z)\|^2\right] + \mathbb{E}\left[\|\partial_r X^\theta_{s,r}(\bar{x}_s|x_t,t) - v^\theta_{r,r}(X^\theta_{s,r}(\bar{x}_s|x_t,t)|x_t,t)\|^2\right]$$

$$L_{\text{Eul}}(\theta) = \mathbb{E}\left[\|v^\theta_{s,s}(\bar{x}_s|x_t,t) - u_s(\bar{x}_s|z)\|^2\right] + \mathbb{E}\left[\|\partial_s X^\theta_{s,r}(\bar{x}_s|x_t,t) + \nabla_{\bar{x}_s} X^\theta_{s,r}(\bar{x}_s|x_t,t) v_{s,r}(\bar{x}_s|x_t,t)\|^2\right]$$

where the expectation is over $z \sim p_{\text{data}}$, the times s, r, t are sampled uniformly, and $x_t \sim p_t(\cdot|z), \bar{x}_s \sim p_s(\cdot|z)$, and $u_s(\bar{x}_s|z)$ is the conditional vector field (see (19)).

A.6 Analytical formulas for g^{-1}

We derive specific formulas for g, g^{-1} for various schedulers α_t, σ_t. This is stated here for completeness (Holderrieth et al., 2025).

Linear schedule. Let us set

$$\alpha_t = t, \quad \sigma_t = 1 - t$$

Then:

$$g(t) = \frac{(1-t)^2}{t^2} = \left(\frac{1}{t} - 1\right)^2$$
$$g^{-1}(y) = \frac{1}{1+\sqrt{y}}$$

Variance-preserving schedule. Let us set:

$$\sigma_t = \sqrt{1-t}, \quad \alpha_t = \sqrt{t}$$

Then:

$$g(t) = \frac{1-t}{t} = \frac{1}{t} - 1$$
$$g^{-1}(y) = \frac{1}{1+y}$$

Variance-exploding schedule. Let us set:

$$\sigma_t = \sqrt{t}, \quad \alpha_t = 1$$

Then:

$$g(t) = t$$
$$g^{-1}(y) = \frac{1}{y}$$

A.7 Derivation of r^*

Recall the definitions:

$$g(t) = \frac{\sigma_t^2}{\alpha_t^2}$$
$$t^*(s,t) = g^{-1}\left(\frac{\sigma_t^2\sigma_s^2}{\alpha_s^2\sigma_t^2 + \alpha_t^2\sigma_s^2}\right) = g^{-1}\left(\frac{1}{\frac{1}{g(t)} + \frac{1}{g(s)}}\right)$$
$$r^*(t,t') = g^{-1}\left(\frac{g(t)g(t')}{g(t) - g(t')}\right)$$

As described in the main text, r^* was chosen such that $t^*(r^*, t) = t'$. To prove this, we simply plugin the equations:

$$\begin{aligned} t^*(r^*(t,t'),t) &= g^{-1}\left(\frac{1}{\frac{1}{g(t)} + \frac{g(t)-g(t')}{g(t)g(t')}}\right) \\ &= g^{-1}\left(\frac{1}{\frac{g(t)}{g(t)g(t')}}\right) \\ &= g^{-1}\left(g(t')\right) \\ &= t' \end{aligned}$$

A.8 Formal definition of DDPM transitions and proof of Theorem 4.2

The forward process (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020b) is defined via a transition kernel $q_{t|t'}$ in reverse-time given by:

$$q_{t|t'}(x_t|x_{t'}) = \mathcal{N}\left(\frac{\alpha_t}{\alpha_{t'}}x_{t'}; \left(\sigma_t^2 - \frac{\alpha_t^2}{\alpha_{t'}^2}\sigma_{t'}^2\right)I\right), \quad (t < t')$$

The time-reversal SDE or DDPM transitions (Song et al., 2020b; Sohl-Dickstein et al., 2015; Ho et al., 2020) are then given by

$$p_{t'|t}^{\text{DDPM}}(x_{t'}|x_t) = q_{t|t'}(x_t|x_{t'})\frac{p_{t'}(x_{t'})}{p_t(x_t)}$$

Proof of Theorem 4.2. For $\bar{x}_0 \sim \mathcal{N}(0, I_d)$, we define the random variable

$$\tilde{X}_{t'} = \alpha_{t'} S_{r^*,t}(X_{0,r^*}(\bar{x}_0|x_t, t))$$

Our goal is to show that

$$\tilde{p}(\tilde{X}_{t'} = x_{t'}|X_t = x_t) = p_{t'|t}^{\text{DDPM}}(X_{t'} = x_{t'}|X_t = x_t)$$

i.e. that the distribution of $\tilde{X}_{t'}$ given $X_t = x_t$ coincides with the transition kernel $p_{t'|t}^{\text{DDPM}}$ of the DDPM transitions. To show this, it is sufficient to show that the joint distributions coincide

$$\tilde{p}(\tilde{X}_{t'} = x_{t'}, X_t = x_t) = p_{t',t}(X_{t'} = x_{t'}, X_t = x_t) \tag{35}$$

for $x_t \sim p_t$. As the DDPM transition of $q_{t'|t}$, we know that:

$$p_{t,t'}(x_t, x_{t'}) = \int q_{t|t'}(x_t|x_{t'})p_{t'}(x_{t'}|x_1)p_{\text{data}}(x_1)\mathrm{d}x_1$$

$$\text{where} \quad q_{t|t'}(x_t|x_{t'}) = \mathcal{N}\left(\frac{\alpha_t}{\alpha_{t'}}x_{t'}; (\sigma_t^2 - \frac{\alpha_t^2}{\alpha_{t'}^2}\sigma_{t'}^2)I\right), p_{t'}(x_{t'}|x_1) = \mathcal{N}(\alpha_{t'}x_1; \sigma_{t'}^2 I)$$

In other words, we get a sample $(X_t, X_{t'}) \sim p_{t,t'}(x_t, x_{t'})$ via

$$\begin{aligned} X_1 \sim& p_{\text{data}}, \\ X_{t'} &= \alpha_{t'}X_1 + \sigma_{t'}\epsilon_1, \\ X_t &= \frac{\alpha_t}{\alpha_{t'}}X_{t'} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_{t'}^2}\sigma_{t'}^2}\epsilon_2 \quad \epsilon_1, \epsilon_2 \sim \mathcal{N}(0, I_d) \end{aligned}$$

Next, to sample from $\tilde{p}(\tilde{x}_t, x_t)$, we know that we can equivalently do the following procedure: Sample $X_1 \sim p_{\text{data}}$ and then noise X_t and X_{r^*} independently and map X_{r^*} to $X_{t'}$. In other words, we can use the following sampling procedure:

$$X_1 \sim p_{\text{data}}, \tag{36}$$

$$X_{r^*} = \bar{\alpha}_{r^*}X_1 + \bar{\sigma}_{r^*}\tilde{\epsilon}_1 \tag{37}$$

$$X_t = \alpha_t X_1 + \sigma_t \tilde{\epsilon}_2 \tag{38}$$

$$X_{t'} = \alpha_{t'}\frac{\bar{\alpha}_{r^*}\sigma_t^2 X_{r^*} + \alpha_t\bar{\sigma}_{r^*}^2 X_t}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \tag{39}$$

This is because of the definition of the GLASS velocity field (Holderrieth et al., 2025) and GLASS probability path (Holderrieth et al., 2025, Section 4.2.2). We now only have to show that $(X_t, X_{t'})$ sampled via this procedure has the same distribution as sampling them via the SDE/DDPM procedure. However, for that, it is sufficient to show that their distribution conditioned on X_1 is equal. Therefore, let's fix $X_1 = x_1$. Then for the DDPM/SDE case, it holds that

$$(X_t, X_{t'})|X_1 = x_1 \sim \mathcal{N}\left(\begin{pmatrix}\alpha_t x_1 \\ \alpha_{t'} x_1\end{pmatrix}, \begin{pmatrix}\sigma_t^2 & \frac{\alpha_t}{\alpha_{t'}}\sigma_{t'}^2 \\ \frac{\alpha_t}{\alpha_{t'}}\sigma_{t'}^2 & \sigma_{t'}^2\end{pmatrix}\right)$$

Conversely, for the posterior flow case, we can derive the covariance terms conditioned on X_1 using (39) and the fact that X_t, X_{r^*} are independent given X_1:

$$\begin{aligned}
\text{Cov}(X_t, X_{t'}|X_1) &= \alpha_{t'} \frac{\bar{\alpha}_{r^*}\sigma_t^2 \text{Cov}(X_{r^*}, X_t|X_1) + \alpha_t \bar{\sigma}_{r^*}^2 \text{Cov}(X_t, X_t|X_1)}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \\
&= \alpha_{t'} \frac{\alpha_t \bar{\sigma}_{r^*}^2 \text{Cov}(X_t, X_t|X_1)}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \\
&= \alpha_{t'}\alpha_t \frac{\bar{\sigma}_{r^*}^2 \text{Var}(X_t|X_1)}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \\
&= \alpha_{t'}\alpha_t \frac{\bar{\sigma}_{r^*}^2 \sigma_t^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2}
\end{aligned}$$

Finally, by construction

$$\begin{aligned}
\sigma_{t'}^2 &= \alpha_{t'}^2 g(t') = \alpha_{t'}^2 g(t^*(r^*, t)) \\
&= \alpha_{t'}^2 \frac{\sigma_t^2\bar{\sigma}_{r^*}^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} = \frac{\alpha_{t'}}{\alpha_t}\alpha_{t'}\alpha_t \frac{\bar{\sigma}_{r^*}^2\sigma_t^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} = \frac{\alpha_{t'}}{\alpha_t}\text{Cov}(X_t, X_t'|X_1)
\end{aligned}$$

Similarly, it holds that

$$\begin{aligned}
\text{Var}(X_{t'}|X_1) &= \alpha_{t'}^2 \frac{\bar{\alpha}_{r^*}^2\sigma_t^4\bar{\sigma}_{r^*}^2 + \alpha_t^2\bar{\sigma}_{r^*}^4\sigma_t^2}{(\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2)^2} \\
&= \alpha_{t'}^2 \frac{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \frac{\sigma_t^2\bar{\sigma}_{r^*}^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} \\
&= \alpha_{t'}^2 \frac{\sigma_t^2\bar{\sigma}_{r^*}^2}{\bar{\alpha}_{r^*}^2\sigma_t^2 + \alpha_t^2\bar{\sigma}_{r^*}^2} = \alpha_{t'}^2 g(t') = \sigma_{t'}^2
\end{aligned}$$

Therefore, it holds that

$$(X_t, X_{t'})|X_1 = x_1 \sim \mathcal{N}\left(\begin{pmatrix}\alpha_t x_1 \\ \alpha_{t'} x_1\end{pmatrix}, \begin{pmatrix}\sigma_t^2 & \frac{\alpha_t}{\alpha_{t'}}\sigma_{t'}^2 \\ \frac{\alpha_t}{\alpha_{t'}}\sigma_{t'}^2 & \sigma_{t'}^2\end{pmatrix}\right)$$

which coincides with distribution of the memoryless SDE conditioned on X_1. Therefore, they also must coincide if we make X_1 random with p_{data}. This proves that the joints coincide (see (35)). And therefore, also the distributions conditioned on $X_t = x_t$. This finishes the proof. □

A.9 Proof of Theorem 5.1

Proof. Let $X_{t,r}$ be a standard (i.e. deterministic) ground-truth flow map. Recall the definitions

$$\begin{aligned}
q_{t'|t}(x_{t'}|x_t) &= \mathcal{N}\left(\frac{\alpha_{t'}}{\alpha_t}x_t; (\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2)I\right) \\
x_{t'}(x_t, \epsilon) &= \frac{\alpha_{t'}}{\alpha_t}x_t + \left(\sqrt{\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2}\right)\epsilon.
\end{aligned}$$

By the construction of the flow map, if $x_{t'} \sim p_{t'}$, then it also holds that

$$z = X_{t',1}(x_{t'}) \sim p_{\text{data}}.$$

Hence, we can derive:

$$\begin{aligned}
V_t(x_t) &= \log \int \exp(r(z)) \frac{p_t(x_t|z)p_{\text{data}}(z)}{p_t(x_t)} \mathrm{d}z \\
&= \log \int \exp(r(z)) p_t(x_t|z) p_{\text{data}}(z) \mathrm{d}z - \log p_t(x) \\
&= \log \mathbb{E}_{z\sim p_{\text{data}}} \left[\exp(r(z)) p_t(x_t|z)\right] - \log p_t(x_t) \\
&= \log \mathbb{E}_{x_{t'}\sim p_{t'}} \left[\exp(r(X_{t',1}(x_{t'}))) p_t(x_t|X_{t',1}(x_{t'}))\right] - \log p_t(x_t) \\
&= \log \mathbb{E}_{x_{t'}\sim q_{t'|t}(\cdot|x_t)} \left[\exp(r(X_{t',1}(x_{t'}))) \frac{p_t(x_t|X_{t',1}(x_{t'}))p_{t'}(x_{t'})}{q_{t'|t}(x_{t'}|x_t)}\right] - \log p_t(x_t)
\end{aligned}$$

Furthermore, we know that

$$\begin{aligned}
\log p_{t'}(x'_t) &= \log p_{t'}(x_t) + \int_0^1 \frac{\mathrm{d}}{\mathrm{d}u} \log p_{t'}(x_t + u(x_{t'} - x_t)) \mathrm{d}u \\
&= \log p_{t'}(x_t) + \left[\int_0^1 \nabla \log p_{t'}(x_t + u(x_{t'} - x_t))^T \mathrm{d}u\right]^T (x_{t'} - x_t) \\
&=: \log p_{t'}(x_t) + \gamma_{t,t'}(x_t, x_{t'}), \qquad (40)
\end{aligned}$$

where above, $\gamma_{t,t'}(x_t, x_{t'})$ is the second term on the second line. From this expression and the chain rule (with the definition of $x_{t'}(x_t, \epsilon)$, we can derive the following identity that we will later use:

$$\begin{aligned}
\nabla_{x_t} \gamma_{t,t'}(x_t, x_{t'}(x_t, \epsilon)) &= \nabla_{x_t} \log p_{t'}(x'_t(x_t, \epsilon)) - \nabla_{x_t} \log p_{t'}(x_t) \\
&= \nabla_{x_{t'}} \log p_{t'}(x'_t) \frac{\alpha_{t'}}{\alpha_t} - \nabla_{x_t} \log p_{t'}(x_t) \qquad (41)
\end{aligned}$$

In addition, simply exponentiation of (40) yields

$$p_{t'}(x_{t'}) = p_{t'}(x_t) \exp(\gamma_{t,t'}(x_t, x_{t'})).$$

Therefore, we can summarize that

$$V_t^r(x_t) = \log \mathbb{E}_{q_{t'|t}(\cdot|x_t)} \left[\exp(r(z) + \gamma_{t,t'}(x_t, x_{t'})) \frac{p_t(x_t|z)}{q_{t'|t}(x_{t'}|x_t)}\right] + \log p_{t'}(x_t) - \log p_t(x_t) \quad (42)$$

Next, we can reparametize sampling $x_{t'}$ as sampling noise ϵ and applying the renoising operator $x_{t'}(\cdot, \epsilon)$:

$$\begin{aligned}
V_t^r(x_t) = &\underbrace{\log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)} \left[\exp(r(X_{t',1}(x_{t'}(x_t,\epsilon))) + \gamma_{t,t'}(x_t, x_{t'}(x_t,\epsilon))) \frac{p_t(x_t|X_{t',1}(x_{t'}(x_t,\epsilon)))}{q_{t'|t}(x_{t'}(x_t,\epsilon)|x_t)}\right]}_{A} \\
&+ \underbrace{\log p_{t'}(x_t) - \log p_t(x_t)}_{\delta_{\text{log-ll}}}
\end{aligned}$$

The first term A can be estimated via the consistent estimator ($\epsilon_i \sim \mathcal{N}(0, I_d)$:

$$\hat{A} := \log \frac{1}{N} \sum_{i=1}^{N} \underbrace{\exp(r(X_{t',1}(x_{t'}(x_t,\epsilon_i))) + \gamma_{t,t'}(x_t, x_{t'}(x_t,\epsilon_i))) \frac{p_t(x_t|X_{t',1}(x_{t'}(x_t,\epsilon_i)))}{q_{t'|t}(x_{t'}(x_t,\epsilon_i)|x_t)}}_{=:\exp(v_i(x_t,\epsilon))}$$

where $\quad v_i \;:=\; v_i(x_t, \epsilon) = r(X_{t',1}(x_{t'}(x_t, \epsilon_i)))$
$\qquad + \gamma_{t,t'}(x_t, x_{t'}(x_t, \epsilon_i)) + \log p_t(x_t|X_{t',1}(x_{t'}(x_t, \epsilon_i))) - \log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t).$

Taking the gradient results in

$$\nabla_{x_t} \hat{A} = \sum_{i=1}^{N} \text{softmax}(v_1, \cdots, v_N)[i] \nabla_{x_t} v_i(x_t, \epsilon_i)$$

Recalling the definition of $\delta^i_{\text{score}} := \nabla_{x'_t} \log p_{t'}(x_{t'}) \frac{\alpha_{t'}}{\alpha_t} - \nabla_{x_t} \log p_t(x_t)$, we can use (41) to obtain:

$$
\begin{aligned}
&\nabla_{x_t} v_i(x_t, \epsilon_i) \\
=&\nabla_{x_t} r(X_{t',1}(x_{t'}(x_t, \epsilon_i))) + \nabla_{x_t} \gamma_{t,t'}(x_t, x_{t'}(x_t, \epsilon_i)) + \nabla_{x_t} \log p_t(x_t | X_{t',1}(x_{t'}(x_t, \epsilon_i))) \\
&- \nabla_{x_t} \log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t) \\
&= \nabla_{x_t} r(X_{t',1}(x_{t'}(x_t, \epsilon_i))) + \nabla_{x_{t'}} \log p_{t'}(x^i_{t'}) \frac{\alpha_{t'}}{\alpha_t} \\
&- \nabla_{x_t} \log p_{t'}(x_t) + \nabla_{x_t} \log p_t(x_t | X_{t',1}(x_{t'}(x_t, \epsilon_i))) - \nabla_{x_t} \log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t) \\
&= \underbrace{\nabla_{x_t} r(X_{t',1}(x_{t'}(x_t, \epsilon_i))) + \delta^i_{\text{score}} + \nabla_{x_t} \log p_t(x_t | X_{t',1}(x_{t'}(x_t, \epsilon_i))) - \nabla_{x_t} \log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t)}_{=:B^i} \\
&+ \nabla \log p_t(x_t) - \nabla \log p_{t'}(x_t),
\end{aligned}
$$

where the last line telescopes out $\nabla \log p_{t'}(x_t)$ to replace it with $\nabla_{x_t} \log p_t(x_t)$. There are two key intuitions behind this step: (1) the δ^i_{score} term now evaluates scores the same distributions as its arguments, and (2) the right last two terms cancel under the softmax with the corresponding term $\delta_{\log\text{-II}}$ above. Hence, a consistent estimator of the value function can thereby be obtained via

$$
\begin{aligned}
\nabla_{x_t} V^r_t(x_t) &\approx \sum_{i=1}^N \text{softmax}(v_1, \cdots, v_N)[i](B^i + \nabla \log p_t(x_t) - \nabla \log p_{t'}(x_t)) + \nabla \log p_{t'}(x_t) - \nabla \log p_t(x_t) \\
&= \sum_{i=1}^N \text{softmax}(v_1, \cdots, v_N)[i] B^i
\end{aligned}
$$

It remains to compute the v^i and B^i terms:

$$
\begin{aligned}
p_t(x_t|z^i(x_t, \epsilon_i)) &= \mathcal{N}(x_t; \alpha_t z^i(x_t, \epsilon_i), \sigma_t^2 I_d) \\
\log p_t(x_t|z^i(x_t, \epsilon_i)) &= -\frac{1}{2\sigma_t^2} \|x_t - \alpha_t z^i(x_t, \epsilon_i)\|^2 + C \\
q_{t'|t}(x_{t'}|x_t) &= \mathcal{N}\left(\frac{\alpha_{t'}}{\alpha_t} x_t; (\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2) I\right) \\
\log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t) &= -\frac{1}{2(\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2)} \|x_{t'}(x_t, \epsilon_i) - \frac{\alpha_{t'}}{\alpha_t} x_t\|^2 + C' \\
&= -\frac{1}{2(\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2)} \|\frac{\alpha_{t'}}{\alpha_t} x_t + \left(\sqrt{\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2}\right)\epsilon - \frac{\alpha_{t'}}{\alpha_t} x_t\|^2 + C' \\
&= -\frac{1}{2(\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2)} \|\left(\sqrt{\sigma_{t'}^2 - \frac{\alpha_{t'}^2}{\alpha_t^2}\sigma_t^2}\right)\epsilon\|^2 + C' \\
&= -\frac{1}{2}\|\epsilon\|^2 + C' \\
\nabla_{x_t} \log q_{t'|t}(x_{t'}(x_t, \epsilon_i)|x_t) &= 0
\end{aligned}
$$

Therefore, we get:

$$
\begin{aligned}
v_i &= r_{\text{local}}(x_t, \epsilon_i) + \gamma_{t,t'}(x_t, x_{t'}(x_t, \epsilon_i)) + \frac{1}{2}\|\epsilon_i\|^2 + \tilde{C} \\
B^i &= \nabla_{x_t} r_{\text{local}}(x_t, \epsilon_i) + \delta^i_{\text{score}}
\end{aligned}
$$

for a constant $\tilde{C}$ that we can simply drop because we apply the softmax operation. This finishes the proof. □

Algorithm 1 Guidance with Posterior Diamond Maps

Require: Diamond flow model $X^{\theta}_{r,s}(\bar{x}_s|x_t, t)$, pre-trained flow model $u_t(x)$, N simulation steps, K Monte Carlo samples, differentiable reward r
Init: $x_0 \sim \mathcal{N}(0, I_d)$
Set $h \leftarrow 1/N, t \leftarrow 0$
for $n = 0$ to $N - 1$ **do**
  for $k = 1$ to K **do**
    $z^i \leftarrow X^{\theta}_{0,1}(\bar{x}^k_0|x_t, t) \quad (\bar{x}^k_0 \sim \mathcal{N}(0, I_d))$
    $\nabla_{x_t} r(z^k) \leftarrow r(z^i)$.backward$(x_t)$
  end for
  $\nabla_{x_t} V^r_t(x_t) \leftarrow \sum_k \text{softmax}(r(z^j)_{1\leq j\leq K})[k]\nabla_{x_t} r(z^k)$
  $u^r_t \leftarrow u_t(x) + b_t \nabla_{x_t} V^r_t(x_t)$
  $X_{t+h} \leftarrow X_t + h u^r_t, \quad t \leftarrow t + h$
end for
Return: X_1

A.10 Value function estimation with Weighted Diamond Map

Finally, we remark that one could also estimate the value function (instead of its gradient) via a weighted Diamond map. By the Continuity equation

$$
\begin{aligned}
&\log p_t(x_t) - \log p_{t'}(x_t) \\
&= \int_{t'}^{t} \partial_r \log p_r(x_t) \mathrm{d}r \\
&= \int_{t'}^{t} -\nabla \log p_r(x_t)^T u_r(x_t) - \nabla \cdot u_r(x_t) \mathrm{d}r \\
&= -\mathbb{E}_{r\sim \text{Unif}_{[t',t]}, \epsilon\sim\mathcal{N}(0,I_d)} \left[\nabla \log p_r(x_t)^T u_r(x_t) + \epsilon^T D_{x_t} u_r(x_t)\epsilon\right]
\end{aligned}
$$

Therefore, we know that by equation (42):

$$
\begin{aligned}
&V^r_t(x_t) \\
&= \log \mathbb{E}_{q_{t'|t}(\cdot|x_t)} \left[\exp(r(z) + \gamma_t(x_{t'})) \frac{p_t(x_t|z)}{q_{t'|t}(x_{t'}|x_t)}\right] \\
&+ \mathbb{E}_{r\sim \text{Unif}_{[t',t]}, \epsilon\sim\mathcal{N}(0,I_d)} \left[\nabla \log p_r(x_t)^T u_r(x_t) + \epsilon^T D_{x_t} u_r(x_t)\epsilon\right]
\end{aligned}
$$

The last term requires backpropagation and constitutes a Hutchinson trace estimator term. Hence, this might be less efficient, yet note that the Hutchinson's trace estimator is for fixed x_t (does not depend on the sample $x_{t'}$ that is drawn) and therefore the second summand constitutes a fixed offset.

B Details on Diamond Maps

B.1 Guidance with Posterior Diamond Maps

We present an algorithm in Algorithm 1.

B.2 Search or Sequential Monte Carlo (SMC) with Posterior Diamond Maps

We discuss Sequential Monte Carlo (SMC) with Diamond Maps. SMC evolves M particles $X^1_t, \cdots, X^M_t$ simultaneously. To evolve them in time, we use a proposal distribution given by the DDPM transitions either sampled via Theorem 4.2 in one step. Further, resampling happens based on potentials given by $U_{t,t'}(x_t, x_{t'})$. Here, we choose $U_{t,t'}(x_t, x_{t'}) = V^r_{t'}(x_{t'}) - V^r_t(x_t)$, i.e. the potentials favors particles promising to increase the value function most. We can estimate the

Algorithm 2 Sequential Monte Carlo with Posterior Diamond Maps

Require: Diamond flow $X^{\theta}_{s,r}(\bar{x}_s|x_t, t)$, N transitions, M num. of partic., Monte Carlo samples K, reward r
Init: $x_0^m \sim \mathcal{N}(0, I_d), V_0^m = U_0^m = 0 \quad (m \leq M)$
Set $h \leftarrow 1/N, t \leftarrow 0$
for $n = 0$ to $N - 1$ **do**
  for $m = 1$ to M **do**
    $x_{t+h}^m \sim p^{\text{DDPM}}_{t'|t}(\cdot|x_t^m)$ (via Theorem 4.2)
    $t \leftarrow t + h$
    $V_t^m \leftarrow 0$
    for $k = 1$ to K **do**
      $z^k \leftarrow X^{\theta}_{0,1}(\bar{x}_0^{k,m}|x_t, t) \quad (\bar{x}_0^{k,m} \sim \mathcal{N}(0, I_d))$
      $V_t^m \leftarrow V_t^m + \frac{1}{K}\exp(r(z^i))$
    end for
    $V_t^m = \log V_t^m$
    $U_t^m \leftarrow U_t^m + V_t^m - V_t^m$
  end for
  Resample particles: $i_k \sim \text{softmax}(U_t^1, \ldots, U_t^M)$
  $x_t^k \leftarrow x_t^{i_k}, U_t^k \leftarrow U_t^{i_k}, V_t^k \leftarrow V_t^{i_k}$
end for
Return: $x_1^1, \cdots, x_1^K$

value function $V_t^r(x_t)$ via

$$V_t^r(x_t) \approx \log \frac{1}{N} \sum_{i=1}^{N} \exp(r(z^i)) \quad z^i = X^{\theta}_{0,1}(\bar{x}_0^i|x_t, t) \tag{43}$$

where $\bar{x}_0^i \sim \mathcal{N}(0, I_d)$ for every $i = 1, \cdots, N$. The full method is summarized in Algorithm 2.

Same as for guidance, achieving the same estimators of the value function would be $\mathcal{O}(N)$ steps more expensive for a standard flow model. Here, this speed-up is explained by two factors: (1) Transitions $p^{\text{DDPM}}_{t'|t}$ can be obtained in one step (2) Samples $p_{1|t}$ can be obtained in one step.

Remark - Search. We can apply Diamond Maps similarly to search and obtain the same speed-up. Search proceeds similar to SMC but unlike SMC it does not keep a population of particles but collapses all particles into a single particle with maximum potential, i.e. the softmax is taken with respect to zero temperature. With this difference, Algorithm 2 can be easily modified to apply Diamond Maps to search methods.

B.3 General GLASS transitions

General transitions. We note that our framework would also work for general GLASS transitions (i.e. not limited to the DDPM transitions) and we could similarly distill these transitions into a flow map. However, we would need to condition the flow map on an additional input t', which adds complexity and redundancy to the learning problem that we choose to avoid here by leveraging Theorem 4.2.

B.4 Guidance with Weighted Diamond Maps

We present an algorithm in Algorithm 3.

C Extended Related Work

Inference-time reward alignment. Guidance methods (Chung et al., 2022; Abdolmaleki et al.; Ye et al., 2024; Yu et al., 2023; Bansal et al., 2023; He et al., 2023; Graikos et al., 2022; Song et al., 2023b;a; Feng et al., 2025) were used particularly for solving inverse problems such as Gaussian

Algorithm 3 Guidance with Weighted Diamond Maps

Require: Diamond flow model $X^{\theta}_{r,s}(\bar{x}_s|x_t, t)$, pre-trained flow model $u_t(x)$, N simulation steps, K Monte Carlo samples, differentiable reward r
Init: $x_0 \sim \mathcal{N}(0, I_d)$
Set $h \leftarrow 1/N, t \leftarrow 0$
for $n = 0$ to $N-1$ **do**
  $t' = g^{-1}\left(\lambda \frac{\sigma_t^2}{\alpha_t^2}\right)$
  $v_t \leftarrow u_t(x)$
  $v_{t',t} \leftarrow u_{t'}(x)$
  $s_t \leftarrow \text{ConversionVelocityToScore}(v_t, t)$
  $s_{t'} \leftarrow \text{ConversionVelocityToScore}(v_{t',t}, t')$
  for $k = 1$ to K **do**
    $x^k_{t'} \leftarrow \frac{\alpha_{t'}}{\alpha_t} x_t + \sqrt{\sigma^2_{t'} - \frac{\alpha^2_{t'}}{\alpha^2_t}\sigma^2_t}\epsilon^k \quad (\epsilon^k \sim \mathcal{N}(0, I_d))$
    $z^k \leftarrow X^{\theta}_{t',1}(x^k_{t'})$
    $r^k_{\text{local}} = r(z^k) - \frac{\|x_t - \alpha_t z^k\|^2}{2\sigma_t^2}$
    $\nabla_{x_t} r^k_{\text{local}} \leftarrow r^k_{\text{local}}.\text{backward}(x_t)$
    $v^k_{t'} \leftarrow u_{t'}(x^k_{t'})$
    $s^k_{t'} \leftarrow \text{ConversionVelocityToScore}(v^k_{t'}, t')$
    $\delta^i_{\text{score}} = s^k_{t'} \frac{\alpha_{t'}}{\alpha_t} - s_t$
    $\gamma_k \leftarrow \frac{1}{2}\left(s^k_{t'} + s_{t'}\right)^T (x^k_{t'} - x_t)$
    $v_k \leftarrow r^k_{\text{local}} + \gamma_k + \frac{1}{2}\|\epsilon_k\|^2$
  end for
  $\nabla_{x_t} V^r_t(x_t) \leftarrow \sum_k \text{softmax}(v_1, \cdots, v_K)[k](\nabla_{x_t} r^k_{\text{local}} + \delta^i_{\text{score}})$
  $u^r_t \leftarrow v_t + b_t \nabla_{x_t} V^r_t(x_t)$
  $X_{t+h} \leftarrow X_t + h u^r_t$
  $t \leftarrow t + h$
end for
Return: X_1

deblurring or inpainting. The inaccurate denoiser approximation (see (8)) is a known limitation of guidance methods and many methods aim to find better - yet biased - approximations (Song et al., 2023a). Here, we are taking a more "radical" step of learning the posterior directly. SMC (Singhal et al., 2025; Skreta et al., 2025; Wu et al., 2023a; Mark et al., 2025; He et al., 2025) and search (Li et al., 2025b; Zhang et al., 2025) evolve a population of particles and are based on filtering out unpromising particles. Their main challenges are expensive or biased estimation of the value function $V_t(x_t)$ or a collapse of diversity of the population of particles. Many other approaches (Yeh et al., 2024; Wu et al., 2024; Krishnamoorthy et al., 2023) exist and we refer to (Uehara et al., 2025) for a review.

Reward fine-tuning. Reward fine-tuning methods fine-tune flow and diffusion models at training based on a variery of RL techniques such as GRPO (Xue et al., 2025; Li et al., 2025a; Liu et al.), stochastic optimal control (Liu et al.; Domingo-Enrich et al., 2024), DPO (Wallace et al., 2024). Many of these methods require expensive simulation and stochastic transitions for exploration during training (Liu et al., 2025; Xue et al., 2025; Li et al., 2025a; Domingo-Enrich et al., 2024) that we learn in a one-step sampler.

D Experimental Details and Further Experiments

D.1 Posterior Diamond Map

Architecture The flow map is a EDM2 model Karras et al. (2023) with configuration from Table 2 that has time parameters t, t' and input x_t. The Diamond Map is a similar EDM2 model with 2 additional time parameters s, r and input $\bar{x}_s$ and conditioning variable x_t.

	CIFAR-10	**CelebA-64**
Dataset Properties		
Dimensionality	$3 \times 32 \times 32$	$3 \times 64 \times 64$
Samples	50k	203k
Network		
Architecture	EDM2	EDM2
Hidden/base channels	128	128
Channel multipliers	[2, 2, 2]	[1, 2, 3, 4]
Residual blocks	4 per resolution	3 per resolution
Attention resolutions	16×16	$16 \times 16, 8 \times 8$
Dropout	0.13	0.0
Hyperparameters		
Batch size	512	256
Training steps	400,000	800,000
Total samples	204.8×10^6	204.8×10^6
Optimizer	RAdam	RAdam
Learning rate	10^{-2}	10^{-2}
LR schedule	Sqrt decay at 35k	Sqrt decay at 35k
Gradient clipping	1.0	1.0
Diagonal fraction η	0.75	0.75
EMA decay	0.9999	0.9999
Evaluation		
Metric	FID	FID
Sample count	50,000	50,000
Methods		
Algorithms	LSD	LSD
Compute		
GPU	8x L40S	6x 40GB A100
Precision	fp32	bfloat16

Table 2: **Experimental setup**. Flow map and Diamond map both trained using these configurations. Diamond map contains additional embeddings and channels for extra time steps and conditioning variables as described in (15).

Experimental Setup We fix $t' = 1$ for easier training but can still sample arbitrary t' using Diamond DDPM Early stop sampling. Because of singularities for calculating the time reparameterization r^*, we limit t to $[0.01, 1.0]$ during training and sampling as well. We sample t uniformly $t \sim U([0.01, 1.0])$ and s, r uniformly from the upper triangle of $[0.01, 0.99]^2$. We embed $t, t' - t$ and $s, r - s$ based on empirical benefits seen in Boffi et al. (2025b). We also simiarly employ loss reweighting but calculate weights with all 4 diamond time steps instead of just 2 for a flow map. We use linear outer time schedulers $\alpha_t = t, \sigma_t = 1 - t$ and the corresponding inner time schedulers $\bar{\alpha}_s, \bar{\sigma}_s$ from Holderrieth et al. (2025). We compute FID for 4 different NFEs: $1, 2, 4, 8$ (Table 3). The flow method uses the flow map but gets the velocity by purely using the diagonal $t = t'$. The flow map method uses the vanilla flow map sampling with $t \to t'$ where $t < t'$. GLASS follows the same sampling procedure as Holderrieth et al. (2025) and uses the same velocity as the flow. Lastly, we sample from the $p^{\text{DDPM}}_{1|0}(x_1|x_0)$ transition kernel with inner steps equal to NFE for the Diamond Map.

D.2 Weighted Diamond Maps

Hyperparameters. Weighted Diamond maps has 4 main parameters: the amount of particles K, the amount of guidance steps N, the renoising time $t' < t$ and time interval to apply guidance in. We set b_t according to $b_t = \sigma_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\sigma}_t \sigma_t$.

Dataset	Method	Step Count			
		1	2	4	8
CIFAR-10 (FID ↓)	Flow	402.049	145.275	41.231	14.187
	Flow Map	**10.604**	**4.596**	**4.180**	**4.880**
	GLASS	378.674	157.554	39.472	11.597
	Diamond Map	12.551	5.799	5.803	6.732
CelebA-64 (FID ↓)	Flow	199.933	89.582	45.461	24.762
	Flow Map	**9.534**	**4.052**	**3.083**	**3.146**
	GLASS	208.509	95.245	51.803	26.333
	Diamond Map	16.084	9.160	6.736	5.776

Table 3: **Pretraining results**. Flow map and Diamond flow map are trained with LSD loss. Performance across sampling step counts (1-8) for CIFAR-10 and CelebA-64 datasets (FID). Best method per dataset and step count shown in **bold**. As one can see, the flow map contained in a Posterior Diamond Map performs almost on par with the standard Flow Map.

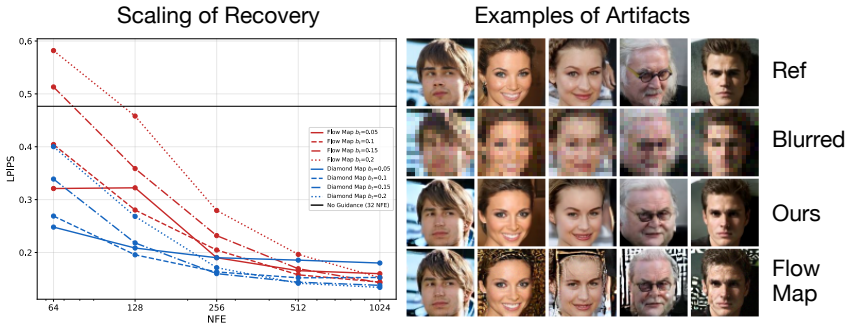

Figure 6: . Applying Posterior Diamond Maps (see Section 4.1) to super resolution for various reward scales b_t. Posterior Diamond Maps has a better Pareto frontier of optimal results for optimal compute. The reason for that is that we find is that Posterior Diamond Maps shows significantly higher robustness towards high reward scales, likely caused by stochasticity, compared to guidance with a naive flow map approximation.

We first discuss, how one should select $t' < t$? We propose to set such that the signal-to-noise ratio for t' increases by a factor of λ: specifically, for $g(t) = \sigma_t^2/\alpha_t^2$, we set via $t' = g^{-1}\left(\lambda \frac{\sigma_t^2}{\alpha_t^2}\right)$. Through this, we move the hyperparameter to the SNR-ratio λ.

SNR and Number of Particles. We first ablate the SNR together with the number of particles while keeping guidance steps fixed to 5 in time-frame $[0.05, 0.25]$. As seen in Table 4, we find that while using a few particles $K > 2$ gives significant improvements, scaling the number of particles up, does not result in additional improvements.

Number of guidance steps. Next we ablate the number of guidance steps. As visualized in Figure 5, we do see significant improvements when scaling up the number of guidance steps with diminishing returns after scaling to $N > 10$. Thus, we recommend keeping particles fixed to a range of [4, 8] while altering the number of guidance steps depending on available compute. For the time-frame, we find that $[0.05, 0.25]$ provides the better performance compared to longer time-frames with the same amount of steps, e.g. $[0.05, 0.45]$(Table 5).

Figure 7: Qualitative examples of CLIP-reward alignment with Sequential Monte Carlo with Posterior Diamond Maps. Model trained on CelebA.

Table 4: **Ablation of SNR and Particle Scaling**. Mean GenEval scores using 5 guidance steps ($0.05 \rightarrow 0.25$). Total NFE $= 35$. The baseline configuration is highlighted.

SNR (λ)	**Particles** (P)				
	1	**2**	**4 (Default)**	**16**	**32**
2.0	0.7775	0.8177	0.8202	0.8182	0.8207
5.0	0.7821	0.8118	0.8136	0.8089	0.8111
10.0	0.7896	0.8155	0.8067	0.8210	0.8223
20.0	0.8008	0.8095	**0.8272**	0.8239	0.8285

Table 5: **Guidance Strategy Ablation**. Comparison of step counts and time horizons at SNR 20.0 and 4 Particles. NFE is calculated as $5N + N$.

Guidance Range	**Steps** (N)	**NFE**	**Mean** $\uparrow$
0.05 $\rightarrow$ 0.25 (Default)	**5**	**30**	**0.8272**
0.05 $\rightarrow$ 0.45	5	30	0.8163
0.05 $\rightarrow$ 0.25	10	60	0.8229
0.05 $\rightarrow$ 0.45	10	60	0.8284