ANNEALED BACKWARD–BACKWARD SPLITTING FOR TRAINING-FREE DIFFUSION GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free guidance specifies a condition for a pretrained diffusion model via a differentiable loss without additional training, and is therefore applicable across tasks and domains. Most existing methods add to each reverse step a gradient step on the loss, whose step size controls how far the guided iterate moves. Choosing this step size is difficult. The standard rule sets it from the smoothness of the loss, but this smoothness is often unavailable (e.g., when the loss involves a pretrained network), so the step size must be tuned separately for each task. In this paper, we remove the step size from the design. We formulate conditional generation as a sequence of subproblems indexed by the noise level, and we address each subproblem by composing two proximal steps, one for the prior and one for the condition, so that each subproblem (rather than step size) determines how far the guided iterate moves. In training-free guidance, we cannot compute either proximal step in closed form, because the prior is available only through the pretrained diffusion model and the loss may involve a pretrained network. We therefore approximate the proximal step for the prior by one reverse step, whose scale we identify from the noise schedule with a bound on its approximation error, and the proximal step for the condition by an inexact proximal step. We refer to the resulting method as ABBS (annealed backward-backward splitting). In our experiments, ABBS achieves the best guidance validity on three of four image tasks, in data space and latent space, with fidelity comparable to training-free guidance baselines. ABBS also applies to human motion editing, where it achieves editing performance on par with the strongest motion-editing baseline at a substantially lower guidance cost.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Song et al., 2021) provide expressive priors over high-dimensional data, and they support generation in imaging, audio, and motion (Daras et al., 2024; Cao et al., 2025; Lemercier et al., 2025; Tevet et al., 2023). Tasks in these domains, e.g., restoring a degraded image or editing motion to follow a target trajectory, are posed as conditional generation, i.e., producing an output that satisfies an externally specified condition while remaining consistent with the prior. Conditional generation with diffusion models is commonly addressed with classifier guidance (Dhariwal & Nichol, 2021) or classifier-free guidance (Ho & Salimans, 2022), both of which require training for each condition type. In contrast, training-free guidance methods specify the condition for a pretrained diffusion model through a differentiable loss without additional training (Ye et al., 2024), which makes them applicable across tasks and domains.

Most existing training-free guidance methods add a gradient step on the guidance loss to each reverse step, and they evaluate the loss at the posterior-mean estimate of the clean sample (He et al., 2024; Chung et al., 2023; Yu et al., 2023; Song et al., 2023; Ye et al., 2024). The step size of that gradient step determines the displacement, i.e., the distance between the iterate after guidance and the output of the reverse step. If the displacement is too small, the condition is not satisfied, and if it is too large, the iterate leaves the data manifold (Chung et al., 2023). To choose the step size, we can classically rely on an a priori bound in terms of the Lipschitz constant of the loss gradient (see, e.g., Nesterov, 2004), provided that we read the gradient step as a step for minimizing the loss. However, that constant is not available in closed form when a pretrained network supplies the loss. The benchmark

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Summary of existing methods for training-free guidance and the proposed method. †Gradient-step guidance refers to DPS (Chung et al., 2023), LGD (Song et al., 2023), MPGD (He et al., 2024), FreeDoM (Yu et al., 2023), and TFG (Ye et al., 2024), and ADMMDiff refers to the method of Zhang et al. (2025). "Applied" means that the weights of the subproblem are set so that one reverse step performs the proximal step on the prior.

Method	What sets the displacement	Problem formulation (Sec. 3.1)	Scale of the prior step (Sec. 3.2)
†Gradient-step guidance	Step size	Single problem	Not considered
ADMMDiff	**Subproblem scale**	Single problem	Stated, unexamined
ABBS (ours)	**Subproblem scale**	**Annealed subproblems**	**Identified and applied**

of Ye et al. (2024) reports that the appropriate step size varies considerably across tasks and datasets, and each of the methods above therefore tunes the step size per task.

Another line of work obtains the displacement from a subproblem rather than from a step size. These methods formulate conditional generation as the minimization of two terms, one built from the condition and one built from the prior, and variable splitting (see, e.g., Boyd et al., 2011) assigns the two terms to two variables coupled by an equality constraint. Each variable is updated by a proximal step, i.e., by the proximity operator of one term (see Eq. 1). Variable splitting applies to a diffusion model because one reverse step corresponds to an approximate proximal step on the negative log density of the prior (Zhu et al., 2023, App. A.1). ADMMDiff (Zhang et al., 2025) adopts this splitting, couples the two variables by the alternating direction method of multipliers (ADMM) (see, e.g., Boyd et al., 2011), and solves the condition subproblem by an inner loop of gradient steps. In ADMMDiff, the correspondence serves only to justify replacing the proximal step on the prior by a reverse step (Zhang et al., 2025, Prop. 1). The scale of that proximal step (equivalently, the weight of the prior in the subproblem) is left unexamined.

To develop a guidance method effective across a broad range of guidance tasks, we follow this line of work, which obtains the displacement from a subproblem rather than from a step size. We formulate conditional generation as loss minimization on the data manifold. We replace the manifold constraint by a penalty on the prior, smoothed at the noise level of one time step, which yields one subproblem per time step. In each subproblem, we assign the penalty and the condition loss to two variables coupled by an equality constraint, and we address it by composing two proximal steps, one on the penalty and one on the loss. This composition is known as a backward–backward splitting (see, e.g., Combettes & Pesquet, 2011). The core of our approach is to identify precisely how one reverse step relates to this composition. We show that, for a family of samplers that includes those of Ho et al. (2020) and Song et al. (2021), one reverse step is an inexact proximal step on the prior, at an explicit scale determined by the noise schedule and the sampler, together with an explicit bound on the residual (Proposition 1). This is what distinguishes our work from prior methods that also use schedule-derived quantities. TFG (Ye et al., 2024) and ADMMDiff (Zhang et al., 2025) set a weight from the schedule, but they do not identify which proximal step a reverse step performs. ADMMDiff (Zhang et al., 2025, Prop. 1) uses the reverse step only to justify replacing the proximal step on the prior, and it leaves the scale of that step unexamined. In contrast, we identify that a reverse step is an inexact proximal step on the prior at this scale, and we bound the approximation it incurs. This scale is what sets the displacement of one guided step, so identifying it, rather than leaving it as a free weight, is what removes the step size from the design. Because neither proximity operator has a closed form (the prior is available only through the pretrained diffusion model and the loss may involve a pretrained network), we approximate the proximity operator of the prior term by one reverse step and that of the condition term by an inexact proximal step, i.e., by running an iterative method for a fixed number of iterations. We refer to the proposed method as ABBS (annealed backward-backward splitting), because it applies a backward-backward splitting at each noise level. Table 1 compares ABBS with the existing methods.

Main contributions. We summarize the contributions of this paper as follows.

(C1) We formulate conditional generation as a sequence of subproblems indexed by the noise level, and for each subproblem we derive a backward-backward splitting, one proximity operator

for the prior term and one for the condition term (Section 3.1). The scale of each proximity operator, and not a step size, therefore sets the displacement of one reverse step.

(C2) We show that one reverse step, for a family of samplers that includes those of Ho et al. (2020) and Song et al. (2021), approximates the proximity operator of the prior term at a scale that the noise schedule and the sampler determine, and we give an explicit bound on the residual (Proposition 1). Using Proposition 1, we derive ABBS, which computes the backward-backward step of (C1) with a pretrained diffusion model (Section 3.2).

(C3) We evaluate ABBS on four image tasks, namely super-resolution, deblurring, segmentation map guidance, and style guidance, in data space and in latent space, and it attains the best guidance validity on three of the four tasks with fidelity comparable to the baselines (Section 4.1). We further apply it to human motion editing, where it reaches editing performance competitive with the strongest baselines at a substantially lower guidance cost (Section 4.2).

Notation. $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{++}$ denote respectively the sets of all nonnegative integers, all real numbers, all nonnegative real numbers, and all positive real numbers. $\mathcal{X}$, $\mathcal{Z}$, and $\mathcal{Y}$ denote respectively the data space, the latent space of a pretrained autoencoder, and the observation space of the task, and $\mathcal{S} \in \{\mathcal{X}, \mathcal{Z}\}$ denotes the space on which the pretrained diffusion model is defined (see Section 2). Each of these spaces is a Euclidean space with the inner product $\langle \cdot, \cdot \rangle$ and the induced norm $\|\cdot\|$. Id, $\mathbf{0}$, and I stand respectively for the identity mapping, the zero vector, and the identity matrix. For a linear map $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$, $\|\mathcal{A}\|_{\mathrm{op}} := \sup\{\|\mathcal{A}\boldsymbol{x}\| \mid \boldsymbol{x} \in \mathcal{X},\ \|\boldsymbol{x}\| \leq 1\}$ denotes the operator norm of $\mathcal{A}$. For $h : \mathcal{S} \to \mathbb{R} \cup \{+\infty\}$ and $\gamma \in \mathbb{R}_{++}$, the proximity operator of h of index γ is the set-valued mapping

$$\mathrm{prox}_{\gamma h} : \mathcal{S} \to 2^{\mathcal{S}} : \boldsymbol{s} \mapsto \arg\min_{\boldsymbol{u} \in \mathcal{S}} \Big(h(\boldsymbol{u}) + \tfrac{1}{2\gamma} \|\boldsymbol{u} - \boldsymbol{s}\|^2 \Big). \tag{1}$$

2 Problem formulation

Diffusion model

A diffusion model samples from the clean-data distribution p_0 on $\mathcal{S}$ with a noise-prediction network $\boldsymbol{\varepsilon}_\theta : \mathcal{S} \times \{1, \ldots, T\} \to \mathcal{S}$. With the variance schedule $(\beta_t)_{t=1}^T \subset (0, 1)$ of Ho et al. (2020), $\alpha_t := 1 - \beta_t$, and $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$, the forward process adds Gaussian noise to $\boldsymbol{x}_0 \sim p_0$ at the time step $t \in \{1, \ldots, T\}$ as

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\, \boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\varepsilon}, \tag{2}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, I)$, and we write p_t for the distribution of $\boldsymbol{x}_t$. The network is trained by solving

$$\boldsymbol{\varepsilon}_\theta = \arg\min_{\boldsymbol{\xi}} \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_0 \sim p_0,\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, I)} \big\| \boldsymbol{\xi}(\boldsymbol{x}_t, t) - \boldsymbol{\varepsilon} \big\|^2 \tag{3}$$

with $\boldsymbol{x}_t$ of Eq. 2, and its exact solution satisfies $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) = -\sigma_t \nabla \log p_t(\boldsymbol{x}_t)$ with $\sigma_t := \sqrt{1 - \bar{\alpha}_t}$. For sampling, we use the estimate of $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$ that Tweedie's formula (Efron, 2011) provides,

$$\hat{\boldsymbol{x}}_{0|t} := \mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_t] = \tfrac{1}{\sqrt{\bar{\alpha}_t}} \big(\boldsymbol{x}_t + \sigma_t^2 \nabla \log p_t(\boldsymbol{x}_t) \big) = \tfrac{1}{\sqrt{\bar{\alpha}_t}} \big(\boldsymbol{x}_t - \sigma_t\, \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) \big). \tag{4}$$

One reverse step carries $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-1}$ by

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, \hat{\boldsymbol{x}}_{0|t} + \sqrt{1 - \bar{\alpha}_{t-1} - \omega_t^2}\, \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) + \omega_t\, \boldsymbol{\varepsilon}', \tag{5}$$

where $\boldsymbol{\varepsilon}' \sim \mathcal{N}(\mathbf{0}, I)$ and $\omega_t \in [0, \sqrt{1 - \bar{\alpha}_{t-1}}]$ is the level of the injected noise, so a sampler is specified by the choice of ω_t alone. The samplers of Ho et al. (2020) and Song et al. (2021) are the choices $\omega_t = \sigma_{t-1}\sqrt{\beta_t}/\sigma_t$ and $\omega_t = 0$, respectively.

Conditional generation for diffusion guidance

We state the conditional generation task as a constrained minimization problem. A conditionally generated sample must meet two requirements: (i) it should agree with the conditional signal $\boldsymbol{y} \in \mathcal{Y}$ that specifies the task (see Example 1 for instances); (ii) it should lie on the support of the clean-data distribution p_0, because agreement with $\boldsymbol{y}$ alone does not restrict a sample to that support.

Following the manifold hypothesis used by He et al. (2024), we write $\mathcal{M} := \operatorname{supp} p_0 \subset \mathcal{S}$ for the clean-data manifold, and we encode the requirement imposed by $\boldsymbol{y}$ through a data-fidelity loss $f_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}_+$, which is differentiable; a smaller value of $f_{\mathcal{S}}(\boldsymbol{x})$ means that the sample $\boldsymbol{x}$ agrees more closely with the conditional signal $\boldsymbol{y}$. Then, the conditional generation is formulated as the following optimization problem:

$$\underset{\boldsymbol{x} \in \mathcal{M}}{\text{minimize}} \quad f_{\mathcal{S}}(\boldsymbol{x}). \tag{6}$$

The manifold $\mathcal{M}$ is not available in closed form, so Problem 6 cannot be solved as a constrained program directly. Therefore, a training-free method must access $\mathcal{M}$ through the pretrained network $\boldsymbol{\varepsilon}_\theta$ alone.

Example 1 (Instances of the guidance loss $f_{\mathcal{S}}$)**.**
The guidance losses that we consider are of the form

$$f_{\mathcal{S}}(\boldsymbol{x}) = \big\|\Phi(\boldsymbol{x}) - \boldsymbol{y}\big\|^2, \tag{7}$$

where $\Phi : \mathcal{S} \to \mathcal{Y}$ carries a sample to the observation space $\mathcal{Y}$ of the task, and they differ only in Φ and $\boldsymbol{y}$. When $\mathcal{S} = \mathcal{Z}$, the map Φ carries the decoder $\mathcal{D}_\theta : \mathcal{Z} \to \mathcal{X}$ inside, so that $f_{\mathcal{S}}$ measures the agreement after decoding. When $\mathcal{S} = \mathcal{X}$, we set $\mathcal{D}_\theta := \mathrm{Id}$, so that $\mathcal{D}_\theta$ is a map from $\mathcal{S}$ to $\mathcal{X}$ in either case. We now give three instances, which differ in Φ and $\mathcal{Y}$ and together cover the tasks of Section 4.1.

(a) (Linear inverse problem in the data space) Let $\mathcal{S} = \mathcal{X}$, let $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ be a linear degradation operator, and let $\boldsymbol{y} = \mathcal{A}\boldsymbol{x} + \boldsymbol{n}$ for a clean sample $\boldsymbol{x} \in \mathcal{M}$ and a noise vector $\boldsymbol{n}$. Then $\Phi := \mathcal{A}$, and $\nabla f_{\mathcal{S}}$ is Lipschitz continuous with the constant $L_f = 2\|\mathcal{A}\|_{\mathrm{op}}^2$, which $\mathcal{A}$ supplies. Here $\mathcal{A}$ is a downsampling operator in super-resolution and a blur operator in deblurring.

(b) (Guidance by a pretrained network in the data space) Let $\mathcal{S} = \mathcal{X}$, let $\mathcal{B} : \mathcal{X} \to \mathcal{Y}$ be a face-parsing network, and let $\boldsymbol{y}$ be a target parsing map. Then $\Phi := \mathcal{B}$, and no Lipschitz constant of $\nabla f_{\mathcal{S}}$ is available in closed form, because Φ is a pretrained network.

(c) (Guidance in the latent space) Let $\mathcal{S} = \mathcal{Z}$, let $\psi : \mathcal{X} \to \mathcal{F}$ be the feature map of an image encoder into a feature space $\mathcal{F}$, let $G : \mathcal{F} \to \mathcal{Y}$ return the Gram matrix of its argument, and let $\boldsymbol{y} = G\big(\psi(\boldsymbol{x}_{\mathrm{ref}})\big)$ for a reference style image $\boldsymbol{x}_{\mathrm{ref}}$. Then $\Phi := G \circ \psi \circ \mathcal{D}_\theta$, i.e., Φ decodes a latent, extracts the features of the decoded image, and returns their Gram matrix. As in (b), no Lipschitz constant of $\nabla f_{\mathcal{S}}$ is available, because ψ and $\mathcal{D}_\theta$ are pretrained networks.

3 Proposed Approach

3.1 Training-Free Guidance as Annealed Optimization

Here, we reformulate Problem 6 as a sequence of subproblems indexed by the noise level $\bar{\sigma}_t$, and we derive the optimization step expressed as a proximal composition. Recall that the proximity operator of Eq. 1 returns a point that decreases h while staying close to the input, and that the index γ sets how far the point may move: a larger γ allows a larger move.

The manifold $\mathcal{M}$ of Problem 6 admits no closed form, so we replace the constraint $\boldsymbol{x} \in \mathcal{M}$ by a penalty whose gradient the pretrained network supplies, namely the negative log-density of the prior smoothed by Gaussian noise. We describe the iterate by $\bar{\boldsymbol{x}}_t := \boldsymbol{x}_t/\sqrt{\bar{\alpha}_t}$ with $\bar{\sigma}_t := \sigma_t/\sqrt{\bar{\alpha}_t}$, in which Eq. 2 reads $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_0 + \bar{\sigma}_t\boldsymbol{\varepsilon}$ and $\bar{\sigma}_t$ alone indexes the noise level.[1] Therefore, the distribution of $\bar{\boldsymbol{x}}_t$ is $\bar{p}_t := p_0 * \mathcal{N}(\mathbf{0}, \bar{\sigma}_t^2 I)$, i.e., the smoothed prior above, and we set $g_t : \mathcal{S} \to \mathbb{R} : \bar{\boldsymbol{x}} \mapsto -\log \bar{p}_t(\bar{\boldsymbol{x}})$. Throughout this paper, the letter g denotes a term built from the prior, in contrast to the letter f, which denotes a term built from the condition. The constraint $\bar{\boldsymbol{x}} \in \mathcal{M}$ of Problem 6 amounts to adding to the objective the indicator $\iota_{\mathcal{M}} : \mathcal{S} \to \mathbb{R} \cup \{+\infty\}$ of $\mathcal{M}$, which takes the value 0 on $\mathcal{M}$ and $+\infty$ off $\mathcal{M}$. We use g_t as a smooth surrogate for this indicator. Indeed, $\bar{p}_t$ is obtained by perturbing a sample of $\mathcal{M}$ with Gaussian noise of level $\bar{\sigma}_t$, so g_t takes larger values off $\mathcal{M}$, and these values grow as $\bar{\sigma}_t$ decreases.

[1] $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{x}_t$ determine each other because $\bar{\alpha}_t \in (0, 1]$, and we use the reverse direction as the identity $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\,\bar{\boldsymbol{x}}_t$ throughout. The distribution of $\boldsymbol{x}_t$ is instead the smoothing of the distribution of $\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0$, whose support is $\sqrt{\bar{\alpha}_t}\mathcal{M}$ and not $\mathcal{M}$.

The function $f_{\mathcal{S}}$ compares a clean sample with $\boldsymbol{y}$, whereas $\bar{\boldsymbol{x}}_t$ carries noise of level $\bar{\sigma}_t$, so we evaluate $f_{\mathcal{S}}$ at the posterior-mean estimate $\hat{\boldsymbol{x}}_{0|t} = \bar{\boldsymbol{x}}_t - \bar{\sigma}_t \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$ of Eq. 4 and not at $\bar{\boldsymbol{x}}_t$. To use $f_{\mathcal{S}}$ in the subproblem below, we need a function of the variable $\bar{\boldsymbol{x}}$, but $\hat{\boldsymbol{x}}_{0|t}$ is a given point that does not depend on $\bar{\boldsymbol{x}}$. We therefore evaluate $f_{\mathcal{S}}$ at $\hat{\boldsymbol{x}}_{0|t}$ displaced by $\bar{\boldsymbol{x}} - \bar{\boldsymbol{v}}_{t-1}$, where $\bar{\boldsymbol{v}}_{t-1} \in \mathcal{S}$ is the point returned by the prior prox of Eq. 12a (which does not involve f_t):

$$f_t : \mathcal{S} \to \mathbb{R}_+ : \bar{\boldsymbol{x}} \mapsto f_{\mathcal{S}}\big(\hat{\boldsymbol{x}}_{0|t} + \bar{\boldsymbol{x}} - \bar{\boldsymbol{v}}_{t-1}\big), \tag{8}$$

so that f_t agrees with $f_{\mathcal{S}}(\hat{\boldsymbol{x}}_{0|t})$ at the base point $\bar{\boldsymbol{v}}_{t-1}$, i.e., $f_t(\bar{\boldsymbol{v}}_{t-1}) = f_{\mathcal{S}}(\hat{\boldsymbol{x}}_{0|t})$. This shift makes f_t a function of $\bar{\boldsymbol{x}}$ through the base point $\bar{\boldsymbol{v}}_{t-1}$ alone, so minimizing f_t later moves the iterate without recomputing $\hat{\boldsymbol{x}}_{0|t}$, i.e., without another call to $\boldsymbol{\varepsilon}_\theta$. By replacing the constraint of Problem 6 by the penalty g_t, we obtain at each time step t the following problem:

$$\underset{\bar{\boldsymbol{x}} \in \mathcal{S}}{\text{minimize}} \quad f_t(\bar{\boldsymbol{x}}) + \lambda_t\, g_t(\bar{\boldsymbol{x}}), \tag{9}$$

where $\lambda_t \in \mathbb{R}_{++}$. The prior term g_t is accessible only through the pretrained network $\boldsymbol{\varepsilon}_\theta$, whereas the condition term f_t is accessible through $\nabla f_{\mathcal{S}}$. To treat the two terms separately, we introduce an auxiliary variable $\bar{\boldsymbol{v}} \in \mathcal{S}$ on which g_t acts, and we couple it with $\bar{\boldsymbol{x}}$ by an equality constraint:

$$\underset{\bar{\boldsymbol{x}}, \bar{\boldsymbol{v}} \in \mathcal{S}}{\text{minimize}} \quad f_t(\bar{\boldsymbol{x}}) + \lambda_t\, g_t(\bar{\boldsymbol{v}}) \qquad \text{s.t.} \qquad \bar{\boldsymbol{x}} = \bar{\boldsymbol{v}}. \tag{10}$$

We relax the equality constraint of Problem 10 by a quadratic penalty with $\rho_t \in \mathbb{R}_{++}$, which gives the relaxed objective

$$\mathcal{H}_t : \mathcal{S} \times \mathcal{S} \to \mathbb{R} : (\bar{\boldsymbol{x}}, \bar{\boldsymbol{v}}) \mapsto f_t(\bar{\boldsymbol{x}}) + \lambda_t\, g_t(\bar{\boldsymbol{v}}) + \frac{\rho_t}{2}\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{v}}\|^2. \tag{11}$$

We minimize $\mathcal{H}_t$ over $\bar{\boldsymbol{v}}$ with $\bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}_t$ held fixed, and then over $\bar{\boldsymbol{x}}$ with the updated $\bar{\boldsymbol{v}} = \bar{\boldsymbol{v}}_{t-1}$ held fixed. By Eq. 1, these two minimizations are written as

$$\bar{\boldsymbol{v}}_{t-1} \in \operatorname{prox}_{\mu_t g_t}(\bar{\boldsymbol{x}}_t) = \arg\min_{\bar{\boldsymbol{v}} \in \mathcal{S}} \Big[g_t(\bar{\boldsymbol{v}}) + \tfrac{1}{2\mu_t}\|\bar{\boldsymbol{v}} - \bar{\boldsymbol{x}}_t\|^2 \Big], \tag{12a}$$

$$\bar{\boldsymbol{x}}_{t-1} \in \operatorname{prox}_{f_t/\rho_t}(\bar{\boldsymbol{v}}_{t-1}) = \arg\min_{\bar{\boldsymbol{x}} \in \mathcal{S}} \Big[f_t(\bar{\boldsymbol{x}}) + \tfrac{\rho_t}{2}\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{v}}_{t-1}\|^2 \Big], \tag{12b}$$

where $\mu_t := \lambda_t/\rho_t$. By default, we perform one pass of alternating minimization per reverse step, i.e., we apply Eq. 12a once and Eq. 12b once at step t.[2] Each line of Eq. 12 takes a backward step on one of the two terms of Problem 9, so one step of the proposed method is the composition $\operatorname{prox}_{f_t/\rho_t} \circ \operatorname{prox}_{\mu_t g_t}$. We call Eq. 12a the *prior prox* and Eq. 12b the *condition prox*. A composition of two proximity operators is called a backward-backward step in proximal splitting (see, e.g., Combettes & Pesquet, 2011). Eq. 12 is the ideal step, and in general neither of its two proximity operators is available in closed form. In Section 3.2, we derive algorithms based on an approximate computation of both operators.

3.2 Approximation of Backward–Backward Composition on Diffusion Reverse Step

Here, we derive ABBS, an approximate implementation of Eq. 12, by approximating its two proximity operators.

We approximate the prior prox of Eq. 12 by one reverse step.

Proposition 1 (One reverse step is an inexact proximal step on the prior)**.** *Let $t \in \{1, \dots, T\}$, $\bar{\boldsymbol{x}}_t \in \mathcal{S}$, and $\boldsymbol{\varepsilon}' \in \mathcal{S}$, and let $\bar{\boldsymbol{v}}_{t-1} := \boldsymbol{x}_{t-1}/\sqrt{\bar{\alpha}_{t-1}}$ with $\boldsymbol{x}_{t-1}$ given by Eq. 5. Set $b_t := \omega_t/\sqrt{\bar{\alpha}_{t-1}}$, $c_t := 1 - \sqrt{\bar{\sigma}_{t-1}^2 - b_t^2}/\bar{\sigma}_t$, and $\nu_t := c_t \bar{\sigma}_t^2$, so that $\bar{\boldsymbol{v}}_{t-1} = (1 - c_t)\bar{\boldsymbol{x}}_t + c_t \hat{\boldsymbol{x}}_{0|t} + b_t \boldsymbol{\varepsilon}'$ holds, i.e., b_t and c_t are the coefficients of one reverse step at the scale of $\bar{\boldsymbol{x}}_t$ (see Step 2 of Appendix B). Set also $\boldsymbol{\varepsilon}_t^\star := \bar{\sigma}_t \nabla g_t(\bar{\boldsymbol{x}}_t)$, i.e., the value that $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$ takes when Eq. 3 is solved exactly. Assume that (A1) ∇g_t is L_t-Lipschitz continuous with $L_t \in \mathbb{R}_{++}$, and (A2) $\nu_t L_t < 1$. Then $\operatorname{prox}_{\nu_t g_t}$ is single valued, and*

$$\bar{\boldsymbol{v}}_{t-1} = \operatorname{prox}_{\nu_t g_t}(\bar{\boldsymbol{x}}_t) + \boldsymbol{r}_t + b_t \boldsymbol{\varepsilon}' \tag{13}$$

holds, where $\boldsymbol{r}_t$ satisfies $\|\boldsymbol{r}_t\| \le \frac{\nu_t^2 L_t}{1 - \nu_t L_t}\big\|\nabla g_t(\bar{\boldsymbol{x}}_t)\big\| + c_t \bar{\sigma}_t \big\|\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\varepsilon}_t^\star\big\|$.

[2] One can also perform more than one pass at step t. When the prior prox is computed by one reverse step, as in Section 3.2, this repeats the reverse step of level t, which is similar to the time-travel strategy of FreeDoM (Yu et al., 2023).

Proof. See Appendix B. □

Proposition 1 states that one reverse step approximates the prior prox at the scale $\nu_t = c_t\bar{\sigma}_t^2$. The prior prox of Eq. 12a has the index $\mu_t = \lambda_t/\rho_t$. To replace this prior prox by one reverse step, we choose the weights so that $\mu_t = \nu_t$, i.e., $\lambda_t = \rho_t\nu_t$ (see Proposition 1). We set $\lambda_t := \lambda$ for all t with $\lambda \in \mathbb{R}_{++}$, so $\rho_t = \lambda/\nu_t$.

Next, to apply the same procedure to every guidance loss, we compute the condition prox of Eq. 12 approximately, i.e., we compute an approximate minimizer of the objective of Eq. 12b,

$$F_t : \mathcal{S} \to \mathbb{R}_+ : \bar{\boldsymbol{x}} \mapsto f_t(\bar{\boldsymbol{x}}) + \tfrac{\rho_t}{2}\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{v}}_{t-1}\|^2, \tag{14}$$

by a closed form when one is available or by an iterative method otherwise (see Remark 1). Solving proximal subproblems approximately by an inner iterative method is common in optimization (see, e.g., Barré et al., 2023). We write $\boldsymbol{e}_t := \nabla F_t(\bar{\boldsymbol{x}}_{t-1}) = \nabla f_{\mathcal{S}}(\hat{\boldsymbol{x}}_{0|t} + \bar{\boldsymbol{x}}_{t-1} - \bar{\boldsymbol{v}}_{t-1}) + \rho_t(\bar{\boldsymbol{x}}_{t-1} - \bar{\boldsymbol{v}}_{t-1})$ for the residual at the returned point $\bar{\boldsymbol{x}}_{t-1}$ (see Eq. 8). One step of ABBS is

$$\begin{cases} \bar{\boldsymbol{v}}_{t-1} \;\approx\; \operatorname{prox}_{\nu_t g_t}\big(\bar{\boldsymbol{x}}_t\big) + b_t\boldsymbol{\varepsilon}', & \text{(prior prox)} \\ \bar{\boldsymbol{x}}_{t-1} \;\approx\; \operatorname{prox}_{f_t/\rho_t}\big(\bar{\boldsymbol{v}}_{t-1}\big), & \text{(condition prox)} \end{cases} \tag{15}$$

which approximates Eq. 12 in both lines, where $\bar{\boldsymbol{v}}_{t-1}$ is the reverse step of Proposition 1 and $\bar{\boldsymbol{x}}_{t-1}$ is the point that the solver of the condition prox returns. Since $\boldsymbol{e}_t = \nabla F_t(\bar{\boldsymbol{x}}_{t-1})$, we have $\bar{\boldsymbol{x}}_{t-1} = \bar{\boldsymbol{v}}_{t-1} - \big(\nabla f_t(\bar{\boldsymbol{x}}_{t-1}) - \boldsymbol{e}_t\big)/\rho_t$, which is the stationarity condition of Eq. 12b for $\boldsymbol{e}_t = \boldsymbol{0}$. We state Eq. 15 and the solver of the condition prox as pseudocode in Algorithms 1 and 2 (see Appendix C). In practice, we apply the condition prox only at the reverse steps in the guidance window $\mathcal{G} := \big\{t \in \{1, \dots, T\} \;\big|\; \bar{\sigma}_t^2 \le \bar{\sigma}_{\max}^2\big\}$, and we set $\bar{\boldsymbol{x}}_{t-1} := \bar{\boldsymbol{v}}_{t-1}$ at the other reverse steps.

Remark 1 (Inner solver for the condition prox)**.**
Since Algorithm 1 uses only the point that the solver of Eq. 12b returns, we do not fix the solver. If Eq. 12b is available in closed form, e.g., for some instances of the linear inverse problems of Example 1(a), we can use the closed form, and then $\boldsymbol{e}_t = \boldsymbol{0}$. Otherwise, we run an iterative method on F_t from $\bar{\boldsymbol{x}}^{(0)} = \bar{\boldsymbol{v}}_{t-1}$ for a fixed number $K \in \mathbb{N}$ of iterations (see, e.g., Lin et al., 2018) and set $\bar{\boldsymbol{x}}_{t-1} := \bar{\boldsymbol{x}}^{(K)}$, whether or not the iterates have converged (see Algorithm 2). In both cases, the condition prox does not call $\boldsymbol{\varepsilon}_\theta$, because f_t depends on $\bar{\boldsymbol{x}}$ only through the shift of the fixed point $\hat{\boldsymbol{x}}_{0|t}$ (see Eq. 8). Hence, with a first-order method, the condition prox adds K evaluations of $\nabla f_{\mathcal{S}}$ to each reverse step in $\mathcal{G}$. In Section 4, we use Adam (Kingma & Ba, 2015) as the iterative method for all tasks.

Remark 2 (Role of the Lipschitz constants)**.**
In training-free guidance, $f_{\mathcal{S}}$ commonly involves a pretrained network, as in Example 1(b) and (c), and then no Lipschitz constant of $\nabla f_{\mathcal{S}}$ is available (see Section 1). Algorithm 1 does not need any Lipschitz constant to run, because it sets the weight $\rho_t = \lambda/\nu_t$ from the noise schedule and the sampler. We note that a Lipschitz constant enters only the error bound of Proposition 1, which justifies approximating the prior prox by one reverse step, and that this constant is the one of ∇g_t and not of $\nabla f_{\mathcal{S}}$. We cannot verify (A2) at a given reverse step, because L_t depends on the unknown p_0. The bound therefore explains what the approximation incurs, and it is not a condition for running Algorithm 1. Since $\nu_t = c_t\bar{\sigma}_t^2$ grows with the noise level, the residual bound of Proposition 1 is looser at large t. This is not harmful in practice, because the condition prox acts only within the guidance window $\mathcal{G}$ at low noise levels, where ν_t is small, and the bound is tight.

4 Experiments

We evaluate the proposed method from two aspects: (i) its guidance quality against existing training-free guidance methods on standard image inverse problems and guidance tasks, with the pretrained diffusion model fixed across the methods compared (Section 4.1); and (ii) its applicability beyond the image domain, through human-motion editing (Section 4.2).

4.1 Guidance on Image Diffusion Models

In this section, we compare the proposed method against representative existing guidance methods, namely FreeDoM (Yu et al., 2023), MPGD (He et al., 2024), TFG (Ye et al., 2024), and ADM-

Table 2: Benchmarking our method and existing training-free guidance methods on vision tasks. We evaluate Guidance Validity (Val.) and Generation Fidelity (Fid.) using task-specific metrics. The best guidance validity is bold. Following Ye et al. (2024), we do not mark the best generation fidelity, because an unconditional model attains a high fidelity at a low validity, so fidelity alone does not rank the methods.

Method	**Deblurring**		**Super resolution**		**Segmentation map**		**Style guidance**	
	Val. (LPIPS ↓)	Fid. (FID ↓)	Val. (LPIPS ↓)	Fid. (FID ↓)	Val. (Distance ↓)	Fid. (FID ↓)	Val. (Style ↓)	Fid. (CLIP ↑)
FreeDoM	0.247	66.03	0.276	66.97	1753	57.34	454.36	26.53
MPGD	0.265	62.62	0.310	82.91	1733	66.07	435.16	25.96
TFG	**0.245**	59.01	0.281	59.22	1690	54.82	370.70	27.07
ADMMDiff	0.249	37.78	0.273	43.26	1710	55.12	350.17	26.77
ABBS (Ours)	**0.245**	43.49	**0.269**	42.20	**1483**	55.01	**331.25**	27.12

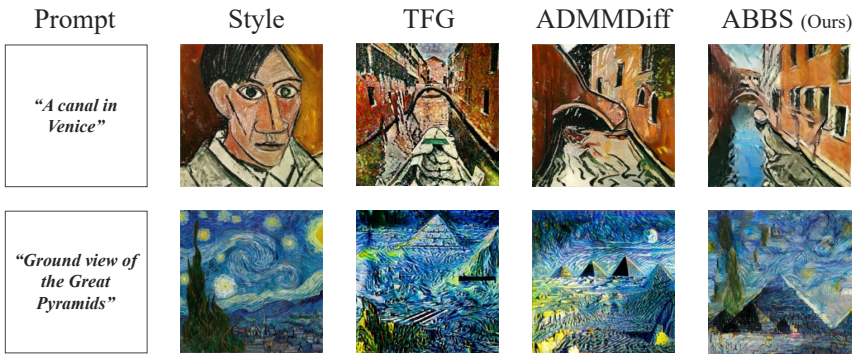

Figure 1: Qualitative comparison on style guidance.

MDiff (Zhang et al., 2025), on four image-domain tasks. Within each task, we fix the pretrained diffusion model across all methods, so that the comparison isolates the guidance procedure.

The tasks of Section 4.1 cover every instance of $f_{\mathcal{S}}$ in Example 1 and both choices of the space $\mathcal{S}$ on which the diffusion model is defined. The four tasks are (i) Gaussian deblurring, (ii) super-resolution, (iii) segmentation-map guidance, and (iv) style guidance. Tasks (i) and (ii) instantiate Example 1(a), Task (iii) instantiates Example 1(b), and Task (iv) instantiates Example 1(c). The measurement map Φ is therefore linear in (i) and (ii) and non-linear in (iii) and (iv). In (i), (ii), and (iii), the diffusion model is defined on the data space, i.e., $\mathcal{S} = \mathcal{X}$, and in (iv) on the latent space of a pretrained autoencoder, i.e., $\mathcal{S} = \mathcal{Z}$. Tasks (i) and (ii) follow the protocol of Chung et al. (2023). Task (iii) follows the protocol of Yu et al. (2023), whose unconditional CelebA-HQ model at 256×256 we use as the pretrained model. The network $\mathcal{B}$ of Example 1(b) is BiSeNet (Yu et al., 2018), and $\boldsymbol{y}$ is a target parsing map. Task (iv) follows the protocol of He et al. (2024) and uses Stable Diffusion v1.4 (Rombach et al., 2022), whose decoder is the $\mathcal{D}_\theta$ of Example 1(c). Appendix D.1 gives further details of the setup.

We assess all methods on the four tasks along the two axes of Ye et al. (2024), guidance validity and generation fidelity, and report the result of each task as a (validity, fidelity) pair. The metric of validity differs between the two inverse problems and the two guidance tasks, because the condition plays a different role in each. In Tasks (i) and (ii), the condition is a degraded observation of a specific image, and that image is available, so we take the LPIPS distance (Zhang et al., 2018) to it as validity. In Tasks (iii) and (iv), we measure validity against the condition itself, namely the parsing-map distance of Yu et al. (2023) for (iii) and the style score for (iv). As fidelity, we report FID for Tasks (i)–(iii) and the CLIP score for Task (iv) (see Appendix D.1).

Table 2 reports Tasks (i) and (ii) in the first two column groups and Tasks (iii) and (iv) in the last two. The proposed method attains the best validity on Tasks (ii), (iii), and (iv), and it ties with TFG on Task (i). Its fidelity is the best on Tasks (ii) and (iv) and second on Tasks (i) and (iii). Figure 1 shows qualitative results on Task (iv). Figures 4 and 5 in Appendix D.1 show qualitative results on Tasks

(i) and (ii) and on Task (iii), respectively. These results show that the proposed method performs favorably against existing guidance methods under both linear and non-linear guidance, and across base models operating in the data space and in the latent space.

The weight λ balances the prior term g_t against the condition term f_t in Problem 9, so a smaller λ weights the condition more heavily and strengthens the guidance. Note that λ is not a step size: it weights the two terms of Problem 9 and does not depend on the smoothness of the loss, so the difficulty of setting a step size discussed in Section 1 does not apply to it. We examine how λ acts on the two axes, taking Task (iv) as an example, for which Table 2 reports ABBS at $\lambda = 5 \times 10^{-2}$ and $K = 30$. Figure 2 varies λ over $\{1, 3, 5, 7, 9\} \times 10^{-2}$, plots the CLIP score against the style score along the resulting curve, and marks on the same plane the point of each baseline. Decreasing λ improves validity at the expense of fidelity, so no single setting is best on both axes. Nevertheless, the setting $\lambda = 5 \times 10^{-2}$ of Table 2 improves on all four baselines in both validity and fidelity. Unlike λ, the number K of inner iterations does not change the subproblem in Eq. 12b, and it determines only how accurately the condition prox solves that subproblem (see Remark 1). Hence, we set $K = 30$, beyond which further increasing K has only a small effect on the results on Task (iv) (Appendix D.1). Since the cost of ABBS grows with K, we also compare the inference time of the methods on Task (iv) in the same appendix.

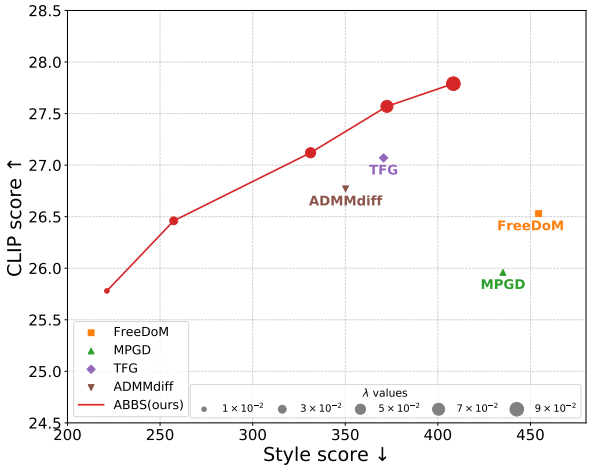

Figure 2: Style score versus CLIP score on the style guidance task. Validity increases to the left, and fidelity increases upward, so a point closer to the upper left attains a better trade-off.

4.2 Application - Human-Motion Editing

This section applies the proposed method to human-motion editing task. We compare the proposed method against representative motion-editing methods, namely OmniControl (Xie et al., 2024), MotionLCM (Dai et al., 2024), and MaskControl (Pinyoanuntapong et al., 2025), which generate in the data space, in a continuous latent space, and in a discrete latent space, respectively.

We adopt two pretrained motion diffusion backbones: Motion Diffusion Model v2 (Tevet et al., 2023), a data-space model, i.e., $\mathcal{S} = \mathcal{X}$, and MLD++ (Dai et al., 2025), a continuous latent-space model, i.e., $\mathcal{S} = \mathcal{Z}$. Since ABBS accesses the backbone only through the reverse step of Eq. 5, together with the decoder $\mathcal{D}_\theta$ when $\mathcal{S} = \mathcal{Z}$, we apply the same guidance procedure to both backbones.

All three baselines attach an additional module of the ControlNet (Zhang et al., 2023) type to a pretrained backbone and train that module on trajectory-labeled data. OmniControl and MaskControl further apply gradient-based guidance at inference. In motion editing, guidance on the controlled joints alone tends to leave the other joints unchanged and to produce unnatural motion, so a trained control module is commonly combined with guidance to keep the whole-body motion natural (Xie et al., 2024). We therefore train a ControlNet for each of our two backbones as well, following Xie et al. (2024) for the data-space backbone and Dai et al. (2024) for the latent-space backbone in both the architecture and the training procedure (see Appendix D.2). We denote by ABBS-$\mathcal{X}$ and ABBS-$\mathcal{Z}$ the combinations of ABBS with the ControlNet on the data-space and on the latent-space backbone, respectively. ABBS-$\mathcal{X}$ and OmniControl thus receive the same kind and the same amount of additional training, and so do ABBS-$\mathcal{Z}$ and MotionLCM, so neither of these two comparisons is confounded by the conditioning module. MaskControl trains a different kind of module on a discrete latent space, so its comparison with the proposed method is not controlled in this sense. That training concerns the conditioning module alone, because the guidance procedure of the proposed method requires no training of its own.

Table 3: Quantitative comparison of human-motion editing on the path-following task. The five editing metrics, highlighted in gray, measure how closely the generated motion follows the target trajectory $\boldsymbol{y}$ and are the primary criterion; bold indicates the best value among the methods in each of them. We measure the AITS of all methods, in seconds, on a single NVIDIA H100 GPU.

Methods	AITS ↓	Traj. err. (>20 cm) ↓	Traj. err. (>50 cm) ↓	Loc. err. (>20 cm) ↓	Loc. err. (>50 cm) ↓	Avg. err.↓	R-Precision Top-3 ↑	FID ↓
Real	–	0.00	0.00	0.00	0.00	0.00	0.797	0.002
OmniControl	74.4	0.263	0.067	0.052	0.007	0.053	0.688	0.191
MotionLCM	11.3	0.085	0.005	0.007	**0.000**	0.039	0.809	0.309
MaskControl	62.3	0.009	0.004	0.001	**0.000**	0.011	0.807	0.070
ABBS-$\mathcal{X}$ (ours)	9.4	**0.008**	**0.000**	**0.000**	**0.000**	**0.004**	0.773	0.148
ABBS-$\mathcal{Z}$ (ours)	0.21	0.070	0.007	0.006	**0.000**	0.019	0.810	0.177

We consider trajectory control, in which $\boldsymbol{y}$ specifies target joint locations for any subset of the joints and any subset of the frames. The loss $f_{\mathcal{S}}$ of Problem 6 is then Eq. 7 with the entries that $\boldsymbol{y}$ leaves unspecified removed by a mask (see Appendix D.2). When $\boldsymbol{y}$ specifies the pelvis location at every frame, the task reduces to path following. We evaluate on the test split of the HumanML3D dataset (Guo et al., 2022). Appendix D.2 states the remaining settings.

We assess all methods along three axes: editing performance, generation quality, and inference cost. Editing performance is the primary criterion, and we report on it the trajectory error, the location error, and the average error of Xie et al. (2024). The first two are the ratios of the generated motions and of the frames whose deviation from $\boldsymbol{y}$ exceeds a threshold, and the third is the mean of that deviation. As generation quality, we report the R-precision and the FID (Guo et al., 2022), and as inference cost, we report AITS, the average inference time per sample.

Table 3 reports the results on path following. On the data-space backbone, ABBS-$\mathcal{X}$ matches or improves on MaskControl, the strongest baseline on the trajectory error, the location error, and the average error, and it does so at a lower AITS than MaskControl. On the latent-space backbone, ABBS-$\mathcal{Z}$ attains an AITS below the lowest one among the baselines (see also Table 6 in Appendix D). The average error of ABBS-$\mathcal{Z}$ is smaller than those of OmniControl and MotionLCM, although MaskControl attains editing errors no larger than those of ABBS-$\mathcal{Z}$. We note that the entries of Table 3 use different backbones, and the generation quality depends on the backbone as well as on the guidance procedure, so the difference in the R-precision and the FID between the entries also reflects the difference in the backbones (see Table 6 in Appendix D). These results show that ABBS-$\mathcal{X}$ attains editing performance comparable to the strongest motion-editing baseline at a lower inference cost, and that the same guidance procedure remains effective on the latent-space backbone. Figure 7 shows qualitative examples on path following (see Appendix D). To isolate the contribution of ABBS, we also report ABBS without the control module on each of the two backbones (see Table 7 in Appendix D). On both backbones, ABBS accounts for most of the editing accuracy, whereas the control module mainly improves the generation quality, in agreement with Xie et al. (2024).

5 Conclusion

We have recast training-free guidance as an optimization procedure, in which the displacement that one reverse step takes is set by the scale of a subproblem. Our starting point is that one reverse step of a sampler in standard use is approximated by a proximal step on the negative log density of the prior, whose scale the noise level determines. We have given an explicit upper bound on the residual of that approximation. Building on this approximation, we have formulated conditional generation as a sequence of subproblems indexed by the noise level, and we have given, at each reverse step, a backward-backward step that addresses the corresponding subproblem. In our experiments, the proposed method performs favorably against existing guidance methods on standard image inverse problems and guidance tasks. As one practical application, we further apply it to human motion editing and compare against existing motion-editing approaches, demonstrating its applicability beyond the image domain. We leave to future work a convergence analysis over the whole sequence of subproblems and an extension of the proposed method to flow matching.

AI USE STATEMENT

We used large language models (LLMs) to polish the writing of this paper and to assist with the implementation of our experiments. For the writing, the models revised drafts that we wrote at the level of sentences and paragraph. For the implementation, the models assisted with writing and debugging the experimental code, and we reviewed and tested all such code. Additionally, we used the models to simulate peer review and critique draft versions of the manuscript to identify potential ambiguities and weaknesses. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available datasets and pretrained models, and it involves no human subjects and no release of new data. As with other conditional generation methods, the proposed method could be used to edit images of real people without their consent.

REPRODUCIBILITY STATEMENT

We give the full proof of Proposition 1 in Appendix B. We give the pseudocode of ABBS in Appendix C. We summarize the experimental settings of each task in Appendix D. For the image tasks, we follow the protocols of Chung et al. (2023), Yu et al. (2023), and He et al. (2024). We describe the evaluation metrics in Appendix D. For human-motion editing, we train the ControlNet modules following the architectures and training procedures of Xie et al. (2024) and Dai et al. (2024) (Appendix D.2). All datasets and pretrained networks that we use are publicly available and are listed in Section 4 and Appendix D. Our code will be made publicly available upon acceptance.

REFERENCES

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Mathieu Barré, Adrien B. Taylor, and Francis Bach. Principled analyses and design of first-order methods with inexact proximal operators. In *Mathematical Programming*. Springer, 2023.

Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2017.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 2011.

Pu Cao, Feng Zhou, Qing Song, and Lu Yang. Controllable generation with text-to-image diffusion models: A survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011.

Wenxun Dai, Ling-Hao Chen, Jingbo Wang, Jinpeng Liu, Bo Dai, and Yansong Tang. Motionlcm: Real-time controllable motion generation via latent consistency model. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

Wenxun Dai, Ling-Hao Chen, Yufei Huo, Jingbo Wang, Jinpeng Liu, Bo Dai, and Yansong Tang. Real-time controllable motion generation via latent consistency model, 2025. URL `https://github.com/Dai-Wenxun/MotionLCM`.

Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Bradley Efron. Tweedie's formula and selection bias. In *Journal of the American Statistical Association*. Taylor & Francis, 2011.

Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. In *Proc. ACM International Conference on Multimedia (ACMMM)*, 2020.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. MoMask: Generative masked modeling of 3D human motions. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. European Conference on Computer Vision (ECCV)*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

Jean-Marie Lemercier, Julius Richter, Simon Welker, Eloi Moliner, Vesa Välimäki, and Timo Gerkmann. Diffusion models for audio restoration: A review. *IEEE Signal Processing Magazine*, 2025.

Jia-Wei Liao, Li-Xuan Peng, Mei-Heng Yueh, Min Sun, Cheng-Fu Chou, and Jun-Cheng Chen. Diffrgd: An inference-time diffusion guidance through riemannian gradient descent. In *Proc. European Conference on Computer Vision (ECCV)*, 2026.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. In *Journal of Machine Learning Research*, 2018.

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2004.

Ekkasit Pinyoanuntapong, Muhammad Usama Saleem, Korrawe Karunratanakul, Pu Wang, Hongfei Xue, Chen Chen, Chuan Guo, Junli Cao, Jian Ren, and Sergey Tulyakov. Maskcontrol: Spatio-temporal control for masked motion synthesis. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. International Conference on Machine Learning (ICML)*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proc. International Conference on Learning Representations (ICLR)*, 2021.

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 *Proc. International Conference on Machine Learning (ICML)*, 2023.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.

Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. Omnicontrol: Control any joint at any time for human motion generation. In *Proc. International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=gd0lAEtWso`.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. TFG: Unified training-free guidance for diffusion models. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Youyuan Zhang, Zehua Liu, Zenan Li, Zhaoyu Li, James J Clark, and Xujie Si. Decoupling training-free guided diffusion by admm. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1219–1229, 2023.

A Related work

A.1 Training-free diffusion guidance

Most training-free guidance methods share a common structure. The loss $f_{\mathcal{S}}$ compares a clean sample with $\boldsymbol{y}$, whereas $\bar{\boldsymbol{x}}_t$ carries noise of level $\bar{\sigma}_t$, so these methods evaluate $f_{\mathcal{S}}$ at the posterior-mean estimate $\hat{\boldsymbol{x}}_{0|t}$ of Eq. 4 and not at $\bar{\boldsymbol{x}}_t$. They then move the iterate along the gradient of the resulting function, i.e., they set $\bar{\boldsymbol{x}}_{t-1} = \bar{\boldsymbol{v}}_{t-1} - \eta_t \nabla_{\bar{\boldsymbol{x}}_t} f_{\mathcal{S}}(\hat{\boldsymbol{x}}_{0|t})$ with a step size $\eta_t \in \mathbb{R}_{++}$.[3] They differ in three respects, namely in how they compute the guidance gradient, in how they repeat the move within one reverse step, and in how they restrict the point where the move arrives.

DPS (Chung et al., 2023), LGD (Song et al., 2023), and MPGD (He et al., 2024) differ from one another in how they compute the guidance gradient. DPS reads that gradient as the score of the likelihood of $\boldsymbol{y}$ given $\boldsymbol{x}_t$, which it obtains by replacing the posterior distribution of the clean sample by the single point $\hat{\boldsymbol{x}}_{0|t}$, and it bounds the Jensen gap that this replacement incurs. LGD attributes the error of that replacement to the single point, so it draws a fixed number of samples around $\hat{\boldsymbol{x}}_{0|t}$ and forms the guidance term from the loss values at those samples by Monte Carlo. MPGD instead differentiates $f_{\mathcal{S}}$ with respect to $\hat{\boldsymbol{x}}_{0|t}$ and updates that estimate directly, so its gradient does not pass through $\boldsymbol{\varepsilon}_\theta$, unlike the gradient of DPS. Once that gradient is assumed to lie in the tangent space of $\mathcal{M}$, the update of MPGD keeps the iterate near the noisy manifold, and composing the loss with a pretrained autoencoder is one way to meet the assumption (He et al., 2024, Thm. 1 and 2). FreeDoM (Yu et al., 2023) and UGD (Bansal et al., 2024) repeat the move within one reverse step. Both renoise the guided iterate back to the level t and apply the guided reverse step again, a fixed number of times per reverse step, under the names time-travel strategy and self-recurrence, respectively. UGD further adds an inner loop that optimizes a correction to $\hat{\boldsymbol{x}}_{0|t}$ by a fixed number of gradient steps and translates the correction back to the noise prediction. TFG (Ye et al., 2024) casts DPS, LGD, FreeDoM, UGD, and MPGD as points of a single hyper-parameter space, so that each of the five occupies a subspace of that space. A search over the space therefore replaces the design of a new method, and TFG proposes such a search because an exhaustive one is out of reach.

DiffRGD (Liao et al., 2026) restricts the point at which the move arrives. DiffRGD confines the guided iterate to a sphere that is centered at the mean of the reverse transition and that carries the radius of the unguided iterate. DiffRGD then moves the iterate on that sphere by Riemannian gradient descent, for a fixed number of iterations per reverse step, and a step size weights each of those iterations.[4]

ADMMDiff (Zhang et al., 2025) writes conditional generation as the minimization of two terms, one built from the prior and one built from the condition, and it assigns the two terms to two variables coupled by an equality constraint. ADMM then updates each variable by a proximal step, so a subproblem and not a step size sets the displacement of one reverse step. ADMMDiff approximates the proximal step on the prior by one reverse step and solves the condition subproblem by an inner loop of gradient steps.

A.2 Human-motion editing

We review the three baselines of Section 4.2, namely OmniControl (Xie et al., 2024), MotionLCM (Dai et al., 2024), and MaskControl (Pinyoanuntapong et al., 2025), which generate in the data space, in a continuous latent space, and in a discrete latent space, respectively. All three attach a module of the ControlNet type (Zhang et al., 2023) to a pretrained backbone, i.e., a trainable copy of the backbone that receives the control signal, and train that module on control signals extracted from the ground-truth motions. They differ in where the training places its supervision, because the backbones differ: OmniControl supervises the control loss in the data space, whereas MotionLCM decodes its latent back to the motion space for the same purpose, and MaskControl combines a loss on the logits with one on the decoded motion. Section 4.2 accounts for this training, so we review here the inference procedure alone.

[3] Some of the methods add the correction to the noise prediction $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$ or to the estimate $\hat{\boldsymbol{x}}_{0|t}$ instead of to $\bar{\boldsymbol{v}}_{t-1}$, and η_t absorbs the factor that the noise schedule and the sampler determine in that case.

[4] DiffRGD is concurrent with the present work.

OmniControl (Xie et al., 2024) generates in the data space on the backbone of MDM (Motion diffusion model; Tevet et al. (2023)), and takes at inference an explicit gradient step on the control loss. That step is repeated within one reverse step, and a hand-set schedule in t fixes the number of repetitions. MotionLCM generates in the continuous latent space of MLD (Chen et al., 2023), and it reduces sampling to a few steps by latent consistency distillation. Its extension (Dai et al., 2025) optimizes the initial latent noise by gradient descent instead of intervening at each reverse step. MaskControl generates discrete tokens through the masked generative model of MoMask (Guo et al., 2024), so one unmasking step plays the role of one reverse step. MaskControl updates the predicted logits by an explicit gradient step on the control loss, and a hand-set number of repetitions is assigned to each unmasking step. The token sampling is not differentiable, so a straight-through Gumbel-softmax estimator replaces it.

In all three, the trained module and the inference procedure act together, so the inference procedure cannot be isolated by comparing the three as they stand. Section 4.2 therefore trains a module of the same type for each of the two backbones that it uses, so that the comparison is not confounded by the conditioning module, although the backbones still differ (see Table 6).

B Proof of Proposition 1

The proof consists of three steps: 1) we show that the proximity operator in Proposition 1 is single-valued and characterize it by its stationarity condition; 2) we write the reverse step without its injected noise as an explicit gradient step on the prior penalty plus the error of the noise prediction; and 3) we bound the difference between that explicit gradient step and the proximity operator, which is the corresponding implicit step.

Proof. **Step 1.** Since $\bar{\sigma}_t = \sqrt{\bar{\alpha}_t^{-1} - 1}$ is strictly increasing in t by $\alpha_t \in (0,1)$, and since $\omega_t \in [0, \sqrt{1-\bar{\alpha}_{t-1}}]$ in Eq. 5 gives $0 \le b_t \le \bar{\sigma}_{t-1}$, we have $\bar{\sigma}_{t-1}^2 - b_t^2 \in [0, \bar{\sigma}_{t-1}^2]$ with $\bar{\sigma}_{t-1} < \bar{\sigma}_t$, and hence $c_t \in (0,1]$ and $\nu_t = c_t \bar{\sigma}_t^2 \in \mathbb{R}_{++}$. For $\boldsymbol{q} \in \mathcal{S}$, we set $\Theta_{\boldsymbol{q}} := g_t + \frac{1}{2\nu_t}\|\cdot - \boldsymbol{q}\|^2$, so that $\operatorname{prox}_{\nu_t g_t}(\boldsymbol{q}) = \arg\min \Theta_{\boldsymbol{q}}$ by Eq. (1). Then

$$\Theta_{\boldsymbol{q}} = \Big(g_t + \tfrac{L_t}{2}\|\cdot\|^2\Big) + \frac{1}{2}\Big(\frac{1}{\nu_t} - L_t\Big)\|\cdot\|^2 - \frac{1}{\nu_t}\langle \cdot, \boldsymbol{q}\rangle + \frac{1}{2\nu_t}\|\boldsymbol{q}\|^2. \tag{B.1}$$

By (A1), $\nabla g_t + L_t \operatorname{Id}$ is monotone, so the first term of Eq. (B.1) is convex (Nesterov, 2004, Thm. 2.1.3). By (A2), $\nu_t^{-1} - L_t > 0$, so $\Theta_{\boldsymbol{q}}$ is strongly convex and has exactly one minimizer (Bauschke & Combettes, 2017, Cor. 11.17), i.e., $\operatorname{prox}_{\nu_t g_t}$ is single valued. Fermat's rule (Bauschke & Combettes, 2017, Thm. 16.3) and $\partial\Theta_{\boldsymbol{q}} = \{\nabla\Theta_{\boldsymbol{q}}\}$ (Bauschke & Combettes, 2017, Prop. 17.31) give

$$\boldsymbol{u} = \operatorname{prox}_{\nu_t g_t}(\boldsymbol{q}) \iff \nabla\Theta_{\boldsymbol{q}}(\boldsymbol{u}) = \boldsymbol{0} \iff \boldsymbol{u} = \boldsymbol{q} - \nu_t \nabla g_t(\boldsymbol{u}). \tag{B.2}$$

Step 2. We have $\sqrt{1-\bar{\alpha}_{t-1}-\omega_t^2}/\sqrt{\bar{\alpha}_{t-1}} = \sqrt{\bar{\sigma}_{t-1}^2 - b_t^2} = (1-c_t)\bar{\sigma}_t$ and $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) = (\bar{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_{0|t})/\bar{\sigma}_t$. Substituting them into Eq. (5) divided by $\sqrt{\bar{\alpha}_{t-1}}$ gives

$$\bar{\boldsymbol{v}}_{t-1} = (1-c_t)\,\bar{\boldsymbol{x}}_t + c_t\,\hat{\boldsymbol{x}}_{0|t} + b_t\boldsymbol{\varepsilon}'. \tag{B.3}$$

Let $\boldsymbol{r}_t^{\mathrm{net}} := -c_t\bar{\sigma}_t\big(\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\varepsilon}_t^\star\big)$. By the definition of $\hat{\boldsymbol{x}}_{0|t}$ and $c_t\bar{\sigma}_t\boldsymbol{\varepsilon}_t^\star = \nu_t\nabla g_t(\bar{\boldsymbol{x}}_t)$, we have

$$(1-c_t)\,\bar{\boldsymbol{x}}_t + c_t\,\hat{\boldsymbol{x}}_{0|t} = \bar{\boldsymbol{x}}_t - c_t\bar{\sigma}_t\,\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) = \bar{\boldsymbol{x}}_t - \nu_t\nabla g_t(\bar{\boldsymbol{x}}_t) + \boldsymbol{r}_t^{\mathrm{net}}. \tag{B.4}$$

Step 3. Let $\boldsymbol{w}_t := \operatorname{prox}_{\nu_t g_t}(\bar{\boldsymbol{x}}_t)$ and $\boldsymbol{r}_t^{\mathrm{fb}} := \bar{\boldsymbol{x}}_t - \nu_t\nabla g_t(\bar{\boldsymbol{x}}_t) - \boldsymbol{w}_t$. By the definition of $\boldsymbol{r}_t^{\mathrm{fb}}$, we have $\bar{\boldsymbol{x}}_t - \nu_t\nabla g_t(\bar{\boldsymbol{x}}_t) = \boldsymbol{w}_t + \boldsymbol{r}_t^{\mathrm{fb}}$, and substituting it into Eq. (B.4) and then into Eq. (B.3) yields Eq. (13) with $\boldsymbol{r}_t := \boldsymbol{r}_t^{\mathrm{fb}} + \boldsymbol{r}_t^{\mathrm{net}}$, so it suffices to bound $\|\boldsymbol{r}_t^{\mathrm{fb}}\|$. Eq. (B.2) with $\boldsymbol{q} = \bar{\boldsymbol{x}}_t$ gives $\boldsymbol{w}_t = \bar{\boldsymbol{x}}_t - \nu_t\nabla g_t(\boldsymbol{w}_t)$, and hence

$$\boldsymbol{r}_t^{\mathrm{fb}} = \nu_t\big(\nabla g_t(\boldsymbol{w}_t) - \nabla g_t(\bar{\boldsymbol{x}}_t)\big), \quad \|\boldsymbol{w}_t - \bar{\boldsymbol{x}}_t\| = \nu_t\|\nabla g_t(\boldsymbol{w}_t)\|. \tag{B.5}$$

Hence

$$\|\nabla g_t(\boldsymbol{w}_t)\| \overset{\text{(A1)}}{\le} \|\nabla g_t(\bar{\boldsymbol{x}}_t)\| + L_t\|\boldsymbol{w}_t - \bar{\boldsymbol{x}}_t\| \overset{\text{(B.5)}}{=} \|\nabla g_t(\bar{\boldsymbol{x}}_t)\| + \nu_t L_t\|\nabla g_t(\boldsymbol{w}_t)\|, \tag{B.6}$$

and

$$\|\boldsymbol{r}_t^{\mathrm{fb}}\| \overset{\text{(A1), (B.5)}}{\leq} \nu_t^2 L_t \|\nabla g_t(\boldsymbol{w}_t)\| \overset{\text{(B.6), (A2)}}{\leq} \frac{\nu_t^2 L_t}{1-\nu_t L_t} \|\nabla g_t(\bar{\boldsymbol{x}}_t)\|. \tag{B.7}$$

The triangle inequality $\|\boldsymbol{r}_t\| \leq \|\boldsymbol{r}_t^{\mathrm{fb}}\| + \|\boldsymbol{r}_t^{\mathrm{net}}\|$, Eq. (B.7), and $\|\boldsymbol{r}_t^{\mathrm{net}}\| = c_t \bar{\sigma}_t \|\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\varepsilon}_t^\star\|$ from the definition of $\boldsymbol{r}_t^{\mathrm{net}}$ in Step 2 give the bound on $\|\boldsymbol{r}_t\|$ in Proposition 1. □

C Algorithm

In Appendix C, we state Eq. 15 and the inner solver of Section 3.2 as pseudocode.

Algorithm 1: ABBS: annealed backward-backward splitting

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, I)$
for $t \leftarrow T$ **to** 1 **do**
  $\hat{\boldsymbol{\varepsilon}}_t \leftarrow \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$
  $\hat{\boldsymbol{x}}_{0|t} \leftarrow \big(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\varepsilon}}_t\big)/\sqrt{\bar{\alpha}_t}$
  prior prox: one reverse step, which approximates $\mathrm{prox}_{\nu_t g_t}$
    $\boldsymbol{\varepsilon}' \sim \mathcal{N}(\mathbf{0}, I)$
    $\boldsymbol{x}_{t-1} \leftarrow \texttt{ReverseStep}\big(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_{0|t}, \hat{\boldsymbol{\varepsilon}}_t, \boldsymbol{\varepsilon}'\big)$
  $s_t \leftarrow (1-\bar{\alpha}_t)/\bar{\alpha}_t$
  condition prox: inexact $\mathrm{prox}_{f_t/\rho_t}$ **(in practice, only for** $t \in \mathcal{G}$**)**
    $s_{t-1} \leftarrow (1-\bar{\alpha}_{t-1})/\bar{\alpha}_{t-1}$
    $\nu_t \leftarrow s_t - \sqrt{s_t\big(s_{t-1} - \omega_t^2/\bar{\alpha}_{t-1}\big)}$ ▷ Prop. 1
    $\rho_t \leftarrow \lambda/\nu_t$
    $\boldsymbol{u}_t \leftarrow \texttt{ConditionProx}\big(\hat{\boldsymbol{x}}_{0|t}, f_{\mathcal{S}}, \rho_t, K\big)$
    $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}_{t-1} + \sqrt{\bar{\alpha}_{t-1}}\big(\boldsymbol{u}_t - \hat{\boldsymbol{x}}_{0|t}\big)$
end
return $\boldsymbol{x}_0$

Algorithm 2: ConditionProx: inexact $\mathrm{prox}_{f_{\mathcal{S}}/\rho}(\hat{\boldsymbol{x}}_0)$

$\boldsymbol{u}^{(0)} \leftarrow \hat{\boldsymbol{x}}_0$; initialize the inner solver
for $k \leftarrow 0$ **to** $K-1$ **do**
  $\boldsymbol{e}^{(k)} \leftarrow \nabla f_{\mathcal{S}}\big(\boldsymbol{u}^{(k)}\big) + \rho\big(\boldsymbol{u}^{(k)} - \hat{\boldsymbol{x}}_0\big)$
  $\boldsymbol{u}^{(k+1)} \leftarrow \texttt{Update}\big(\boldsymbol{u}^{(k)}, \boldsymbol{e}^{(k)}\big)$
end
return $\boldsymbol{u}^{(K)}$

D Experimental details and additional results

In Appendix D, we state the settings that Section 4 leaves out, and we show the qualitative results. Appendix D.1 treats the four image tasks of Section 4.1, and Appendix D.2 treats the human-motion editing of Section 4.2. Table 4 summarizes the backbone, the number T of reverse steps, and the hyperparameters of the proposed method for all tasks, and both appendices refer to it. We implement all methods in PyTorch and run all experiments on a single NVIDIA H100 GPU.

D.1 Image tasks

Setup and evaluation. All methods use the sampler of Ho et al. (2020) with the T of Table 4. For ABBS, we use Adam (Kingma & Ba, 2015) as the inner solver of Algorithm 2, and we reset its moment estimates at the start of every reverse step. The operator $\mathcal{A}$ of Example 1(a) is a 61×61 Gaussian blur with kernel standard deviation 3.0 in Task (i) and bicubic $\times 4$ downsampling from 256×256 to 64×64 in Task (ii). In both cases the measurement is $\boldsymbol{y} = \mathcal{A}\boldsymbol{x} + \boldsymbol{n}$ with $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 I)$,

Table 4: Backbone, number T of reverse steps in sampling, and hyperparameters of the proposed method for the six settings of Section 4.

Task	$\mathcal{S}$	Backbone	T	λ	K
(i) Deblurring	$\mathcal{X}$	Choi et al. (2021)	1000	0.5	20
(ii) Super-resolution	$\mathcal{X}$	Choi et al. (2021)	1000	0.5	20
(iii) Segmentation map	$\mathcal{X}$	Yu et al. (2023)	100	1×10^{-2}	30
(iv) Style	$\mathcal{Z}$	SD v1.4 (Rombach et al., 2022)	100	5×10^{-2}	30
Path following	$\mathcal{X}$	MDM v2 (Tevet et al., 2023)	50	1×10^{-3}	30
Path following	$\mathcal{Z}$	MLD++ (Dai et al., 2025)	50	1×10^{-3}	30

$\sigma_y = 0.05$, on images normalized to $[0, 1]$. In Task (iii), we use the weights of $\mathcal{B}$ that Yu et al. (2023) use. In Task (iv), the feature map ψ of Example 1(c) returns the features of the third layer of the CLIP image encoder (Radford et al., 2021), as in Yu et al. (2023) and He et al. (2024). The CLIP image encoder in ψ is ViT-B/16, and G is the Gram matrix (Johnson et al., 2016) as defined by He et al. (2024). We evaluate Tasks (i) and (ii) on 1,000 images of the validation split of FFHQ, Task (iii) on 1,000 target parsing maps, and Task (iv) on the conditioning pairs formed by 20 text prompts from PartiPrompts (Yu et al., 2022) and 9 reference style images from WikiArt (Saleh & Elgammal, 2015). We compute FID following the protocols of Chung et al. (2023) for Tasks (i) and (ii) and of Yu et al. (2023) for Task (iii). The validity metric of Tasks (iii) and (iv) is $f_{\mathcal{S}}$ itself, evaluated at the generated sample, and hence uses the networks specified above. We use it because Yu et al. (2023) and He et al. (2024) report it for these two tasks. Following Ye et al. (2024), we compute the CLIP score of Task (iv) with CLIP ViT-B/32, which differs from the encoder in ψ, so that the CLIP score is not overestimated.

Effect of the number K of inner iterations. We examine how the number K of inner iterations acts on Task (iv). The number K does not change the subproblem in Eq. 12b, because $\rho_t = \lambda/\nu_t$ (see Section 3.2) does not depend on K. Instead, K determines how far the output $\bar{\boldsymbol{x}}_{t-1}$ of the condition prox departs from the stationarity condition of Eq. 12b. We fix $\lambda = 5 \times 10^{-2}$, i.e., the value of Table 2, and vary K over $\{1, 5, 10, 30, 50, 80\}$. Figure 3 shows the style score and the CLIP score. Most of the change in both scores occurs up to $K = 10$. Beyond $K = 30$, the effect of a further increase in K diminishes, whereas the cost grows in proportion to K. We therefore use $K = 30$ in Table 2 and Figure 2.

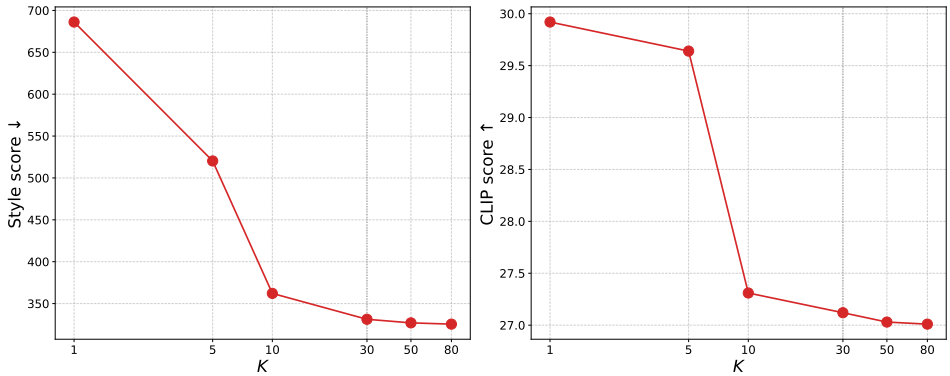

Figure 3: Style score (left) and CLIP score (right) versus the number K of inner iterations on the style guidance task, with $\lambda = 5 \times 10^{-2}$.

Computational cost. Table 5 reports the inference time per sample on Task (iv), together with the style and CLIP scores. We choose Task (iv) because each evaluation of its $\nabla f_{\mathcal{S}}$ is the most expensive among the four tasks, as it requires differentiating through both the decoder and the CLIP image encoder. ABBS with $K = 30$ takes longer than FreeDoM, MPGD, and TFG, and slightly

less time than ADMMDiff, while it improves on ADMMDiff in both scores. With $K = 10$, ABBS takes time comparable to TFG, and it improves on FreeDoM, MPGD, and TFG in both scores.

Table 5: Inference time per sample on the style guidance task (Task (iv)), measured on a single NVIDIA H100 GPU with a batch size of one. The style and CLIP scores are those of Table 2, except for ABBS ($K = 10$), whose scores are those of Figure 3.

Method	Style ↓	CLIP ↑	Time (s) ↓
FreeDoM	454.36	26.53	43.6
MPGD	435.16	25.96	30.4
TFG	370.70	27.07	55.7
ADMMDiff	350.17	26.77	110.3
ABBS ($K = 10$)	362.15	27.31	58.8
ABBS ($K = 30$)	331.25	27.12	105.2

Qualitative results. Figure 4 shows qualitative results on the two inverse problems, i.e., Tasks (i) and (ii), and Figure 5 shows those on segmentation-map guidance, i.e., Task (iii). Figure 6 shows the samples that correspond to each λ of Figure 2.

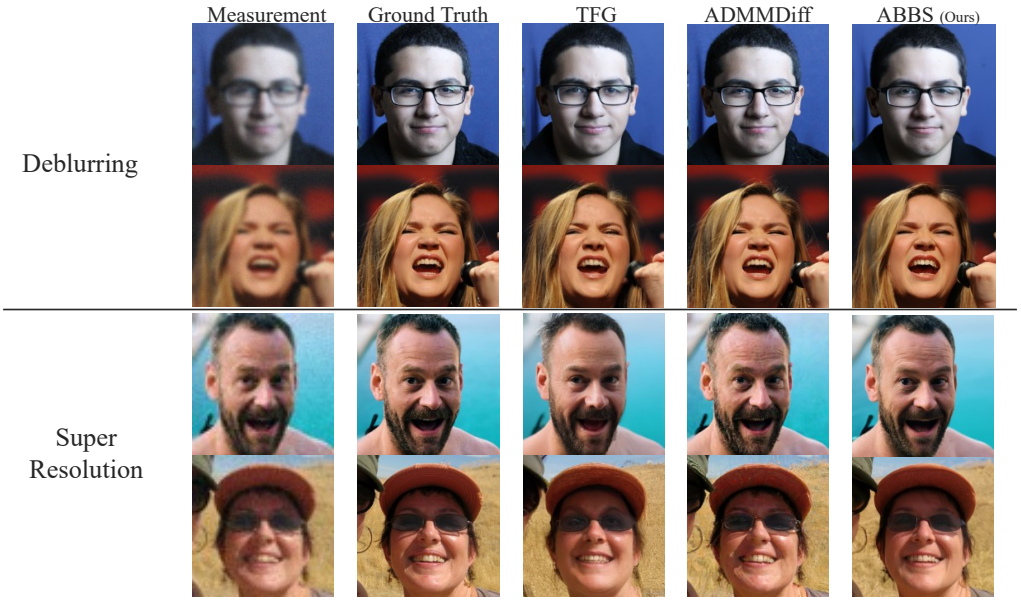

Figure 4: Qualitative comparison on inverse problems.

D.2 Human-motion editing

Setup and evaluation. HumanML3D (Guo et al., 2022) consists of 14,616 human motion sequences with 44,970 text descriptions, collected from AMASS (Mahmood et al., 2019) and HumanAct12 (Guo et al., 2020), and we evaluate on its test split following the evaluation protocol of Guo et al. (2022). In the trajectory control of Section 4.2, we write n and l for the indices of the joint and of the frame, respectively, and we let $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ denote the map that converts motion features, which include the local positions of the joints, to global absolute locations (e.g., Xie et al., 2024), so that $\mathcal{Y}$ is the space of the joint-location arrays $(\boldsymbol{y}_{nl})$. With $\Phi = \mathcal{R} \circ \mathcal{D}_\theta$, where $\mathcal{D}_\theta = \mathrm{Id}$ for ABBS-$\mathcal{X}$ by the convention of Example 1, the loss $f_{\mathcal{S}}$ of Problem 6 is instantiated as

$$f_{\mathcal{S}}(\boldsymbol{x}) = \sum_n \sum_l m_{nl} \left\| \mathcal{R}\big(\mathcal{D}_\theta(\boldsymbol{x})\big)_{nl} - \boldsymbol{y}_{nl} \right\|^2,$$

which is Eq. 7 with the entries that $\boldsymbol{y}$ leaves unspecified removed by the mask $m_{nl} \in \{0, 1\}$. Here $\boldsymbol{y}_{nl}$ is the target location of joint n at frame l, and $m_{nl} = 1$ if and only if $\boldsymbol{y}_{nl}$ is specified. As

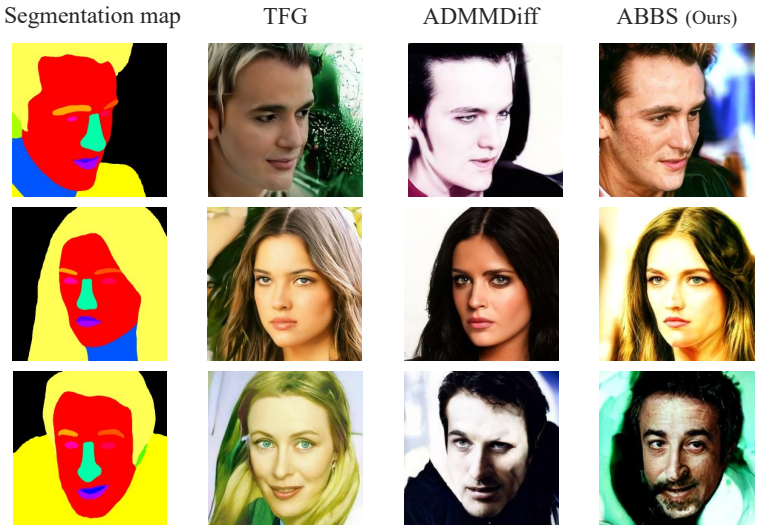

Figure 5: Qualitative comparison on segmentation map guidance.

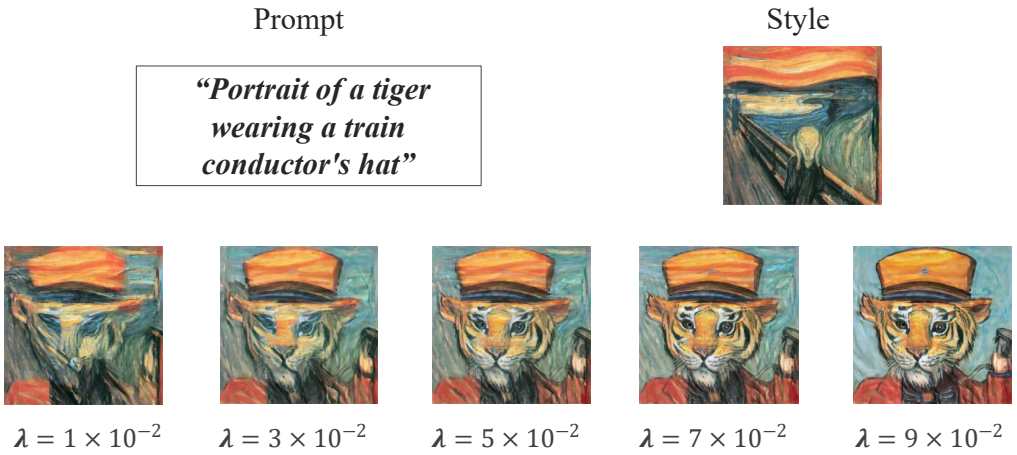

Figure 6: Examples of how λ changes the strength of guidance on the style guidance task.

in Appendix D.1, we use Adam as the inner solver of Algorithm 2 on both backbones. For each of the two backbones, we train a module of the ControlNet type (Zhang et al., 2023) with the same architecture and training settings as Xie et al. (2024) for ABBS-$\mathcal{X}$ and as Dai et al. (2024) for ABBS-$\mathcal{Z}$, where we apply the latter to MLD++. For editing performance, we report the three metrics of Xie et al. (2024), computed from the deviation of the generated pelvis location from its target at each frame, since $\boldsymbol{y}$ specifies the pelvis at every frame in path following. The trajectory error is the ratio of the generated motions that have at least one frame whose deviation exceeds a threshold, the location error is the ratio of the frames whose deviation exceeds that threshold, and the average error is the mean of the deviation over all frames, in meters. Table 3 reports the first two at the thresholds 20 cm and 50 cm. For generation quality, we report the R-precision (Top-3) and the FID of Guo et al. (2022), computed with the feature extractors of the same work. For inference cost, we report AITS, i.e., the average inference time per sample, measured on a single NVIDIA H100 GPU.

Contributions of the backbone, the training-based control module, and the training-free guidance. Tables 6 and 7 show where the results in Table 3 come from. Table 6 compares each method with its backbone alone, i.e., without the control module and without guidance. This comparison shows how much the backbone contributes to the generation quality in Table 3. Table 7 evaluates,

on each of the two backbones, the control module alone, ABBS alone, and their combination, i.e., ABBS-$\mathcal{X}$ or ABBS-$\mathcal{Z}$. This comparison separates the contribution of the control module from that of ABBS.

Table 6: Each method of Table 3 compared with its backbone alone. "w/o" is the backbone that generates from the text without control, i.e., without the control module and without guidance, and "w/" is the method with control, as reported in Table 3.

		AITS ↓		R-Precision ↑		FID ↓	
Methods	Backbone	w/o	w/	w/o	w/	w/o	w/
OmniControl	MDM	24.7	74.4	0.707	0.688	0.489	0.191
MotionLCM	MotionLCM	0.04	11.3	0.798	0.809	0.301	0.309
MaskControl	MoMask	0.09	62.3	0.807	0.807	0.045	0.070
ABBS-$\mathcal{X}$ (ours)	MDM v2	1.6	9.4	0.749	0.773	0.489	0.148
ABBS-$\mathcal{Z}$ (ours)	MLD++	0.02	0.2	0.797	0.810	0.081	0.177

Table 7: Contribution of the control module and of the guidance procedure on the path-following task. For each backbone, we use the control module alone, the guidance procedure of ABBS alone, or both. The row with both is the corresponding entry of Table 3.

Control module	Guidance	AITS ↓	Traj. err. (>20 cm) ↓	Traj. err. (>50 cm) ↓	Loc. err. (>20 cm) ↓	Loc. err. (>50 cm) ↓	Avg. err.↓	R-Precision Top-3 ↑	FID ↓
ABBS-$\mathcal{X}$ (ours; data space)									
✓	–	1.7	0.581	0.373	0.359	0.156	0.268	0.770	0.327
–	✓	9.1	0.033	0.006	0.007	0.001	0.008	0.705	0.511
✓	✓	9.4	0.008	0.000	0.000	0.000	0.004	0.773	0.148
ABBS-$\mathcal{Z}$ (ours; latent space)									
✓	–	0.03	0.524	0.343	0.340	0.138	0.253	0.815	0.146
–	✓	0.20	0.147	0.057	0.022	0.008	0.047	0.760	0.783
✓	✓	0.21	0.070	0.007	0.006	0.000	0.019	0.810	0.177

Qualitative results. Figure 7 shows qualitative examples on the path-following task.

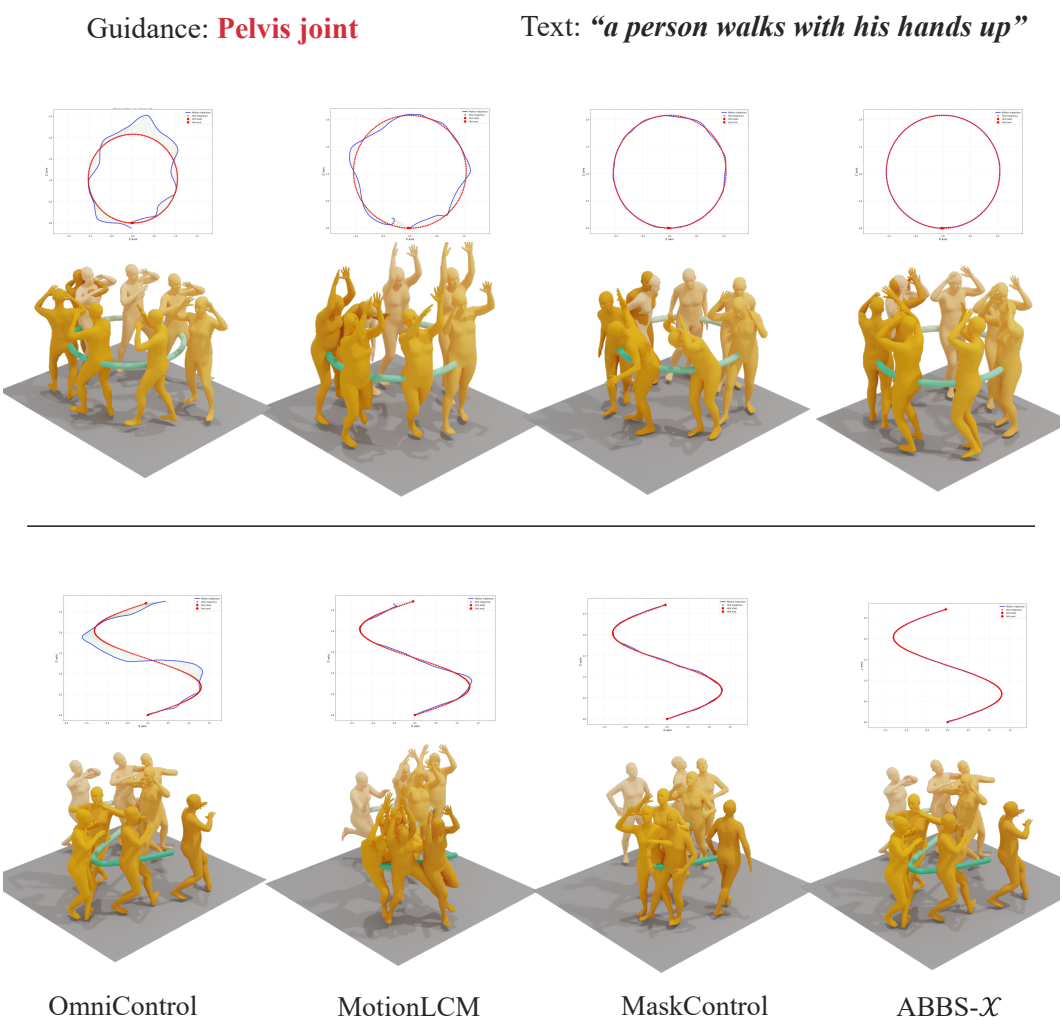

Figure 7: Qualitative comparison on path following. Each sample is generated from a text prompt, and the condition $\boldsymbol{y}$ in Section 4.2 specifies the pelvis location at every frame, i.e., $m_{nl} = 1$ for the pelvis and $m_{nl} = 0$ elsewhere. In each plot, the red curve is the target pelvis trajectory on the xz plane, i.e., the condition $\boldsymbol{y}$ itself, and the blue curve is the pelvis trajectory of the generated sample. The top row uses a circular target path and the bottom row a sinusoidal one. From left to right: OmniControl (Xie et al., 2024), MotionLCM (Dai et al., 2024), MaskControl (Pinyoanuntapong et al., 2025), and the proposed method.