FLOWBENDER: FEEDBACK-AWARE TRAINING FOR SELF-CORRECTING CONDITIONAL FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conditional diffusion and flow models routinely fail to satisfy the very constraints that define their task. For instance, a depth-conditioned model often produces images whose re-extracted depth disagrees with the input, even though the forward operator—the depth predictor defining the constraint—is available during both training and inference. Existing approaches generally fall into two categories: supervised models that treat the conditioning signal as a static cue and ignore alignment information at inference, and guidance-based methods that consult it through hand-tuned linear updates, typically trading fidelity to the condition against the plausibility of the generated sample. We argue that the fundamental gap in both paradigms is that the model is never trained to utilize its own alignment error. We introduce FlowBender, a closed-loop framework that treats this error as a first-class input, training the network to learn a correction policy conditioned on inference-time feedback. At each step, an unguided look-ahead pass estimates the clean signal, a task-specific deviation is computed via the forward operator, and a refinement pass consumes this signal to produce a corrected velocity. We propose several variants of FlowBender, including a gradient-based formulation for differentiable operators and a zero-order variant for non-differentiable settings such as JPEG compression. For efficient sampling, we introduce a prior-step shortcut that enables closed-loop correction at a minimal additional computational cost. Across image-to-image translation, restoration, and 3D mesh texturing, FlowBender consistently outperforms standard supervised baselines, alignment-loss-augmented training, and state-of-the-art inference-time guidance, improving fidelity and plausibility simultaneously rather than trading them against each other.

1 INTRODUCTION

Diffusion and Flow Matching (FM) models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Lipman et al., 2022) have become the dominant paradigm for visual generative modeling, and a primary use-case is *conditional* generation: producing samples $\mathbf{x}$ aligned with an external signal $\mathbf{y}$, $e.g.$, a text prompt, a depth map, or a geometric constraint. Effective conditional sampling requires both *fidelity* to $\mathbf{y}$ and *plausibility* with respect to the target data manifold. Existing methods, however, routinely fail on the first axis: for instance, a ControlNet (Zhang et al., 2023) conditioned on depth or edge maps often produces images whose re-extracted measurements disagree with the input (Fig. 3). This inconsistency occurs even though the forward operator $\mathcal{H}$ that relates the sample to the conditioning signal, such as a depth predictor, edge detector, or renderer, is typically available during both training and inference.

This failure mode reveals a missed opportunity. Supervised conditional models (Nichol et al., 2021; Saharia et al., 2022; Rombach et al., 2022; Zhang et al., 2023) treat $\mathbf{y}$ as a static cue and operate as *open-loop* systems at inference: even when the evolving sample drifts from the constraint, and even when the operator $\mathcal{H}$ is available to compute a deviation signal, the network has no built-in mechanism to consult this feedback and adjust the sampling trajectory. The very alignment information that motivates the task is left on the table.

Guidance-based methods (Dhariwal & Nichol, 2021; Chung et al., 2022a; Bansal et al., 2023; Patel et al., 2025) do consult $\mathcal{H}$ at inference, steering the trajectory using the gradient of a measurement-matching objective. However, these approaches utilize feedback only during inference, through a

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.

manually-tuned linear update rule which creates a fundamental train-test discrepancy and a delicate tuning problem: too little guidance fails to satisfy the constraint, too much pushes the trajectory off the data manifold (He et al., 2023; Ye et al., 2024). Figure 1 illustrates both failure modes on a 2D Archimedean spiral: standard conditional generation drifts across class boundaries (c), while inference-time guidance enforces the radial constraint at the cost of missing the data manifold (d).

We argue the fundamental gap in both paradigms is that the model is never *trained* to utilize its own alignment error. We close this gap by treating this error as a first-class input and training the model to learn a correction policy over it. Concretely, at each sampling step, the model first performs an unguided look-ahead pass to estimate the clean signal, from which we derive a task-specific deviation signal via the operator $\mathcal{H}$. A second refinement pass then consumes this error alongside the standard inputs and emits a corrected update — *bending* the trajectory toward the conditional manifold (Fig. 1(b), Fig. 2). We refer to our framework as **FlowBender** – a closed-loop system that self-corrects throughout sampling.

The framework has four notable properties. First, the learned correction is *not* guidance with better hyperparameters: orthogonal decomposition shows that 80% of the second-pass correction energy lies orthogonal to the gradient, indicating that the model exploits feedback through a non-linear policy that scalar-weighted schemes cannot express (§5.4). Second, FlowBender does not require the error signal to be a gradient: a *zero-order variant* feeds the raw measurement-space error directly to the network, extending correction to non-differentiable or black-box operators, such as JPEG compression and third-party APIs, where gradient guidance is inapplicable. Third, a *prior-step shortcut* allows reducing inference cost to as little as $N{+}1$ evaluations for N-step sampling, nearly matching open-loop efficiency while retaining corrective benefits (§4.5). Fourth, FlowBender retains the standard FM (equivalently, diffusion) training objective (§4.4), and thus its theoretical grounding: feedback is an additional input; it does not change what the model is trained to predict. This is in contrast to reward fine-tuning methods (Li et al., 2024; Clark et al., 2024), which modify the loss to maximize an external reward and are prone to reward hacking, often at the cost of plausibility.

Empirically, FlowBender consistently outperforms standard supervised training, alignment-loss-augmented training (Li et al., 2024), and state-of-the-art inference-time guidance (Patel et al., 2025) across image-to-image translation (super-resolution, depth/edge-to-RGB, JPEG restoration) and 3D mesh texturing. Remarkably, while our closed-loop framework is designed to enhance fidelity, it consistently improves plausibility as well (e.g., FID), in direct contrast to the fidelity–plausibility trade-off that manifests in traditional guidance approaches.

Our main contributions are:

- We present FlowBender, a closed-loop approach for conditional diffusion and FM models that replaces hand-tuned guidance updates with a learned policy over the model's own alignment error. The framework is architecturally agnostic and integrates with existing adapters such as ControlNet (Zhang et al., 2023) and LoRA (Hu et al., 2022).
- We include a zero-order variant that extends learned correction to non-differentiable or black-box operators, a regime where gradient-based guidance is inapplicable.
- We propose a prior-step shortcut for closed-loop correction at near-open-loop inference cost.
- We demonstrate simultaneous gains in fidelity and plausibility across image translation, restoration, and 3D texturing; analysis attributes these results to a learned correction policy that utilizes feedback non-linearly, in contrast to the rigid scalar-weighting of traditional guidance.

2 Related Work

Conditional Sampling via Open-Loop Training. Conditional diffusion and flow-matching models typically parameterize a score function or velocity field given a static conditioning signal $\mathbf{y}$. Specific controls are typically added to pre-trained backbones via adapters such as ControlNet (Zhang et al., 2023) or LoRA (Hu et al., 2022). Although theoretically sampling from the posterior $p(\mathbf{x} \mid \mathbf{y})$ (Batzolis et al., 2021), in practice these models often fail to satisfy conditioning constraints (Ho & Salimans, 2022; Karras et al., 2024; Sobol et al., 2024). We identify a limitation in treating $\mathbf{y}$ as a static hint: the model lacks a mechanism to evaluate or adjust its trajectory even though alignment diagnostics are often available during both training and inference. Enhanced training schemes have been proposed to improve conditional adherence (Li et al., 2024; Xu et al., 2024; Luo et al., 2024;

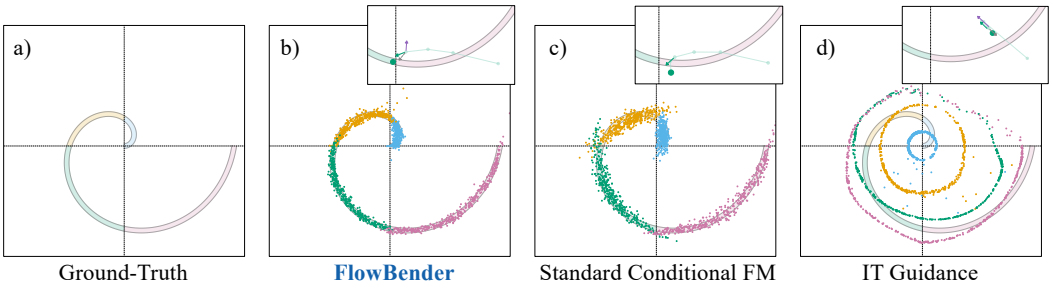


Figure 1: A conditional flow model is trained to sample from the 2D Archimedean spiral distribution shown in (a), which is partitioned by quadrant into four classes representing distinct radius ranges. (b–d) point colors denote the provided condition target classes. (b) **FlowBender** learns to internalize feedback from a radial guidance signal, achieving faithful alignment with both class constraints and the data manifold. (c) **Standard conditional generation** often violates class boundaries and misses the target distribution. (d) **Inference-time (IT) Guidance** satisfies radial constraints but drives samples off-manifold entirely. **Insets** showcase representative sampling trajectories for a green-class target, focusing on a specific velocity prediction at $t = 0.7$. The grey arrows (b, d) represent the first-pass or unconditional prediction, the purple arrow is the guidance component, and the green arrow is the final step taken. In (d), guidance dominates, pushing the final sample (large green dot) off-manifold. While the standard model (c) misses both the class and data distribution, our framework (b) learned to aggregate the components to arrive safely at the correct class within the data manifold. See Appendix A.1 for further details.

Elata et al., 2025), but they remain fundamentally open-loop: they do not provide the model with an estimate of its deviation from $\mathbf{y}$. By introducing alignment error as an explicit input, we transform the generation process into a closed-loop system capable of active self-correction.

Conditional Sampling via Bayesian Guidance. An alternative paradigm treats conditional sampling as Bayesian posterior inference, decomposing the conditional score into prior and likelihood terms:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y} \mid \mathbf{x}_t). \tag{1}$$

While the prior is typically approximated via a pre-trained denoiser, the likelihood term is defined by the intractable integral $p_t(\mathbf{y} \mid \mathbf{x}_t) = \int p(\mathbf{y} \mid \mathbf{x}_1) p(\mathbf{x}_1 \mid \mathbf{x}_t) \mathrm{d}\mathbf{x}_1$. Inference-time guidance methods (Daras et al., 2024; Kawar et al., 2021; 2022; Chung et al., 2022a;b; Song et al., 2023; Wang et al., 2022; Zhang et al., 2025; Bansal et al., 2023; Patel et al., 2025; Kim et al., 2025; Park & Ye, 2025) approximate it using a measurement-space distance between the predicted clean signal $\hat{\mathbf{x}}_1(\mathbf{x}_t)$ and $\mathbf{y}$. Although modular and free of per-task retraining, these approaches rely on *heuristic* weighting of the two terms, creating a trade-off between constraint satisfaction and sampling artifacts or divergence (He et al., 2023; Ye et al., 2024). Unlike methods that learn a scalar guidance weight (Galashov et al., 2025; Yehezkel et al., 2025), we provide the alignment error to the network as a first-class input, enabling it to learn a complex, non-linear policy that improves both conditional fidelity and sample plausibility rather than trading one for the other.

Learned Iterative Refinement. The paradigm of training neural networks to utilize error signals traces back to *learned optimizers*, where a network is trained to predict weight update rules based on gradient information from a base model (Andrychowicz et al., 2016; Wichrowska et al., 2017; Metz et al., 2020; Harrison et al., 2022; Metz et al., 2022). This approach was adapted for inverse problems to iteratively refine estimates by incorporating measurement-space reconstruction error as input (Carreira et al., 2016; Adler & Öktem, 2017; 2018; Li et al., 2018; Ma et al., 2020; Holzschuh & Thuerey, 2024). In computer vision, learned iterative refinement has advanced view synthesis (Flynn et al., 2019) and 3D scene reconstruction (Chen et al., 2024; Kang et al., 2025; Liu et al., 2025b; Yildirim et al., 2026; Liu et al., 2026; Chen et al., 2026; Long et al., 2026; Xu et al., 2025; Wen et al., 2025). By learning to leverage feedback from their own errors, these methods improve fidelity over feed-forward baselines while exceeding the efficiency of per-scene optimization.

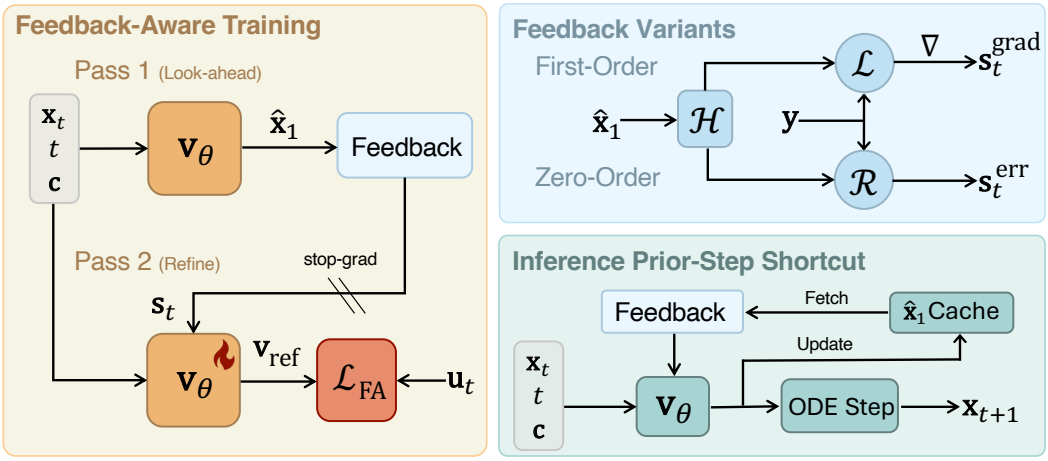


Figure 2: **FlowBender overview. (Left)** Training follows a two-pass strategy: a *look-ahead* pass produces a clean-signal estimate $\hat{\mathbf{x}}_1$ to compute the feedback signal $\mathbf{s}_t$, which then conditions a second *refinement* pass. **(Top-right)** Feedback variants include first-order gradients for differentiable operators and zero-order residuals for non-differentiable or black-box settings. **(Bottom-right)** At inference, an optional shortcut bypasses the look-ahead pass by fetching $\hat{\mathbf{x}}_1$ from a cached estimate of the previous step, significantly reducing computational overhead.

3 Preliminaries: Conditional Flow Matching Models

Flow Matching (FM) (Lipman et al., 2022) models a probability path $p_t(\mathbf{x}_t)$ that interpolates between a noise distribution $p_0(\mathbf{x}_0) \sim \mathcal{N}(0, \mathbf{I})$ and a target data distribution $p_1(\mathbf{x}_1)$. While the term *conditional flow matching* (CFM) often refers specifically to the training framework where paths are constructed relative to data points $\mathbf{x}_1$, we use it here to denote flow models that receive an external task-specific conditioning signal $\mathbf{c}$, *e.g.*, a corrupted image or depth map. The forward process $p_t(\mathbf{x}_t \mid \mathbf{x}_1)$ for $t \in [0, 1]$ is defined by the linear interpolation:

$$\mathbf{x}_t = a_t \mathbf{x}_1 + \sigma_t \mathbf{x}_0, \tag{2}$$

where a_t and σ_t are schedule coefficients. This framework typically involves training a network $\mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c})$ to approximate the vector field $\mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1) = \dot{a}_t \mathbf{x}_1 + \dot{\sigma}_t \mathbf{x}_0$ via the objective:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,(\mathbf{x}_1,\mathbf{c}),\mathbf{x}_0} \left[\|\mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}) - \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)\|^2 \right]. \tag{3}$$

Notably, $\mathbf{v}_\theta$ provides a principled mechanism to compute a point estimate of the clean signal, $\hat{\mathbf{x}}_1$, at any step t. For instance, in optimal transport FM ($a_t = t, \sigma_t = 1 - t$), the clean estimate is $\hat{\mathbf{x}}_1 = \mathbf{x}_t + (1 - t)\mathbf{v}_\theta$.

4 Feedback-Aware Conditional Flows

We propose FlowBender, a framework transforming conditional generation into a closed-loop system by learning to integrate alignment feedback (Fig. 2). In the following sections, we investigate various formulations of the feedback signal and describe how the model is trained to internalize them. Pseudocode for training and inference is provided in Algs. 1 and 2 (Appendix A).

4.1 Problem Formulation

Conditional generative modeling aims to sample from $p(\mathbf{x} \mid \mathbf{c})$, producing samples that satisfy two simultaneous objectives: **plausibility**, adhering to the target data manifold, and **fidelity**, maintaining consistency with the conditioning signal. Given a forward operator $\mathcal{H}$ accessible during training and inference, we define the observation $\mathbf{y} = \mathcal{H}(\mathbf{x})$. The condition $\mathbf{c}$ includes $\mathbf{y}$ and potentially auxiliary signals such as text. This formulation generalizes diverse tasks; for example, $\mathcal{H}$ may represent a renderer for 3D texturing, a depth predictor for depth-to-RGB, or a degradation operator for image restoration. In this context, the objective is to sample from the posterior distribution $p(\mathbf{x} \mid \mathbf{c}, \mathcal{H})$.

4.2 THE TWO-PASS FEEDBACK LOOP

The core mechanism of FlowBender relies on feeding a task-specific feedback signal, derived from the model's current alignment error, back into the network as a first-class input. However, obtaining the clean signal estimate $\hat{\mathbf{x}}_1$ required to compute this feedback $\mathbf{s}_t$ introduces a causal dependency: $\mathbf{s}_t$ must be provided as a model input, yet it depends on $\hat{\mathbf{x}}_1$, which is only available from the model's output. We resolve this via a **two-pass execution strategy** where a single network $\mathbf{v}_\theta$ handles both unguided and feedback-aware regimes. In the first pass, we set $\mathbf{s}_t = \mathbf{0}$ to generate a *look-ahead* velocity $\mathbf{v}_{\text{LA}}$ and derive the initial estimate $\hat{\mathbf{x}}_1$. This estimate allows for computing deviations from target constraints to define the feedback signal $\mathbf{s}_t$. In the second pass, the model consumes $\mathbf{s}_t$ alongside $(\mathbf{x}_t, t, \mathbf{c})$ to produce the final, refined velocity $\mathbf{v}_{\text{ref}}$.

4.3 FEEDBACK VARIANTS

We propose several formulations for the error signal $\mathbf{s}_t$. A key design principle is to provide $\mathbf{s}_t$ as an auxiliary input naturally supported by any model, rendering our framework architecture-agnostic.

First-Order Feedback. For differentiable $\mathcal{H}$, we define an alignment loss $\mathcal{L}(\mathcal{H}(\mathbf{x}), \mathbf{y})$ quantifying measurement-space discrepancy. The first-order feedback signal is derived from the gradient of this loss. While Bayesian formulations (Eq. 1) require gradients w.r.t. $\mathbf{x}_t$, stability and efficiency often favor approximations w.r.t. the estimated clean signal $\hat{\mathbf{x}}_1$ (He et al., 2023; Patel et al., 2025; Kim et al., 2025). We therefore evaluate two candidates for $\mathbf{s}_t^{\text{grad}}$: $\nabla_{\mathbf{x}_t} \mathcal{L}(\mathcal{H}(\hat{\mathbf{x}}_1), \mathbf{y})$ and the "shortcut" $\nabla_{\hat{\mathbf{x}}_1} \mathcal{L}(\mathcal{H}(\hat{\mathbf{x}}_1), \mathbf{y})$. The latter omits the expensive network Jacobian $\frac{\partial \hat{\mathbf{x}}_1}{\partial \mathbf{x}_t}$, reducing memory overhead. The resulting gradient is concatenated to $\mathbf{x}_t$ along the channel dimension, providing an explicit direction for error correction.

Zero-Order Feedback. We additionally consider a derivative-free feedback signal defined by a measurement-space error operator: $\mathbf{s}_t^{\text{err}} = \mathcal{R}(\mathcal{H}(\hat{\mathbf{x}}_1), \mathbf{y})$. $\mathcal{R}$ is task-specific and aligns with the domain of the conditioning signal $\mathbf{y}$, *e.g.*, a pixel-wise residual for images. In this variant, the residual $\mathbf{s}_t^{\text{err}}$ is provided as a conditional input alongside $\mathbf{y}$, forcing the model to learn a mapping from measurement-space errors to signal-space updates. Since no backpropagation through $\mathcal{H}$ is required, this approach improves efficiency and extends to non-differentiable operators, such as JPEG compression or physical simulations, and to black-box systems, such as proprietary APIs or feature extractors, where gradient-based guidance is inapplicable.

Hybrid Feedback. Finally, we consider a composite variant that incorporates both first- and zero-order feedback. In this configuration, the model receives the gradient $\mathbf{s}_t^{\text{grad}}$ and the residual $\mathbf{s}_t^{\text{err}}$ through their respective input channels.

4.4 TRAINING

We optimize the model parameters θ using a joint training paradigm that supports both unguided and feedback-aware modes. For a pair $(\mathbf{x}_1, \mathbf{c})$ and timestep t, we generate $\mathbf{x}_t$ following Eq. 2. First, the model computes a look-ahead estimate $\hat{\mathbf{x}}_1$ by setting the feedback input to zero ($\mathbf{s}_t = \mathbf{0}$). We then compute the feedback signal $\mathbf{s}_t$ from this estimate (Section 4.3). The training objective is:

$$\mathcal{L}_{\text{FA}} = \mathbb{E}[\|\mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}, \text{sg}[\mathbf{s}_t]) - \mathbf{u}_t\|^2],$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operation. To ensure numerical stability and efficiency, we treat $\mathbf{s}_t$ as a constant input rather than differentiating through the look-ahead pass and operator $\mathcal{H}$. Note that $\mathcal{L}_{\text{FA}}$ is the standard CFM regression loss: $\mathbf{s}_t$ enters only as an input to $\mathbf{v}_\theta$, leaving the target $\mathbf{u}_t$ and thus the objective's theoretical guarantees (Lipman et al., 2022) intact. To maintain look-ahead reliability, we randomly replace $\mathbf{s}_t$ with a null vector with probability p_{un}, similar to the training protocol of CFG (Ho & Salimans, 2022). This joint training ensures the model remains accurate in the unguided regime – essential for generating reliable feedback at inference time.

4.5 INFERENCE

Inference follows the two-pass procedure described in Section 4.2: at each step t, an unguided look-ahead pass estimates the clean signal $\hat{\mathbf{x}}_1$ to derive the feedback signal $\mathbf{s}_t$. A subsequent refinement pass then consumes $\mathbf{s}_t$ to yield the final velocity $\mathbf{v}_{\text{ref}}$ used to advance the ODE. We next describe optional modifications to this procedure that enable stronger alignment and more efficient sampling.

Optional CFG. Our approach enables Classifier-Free Guidance (CFG) (Ho & Salimans, 2022) at zero marginal cost. While our framework aims to move beyond heuristic-based sampling, users desiring manual control can elegantly leverage the unguided velocity $\mathbf{v}_{\text{LA}}$, already computed to derive the feedback signal, as the "unconditional" reference. We define the resulting velocity as: $\mathbf{v}_{\text{cfg}} = w \cdot \mathbf{v}_{\text{ref}} + (1 - w) \cdot \mathbf{v}_{\text{LA}}$, where w modulates alignment strength (see Fig. 5 and Appendix D).

Efficient Sampling via Prior-Step Feedback Approximation. The two-pass execution doubles model evaluations as each step requires both a look-ahead and a refinement pass. To alleviate this, we propose an optional shortcut that exploits the similarity of error signals across subsequent timesteps to approximate the feedback without an additional evaluation. Analysis indicates that the feedback derived from the *unguided* prediction at t is increasingly correlated with the feedback computed from the *guided* prediction of the preceding step as the trajectory approaches the clean manifold (see Fig. 6(a–b)). This suggests that the refined prediction from the previous step serves as an effective surrogate for the current unguided clean estimate, particularly in the later stages of generation. We introduce a threshold $t_{\text{thresh}} \in [0, 1]$ to control this approximation; for $t > t_{\text{thresh}}$, we bypass the look-ahead pass by deriving $\mathbf{s}_t$ from the cached estimate $\hat{\mathbf{x}}_1^{\text{prev}}$ from the prior step (Fig. 2, bottom-right). This threshold enables a controllable trade-off between train-test compatibility and sampling speed. While $t_{\text{thresh}} = 1$ preserves the two-pass logic, $t_{\text{thresh}} = 0$ transforms execution into a single-pass loop after an initial bootstrap step – requiring only $N+1$ evaluations for an N-step trajectory, nearly matching vanilla sampling efficiency (Appendix C) while retaining closed-loop corrective benefits.

5 Experiments

We evaluate FlowBender by fine-tuning pre-trained models for image-to-image translation and 3D mesh texturing. We conduct these evaluations using latent flow-matching models, as they represent the current state-of-the-art, though our method fits diffusion and signal-space models as well. We compare our closed-loop approach against three established paradigms for conditional sampling:

1. **Standard Fine-Tuning (Standard FT):** The conventional single-pass supervised approach for learning a conditional velocity field. We implement this using ControlNet (Zhang et al., 2023) and LoRA (Hu et al., 2022), two popular adapters designed to incorporate new conditions into pre-trained models while mitigating catastrophic forgetting.
2. **Fine-Tuning with Alignment Loss (FT + $\mathcal{L}_{\text{align}}$):** Methods that improve fidelity by modifying the training loss, adding an explicit measurement-consistency term (a *reward*). We use ControlNet++ (Li et al., 2024) as the representative baseline and compare against additional reward FT methods (Black et al., 2024; Prabhudesai et al., 2024; Clark et al., 2024) in Appendix D.1.
3. **Inference-Time Guidance (IT Guidance):** A test-time approach that steers the sampling trajectory using gradients of a consistency term, without retraining. Specifically, we compare against FlowChef (Patel et al., 2025), a recent state-of-the-art general guidance framework for FM, as the representative baseline and present comparisons with additional methods in Appendix D.1.

Experimental Protocol. Performance is evaluated across two primary axes: *fidelity*, defined by the alignment between the projected generated sample $\mathcal{H}(\mathbf{x})$ and the target measurement $\mathbf{y}$, and *plausibility*, which assesses adherence to the target data manifold (e.g., perceptual quality). To ensure a controlled comparison and isolate the impact of our feedback mechanism, primary evaluations utilize an identical number of sampling steps across all methods. We provide more extensive comparisons in Appendix D, including results for baselines with doubled sampling budgets, higher-order samplers, and the incorporation of CFG. As we demonstrate, such standard test-time enhancements do not resolve the fundamental shortcomings of existing paradigms.

5.1 Image-to-Image Translation

We evaluate FlowBender on four tasks spanning closed-form, neural-network, and non-differentiable operators: super-resolution ($8\times$ downsampling), depth-to-RGB (DepthAnythingV2 (Yang et al., 2024)), edge-to-RGB (HED (Xie & Tu, 2015)), and JPEG restoration ($\sigma = 10$). We fine-tune Stable Diffusion 3.5 Large (Esser et al., 2024) with ControlNet (Zhang et al., 2023) on Unsplash-25K (Ali et al., 2023); training and sampling settings are in Appendix A.2. We measure *fidelity* against the conditioning image with PSNR, SSIM, and LPIPS (restoration), MAE and MSE (edge), and MAE and $\delta_{1.25}$ (depth); we evaluate *plausibility* with FID (Heusel et al., 2017) throughout. As JPEG compression is non-differentiable, only Standard FT and our zero-order variant apply

Table 1: **Image-to-Image Results.** Shaded rows denote FlowBender variants. Best results in **bold**; second best underlined.

	Super Resolution				Depth-to-RGB			Edge-to-RGB		
	Fidelity			Plausibility	Fidelity		Plausibility	Fidelity		Plausibility
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	MAE ↓	$\delta_{1.25}$ ↑	FID ↓	MAE ↓	MSE ↓	FID ↓
Standard FT	34.35	96.88	0.83	3.93	0.0847	0.7886	18.21	0.0533	0.0137	13.98
FT + $\mathcal{L}_{\text{align}}$	35.21	97.53	0.79	4.11	0.0837	0.7930	17.57	0.0501	0.0128	14.47
IT Guidance	**43.02**	**98.29**	0.65	18.96	0.0866	0.7592	223.54	0.0416	0.0129	97.65
First-order (w.r.t. $\mathbf{x}_t$)	36.27	97.10	0.66	4.30	**0.0747**	**0.8268**	**14.57**	0.0460	0.0111	14.29
First-order (w.r.t. $\hat{\mathbf{x}}_1$)	39.21	97.64	0.43	3.83	0.0818	0.7973	15.89	0.0435	0.0123	14.97
Zero-order	39.25	98.18	**0.21**	**3.36**	0.0763	0.8188	15.70	0.0456	0.0104	14.81
Combined (w.r.t. $\mathbf{x}_t$)	39.95	98.25	**0.21**	3.40	0.0829	0.7949	15.93	0.0435	0.0097	13.91
Combined (w.r.t. $\hat{\mathbf{x}}_1$)	39.77	98.24	0.23	3.39	0.0783	0.8175	15.39	**0.0408**	**0.0085**	**13.68**

Table 2: **JPEG Restoration Results.**

	Fidelity			Plausibility
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Standard FT	26.29	79.45	22.24	4.35
Zero-order (Ours)	**28.86**	**83.13**	**16.33**	**3.80**

Table 3: **Ablation of the feedback signal** (super resolution).

Feedback	PSNR ↑	FID ↓
$\hat{\mathbf{x}}_1$ (self-cond.)	33.45	4.28
Zero-order (Ours)	**39.25**	**3.36**

there. Tables 1 and 2 summarize the results. FlowBender's variants improve over the baselines on both fidelity and plausibility across tasks. In particular, the combined variant (w.r.t. $\hat{\mathbf{x}}_1$) outperforms both *Standard FT* and *FT + $\mathcal{L}_{align}$* on every metric for every differentiable operator (Table 1), making it the most consistent configuration. While *IT Guidance* can be competitive on fidelity, it incurs a severe plausibility cost, which grows sharply on the neural-network operators (FID 223.54 and 97.65). *FT + $\mathcal{L}_{align}$* improves fidelity over *Standard FT*, but often trades away plausibility. Qualitative examples for edge- and depth-to-RGB appear in Fig. 3; further results, restoration examples, and discussion of the fidelity–plausibility trade-off are in Appendix D.1.

5.2 3D Mesh Texturing

We fine-tune the TRELLIS-2 (Xiang et al., 2025) texture transformer with LoRA adapters (Hu et al., 2022), conditioning on the latents of the 3D geometry and the image $\mathbf{y}$, and expanding the input channels to accept the feedback signal $\mathbf{s}_t$. The forward operator $\mathcal{H}$ composes the TRELLIS-2 latent decoder with its differentiable PBR renderer. We use first-order feedback, concatenating $\mathbf{s}_t$ with the noisy latents. We train on 7,500 Objaverse assets (Deitke et al., 2023) and evaluate on 100 held-out Objaverse and 100 Toys4K (Stojanov et al., 2021) assets, reporting fidelity as masked PSNR (M.PSNR, excluding background), SSIM, LPIPS, and CLIP-similarity to the conditioning image, and plausibility as FID and multi-view reconstruction metrics over 50 random renders per asset. Quantitative results in Table 4 show our $\nabla_{\hat{\mathbf{x}}_1}$ variant is best on every metric in both datasets, substantially improving over the supervised baselines, with our $\nabla_{\mathbf{x}_t}$ variant second on most. While IT Guidance enhances fidelity, it fails to match FlowBender in plausibility. Qualitative comparisons (Fig. 4) illustrate that our approach recovers fine-grained details omitted by baselines. Additional visualizations, including multi-view renderings, are in Appendix D.3.

5.3 Ablation Study

We ablate key design choices on super-resolution; further ablations are in Appendix D.2.

Inference-time Shortcut. We analyze the efficiency-performance trade-off controlled by t_{thresh} (Section 4.5). As shown in Fig. 6, reducing t_{thresh} decreases the computational cost toward $n + 1$ NFEs, approaching the speed of vanilla sampling at lower threshold values. Although the shortcut introduces a relative performance dip, our approach consistently outperforms the Standard FT baseline even at low t_{thresh} settings. The fidelity gap at $t_{\text{thresh}} = 0$ is especially notable. These results demonstrate that while our primary approach utilizes a two-pass logic, the framework can provide substantial corrective benefits even with minimal computational overhead compared to standard open-loop sampling.

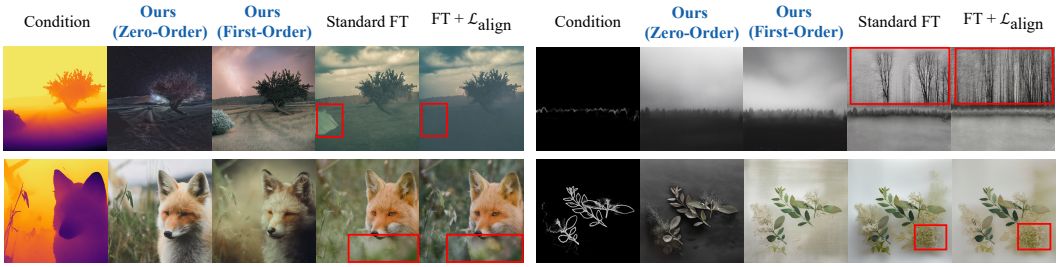


Figure 3: **Image-to-image.** (Left) Depth-to-RGB; (right) Edge-to-RGB. Red boxes highlight deviations from the conditioning image, which FlowBender avoids while maintaining visual quality.

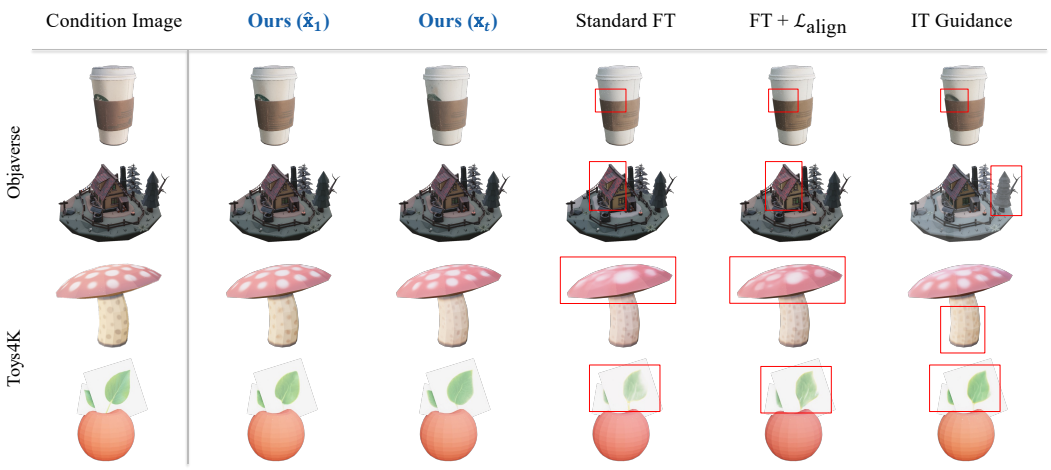


Figure 4: **3D texturing.** Objaverse (rows 1–2) and Toys4K (3–4) assets. The leftmost column provides the input condition image; remaining columns show generated 3D textured assets rendered from corresponding viewpoints. Red boxes highlight conditioning inconsistencies.

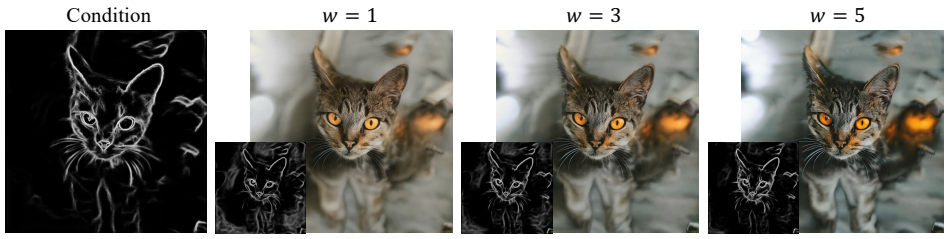


Figure 5: **Effect of Optional CFG.** Increasing guidance strength w can enhance condition fidelity. Insets show re-extracted edge maps.

Deviation as Feedback. Self-conditioning (Chen et al., 2022) feeds the model's own clean-sample prediction back into the network as an additional condition. It can be viewed as a variant of FlowBender in which the feedback never consults the measurement: the network receives $\hat{\mathbf{x}}_1$ itself rather than its deviation from $\mathbf{y}$ under $\mathcal{H}$. This ablation therefore isolates the contribution of the closed loop from that of the second pass alone. As shown in Table 3, a self-conditioning-inspired variant trails our zero-order variant by a wide margin, most notably in fidelity (PSNR). This suggests that the deviation information in the feedback is what drives the performance gain in our setting.

5.4 Is FlowBender Just Gradient Guidance in Disguise?

We investigate whether FlowBender training simply automates tuning for scalar-weighted guidance. Evaluating 180 velocity predictions across 20 assets in the texturing task ($\nabla_{\hat{\mathbf{x}}_1}$ variant), we decom-

Table 4: **Mesh Texturing Results.** Metrics evaluate single-view fidelity to the conditioning image and multi-view (MV) plausibility of the resulting 3D texture across Objaverse and Toys4K datasets. Shaded rows denote FlowBender variants. Best results in **bold**; second best underlined.

	Fidelity				Plausibility				
Method	M. PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	MV-M.PSNR ↑	MV-SSIM ↑	MV-LPIPS ↓	MV-CLIP ↑	FID ↓
Objaverse Dataset									
Pre-Trained	19.20	0.978	0.0221	0.961	19.91	0.987	0.0138	0.961	11.03
IT Guidance	25.86	0.986	0.0191	0.973	22.80	0.989	0.0123	0.968	9.10
Standard FT	21.91	0.981	0.0189	0.969	23.05	0.990	0.0115	0.970	8.74
FT + $\mathcal{L}_{\text{align}}$	22.63	0.983	0.0182	0.970	22.48	0.989	0.0117	0.969	8.94
Ours (w.r.t. $\mathbf{x}_t$)	25.26	0.985	0.0157	0.979	25.91	0.991	0.0105	0.974	7.41
Ours (w.r.t. $\hat{\mathbf{x}}_1$)	**26.39**	**0.987**	**0.0140**	**0.983**	**26.53**	**0.992**	**0.0098**	**0.977**	**6.64**
Toys4K Dataset									
Pre-Trained	20.36	0.984	0.0182	0.965	20.63	0.989	0.0129	0.966	10.66
IT Guidance	26.82	0.990	0.0137	0.982	23.63	0.992	0.0112	0.972	8.43
Standard FT	23.00	0.986	0.0160	0.975	24.38	0.991	0.0114	0.972	8.87
FT + $\mathcal{L}_{\text{align}}$	23.05	0.986	0.0159	0.976	23.25	0.991	0.0111	0.973	8.30
Ours (w.r.t. $\mathbf{x}_t$)	26.54	0.990	0.0129	0.980	27.20	0.993	0.0097	0.977	7.20
Ours (w.r.t. $\hat{\mathbf{x}}_1$)	**27.26**	**0.991**	**0.0116**	**0.984**	**27.21**	**0.993**	**0.0092**	**0.978**	**6.66**

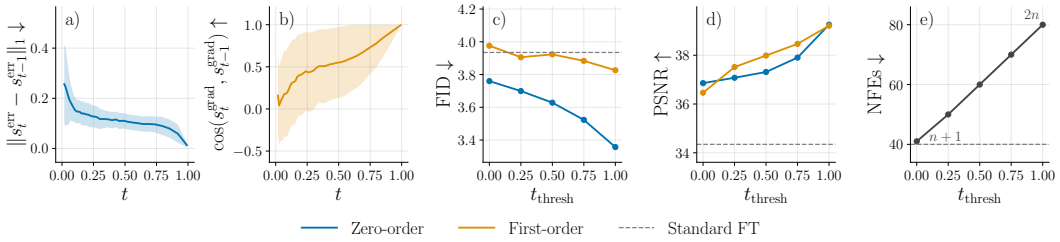


Figure 6: **Prior-Step Shortcut Analysis.** (a–b) Temporal similarity of feedback signals for zero-order (a) and first-order (b) variants; rising correlation as $t \to 1$ motivates the t_{thresh}-controlled shortcut strategy. (c–d) FID and PSNR vs. t_{thresh}; our method maintains a consistent advantage over Standard FT (dashed) even at low t_{thresh} values. (e) Computational cost (NFEs) as a function of t_{thresh}, where n is the number of sampling steps.

pose the learned correction $\Delta\mathbf{v} = \mathbf{v}_{\text{ref}} - \mathbf{v}_{\text{LA}}$ relative to the gradient $\mathbf{s}_t^{\text{grad}}$. The squared norm of the parallel component accounts for approximately 20% of the correction's energy; the majority of the correcting update resides in the component orthogonal to $\mathbf{s}_t^{\text{grad}}$. This relationship remains stable across noise levels ($t \in [0.1, 0.9]$). While a cosine similarity of $\cos(\Delta\mathbf{v}, \mathbf{s}_t^{\text{grad}}) = 0.42 \pm 0.11$ confirms the gradient is utilized, the dominant orthogonal component proves that FlowBender internalizes the gradient as a high-dimensional feature to non-linearly *bend* the flow toward the manifold, transcending the strict additive formulation of traditional Bayesian guidance.

6 Conclusion and Limitations

We presented FlowBender, a framework that addresses the fundamental "open-loop" limitation of existing conditional flow models by treating their own alignment errors as first-class inputs, thus training them to correct ("bend") their initial predictions. By replacing hand-tuned guidance with a learned, non-linear two-pass correction policy, FlowBender simultaneously enhances both conditional fidelity and sample plausibility. Our approach significantly improves over supervised and inference-time baselines across a diverse range of tasks, including image-to-image translation, restoration, and 3D mesh texturing.

Despite these gains, certain limitations remain. First, FlowBender trains a separate model per forward operator. While our models still perform well under operator shift (Appendix D.4), training a single model to handle a family of operators is a natural next step. Second, we assume access to $\mathcal{H}$ during both training and inference; extending the approach to blind settings, where the operator is unknown or partially specified, is a promising extension.

AI USE STATEMENT

We used LLM-based assistants as general-purpose coding assistants during implementation and experimentation, for writing assistance (clarity, concision, and LaTeX formatting), and for feedback on manuscript drafts. We did not use LLMs for research ideation, method design, or experiment design. All AI-assisted work was reviewed by the authors: generated code was verified and tested. 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 concerns conditional generative modeling of visual content. As with other generative models, the outputs may be misused, for example to produce misleading or non-consensual content, and generated content should not be treated as a faithful record of a real scene. Our models are fine-tuned from publicly released backbones on public datasets, and inherit their biases and licensing terms. This work involves no human subjects and releases no new data. We are not aware of other ethical concerns raised by this submission.

REPRODUCIBILITY STATEMENT

The method is described in full in Section 4, with the feedback variants in Section 4.3, the training objective in Section 4.4, and the inference procedure, including the prior-step shortcut, in Section 4.5; step-by-step pseudocode for both training and sampling is given in Algorithms 1 and 2. Implementation details sufficient to reproduce our experiments are provided in Appendix A: the 2D toy setup in Appendix A.1, and, for each real task, the architectural modifications used to inject the feedback signal, its normalization, the dataset construction and captioning pipeline, and all optimization hyperparameters, together with the hardware and GPU-hour cost of every run. The same appendix specifies the configuration of every baseline. All experiments build on publicly available pre-trained checkpoints and public datasets. Extended results covering every configuration we evaluated, beyond those reported in the main text, appear in Appendix D.

REFERENCES

Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

Jonas Adler and Ozan Öktem. Learned primal-dual reconstruction. *IEEE transactions on medical imaging*, 37(6):1322–1332, 2018.

Zahid Ali, Chesser Luke, and Carbone Timothy. Unsplash. `https://github.com/unsplash/datasets`, 2023.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

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.

Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4733–4742, 2016.

Tianyu Chen, Wei Xiang, Kang Han, Yu Lu, Di Wu, Gaowen Liu, and Ramana Rao Kompella. Gifsplat: Generative prior-guided iterative feed-forward 3d gaussian splatting from sparse views. *arXiv preprint arXiv:2602.22571*, 2026.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *European Conference on Computer Vision*, pp. 305–323. Springer, 2024.

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*, 2022a.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024.

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.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Noam Elata, Hyungjin Chung, Jong Chul Ye, Tomer Michaeli, and Michael Elad. Invfusion: Bridging supervised and zero-shot diffusion for inverse problems. *arXiv preprint arXiv:2504.01689*, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2367–2376, 2019.

Alexandre Galashov, Ashwini Pokle, Arnaud Doucet, Arthur Gretton, Mauricio Delbracio, and Valentin De Bortoli. Learn to guide your diffusion model. *arXiv preprint arXiv:2510.00815*, 2025.

James Harrison, Luke Metz, and Jascha Sohl-Dickstein. A closer look at learned optimization: Stability, robustness, and inductive biases. *Advances in neural information processing systems*, 35:3758–3773, 2022.

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.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Benjamin Holzschuh and Nils Thuerey. Flow matching for posterior inference with simulator feedback. *arXiv preprint arXiv:2410.22573*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Gyeongjin Kang, Seungtae Nam, Seungkwon Yang, Xiangyu Sun, Sameh Khamis, Abdelrahman Mohamed, and Eunbyung Park. ilrm: An iterative large 3d reconstruction model. *arXiv preprint arXiv:2507.23277*, 2025.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in neural information processing systems*, 34:21757–21769, 2021.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12328–12337, 2025.

Ming Li, Taojiannan Yang, Huafeng Kuang, Jie Wu, Zhaoning Wang, Xuefeng Xiao, and Chen Chen. Controlnet++: Improving conditional controls with efficient consistency feedback: Project page: liming-ai. github. io/controlnet_plus_plus. In *European Conference on Computer Vision*, pp. 129–147. Springer, 2024.

Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 683–698, 2018.

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

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*, 2025a.

Yueh-Cheng Liu, Lukas Höllein, Matthias Nießner, and Angela Dai. Quicksplat: Fast 3d surface reconstruction via learned gaussian initialization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27851–27861, 2025b.

Yueh-Cheng Liu, Jozef Hladkỳ, Matthias Nießner, and Angela Dai. Diff3r: Feed-forward 3d gaussian splatting with uncertainty-aware differentiable optimization. *arXiv preprint arXiv:2604.01030*, 2026.

Wei Long, Haifeng Wu, Shiyin Jiang, Jinhua Zhang, Xinchun Ji, and Shuhang Gu. Idesplat: Iterative depth probability estimation for generalizable 3d gaussian splatting. *arXiv preprint arXiv:2601.03824*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Grace Luo, Trevor Darrell, Oliver Wang, Dan B Goldman, and Aleksander Holynski. Readout guidance: Learning control from diffusion features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8217–8227, 2024.

Wei-Chiu Ma, Shenlong Wang, Jiayuan Gu, Sivabalan Manivasagam, Antonio Torralba, and Raquel Urtasun. Deep feedback inverse problem solver. In *European conference on computer vision*, pp. 229–246. Springer, 2020.

Yongjia Ma, Donglin Di, Xuan Liu, Xiaokai Chen, Lei Fan, Tonghua Su, and Yue Gao. Adams bashforth moulton solver for inversion and editing in rectified flow. *arXiv preprint arXiv:2503.16522*, 2025.

Luke Metz, Niru Maheswaranathan, C Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. *arXiv preprint arXiv:2009.11243*, 2020.

Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*, 2022.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8280–8290, 2022.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Jonghyun Park and Jong Chul Ye. Flowlps: Langevin-proximal sampling for flow-based inverse problem solvers. *arXiv preprint arXiv:2512.07150*, 2025.

Maitreya Patel, Song Wen, Dimitris N Metaxas, and Yezhou Yang. Flowchef: Steering of rectified flow models for controlled generations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15308–15318, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2024.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Ido Sobol, Chenfeng Xu, and Or Litany. Zero-to-hero: Enhancing zero-shot novel view synthesis via attention map filtering. *Advances in Neural Information Processing Systems*, 37:30522–30553, 2024.

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

Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1798–1808, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

Jing Wen, Alexander G Schwing, and Shenlong Wang. Life-gom: Generalizable human rendering with learned iterative feedback over multi-resolution gaussians-on-mesh. In *13th International Conference on Learning Representations, ICLR 2025*, pp. 40453–40472. International Conference on Learning Representations, ICLR, 2025.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *International conference on machine learning*, pp. 3751–3760. PMLR, 2017.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692*, 2025.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4818–4829, 2024.

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 1395–1403, 2015.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Resplat: Learning recurrent gaussian splats. *arXiv preprint arXiv:2510.08575*, 2025.

Yifeng Xu, Zhenliang He, Shiguang Shan, and Xilin Chen. Ctrlora: An extensible and efficient framework for controllable image generation. *arXiv preprint arXiv:2410.09400*, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in neural information processing systems*, 37:21875–21911, 2024.

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. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024.

Shai Yehezkel, Omer Dahary, Andrey Voynov, and Daniel Cohen-Or. Navigating with annealing guidance scale in diffusion space. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, 2025.

Ahmet Burak Yildirim, Tuna Saygin, Duygu Ceylan, and Aysegul Dundar. Geofusionlrm: Geometry-aware self-correction for consistent 3d reconstruction. In *Computer Graphics Forum*, pp. e70325. Wiley Online Library, 2026.

Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20895–20905, 2025.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023.

APPENDIX CONTENTS

A Implementation Details **16**

A.1 2D Toy Experiment 16

A.2 Image-to-Image Translation 17

A.3 3D Mesh Texturing 20

B Extended Discussion of Closely Related Methods **21**

C Inference-Time Cost **22**

D Additional Results **22**

D.1 Image-to-Image Translation 22

D.2 Additional Ablations 31

D.3 3D Mesh Texturing 32

D.4 Robustness to Operator Shift 36

A Implementation Details

Algorithm 1: Feedback-Aware Training

Require: Dataset $\mathcal{D}$, model $\mathbf{v}_\theta$, prob. p_{un}

while not converged **do**
  Sample $(\mathbf{x}_1, \mathbf{c}) \sim \mathcal{D}$ where $\mathbf{y} \in \mathbf{c}$
  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), t \sim \mathcal{U}[0, 1]$
  $\mathbf{x}_t \leftarrow a_t \mathbf{x}_1 + \sigma_t \mathbf{x}_0$
  $\mathbf{v}_{\text{LA}} \leftarrow \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{0})$ *// Pass 1: Unguided Look-ahead*
  $\hat{\mathbf{x}}_1 \leftarrow \text{EstimateSample}(\mathbf{x}_t, t, \mathbf{v}_{\text{LA}})$
  $\mathbf{s}_t \leftarrow \text{Feedback}(\hat{\mathbf{x}}_1, \mathbf{y})$ *// See Sec. 4.3*
  if $\text{rand}(0, 1) < p_{\text{un}}$ **then** *// Conditioning Dropout*
    $\mathbf{s}_{\text{in}} \leftarrow \mathbf{0}$
  else
    $\mathbf{s}_{\text{in}} \leftarrow \text{stop_grad}(\mathbf{s}_t)$
  $\mathbf{v}_{\text{ref}} \leftarrow \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{s}_{\text{in}})$ *// Pass 2: Refinement*
  $\mathcal{L}_{\text{FA}} \leftarrow \|\mathbf{v}_{\text{ref}} - \mathbf{u}_t\|^2$
  Update $\boldsymbol{\theta}$ via $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{FA}}$

Algorithm 2: Feedback-Aware Inference

Require: $\mathbf{v}_\theta$, condition $\mathbf{c}$ (incl. $\mathbf{y}$), t_{thresh}, ODE Solver

$\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$
$\hat{\mathbf{x}}_1^{\text{prev}} \leftarrow \mathbf{0}$
for $i = 0$ to $N - 1$ **do**
  $t \leftarrow t_i$
  if $t \le t_{\text{thresh}}$ **or** $i = 0$ **then** *// Pass 1: Unguided Look-ahead*
    $\mathbf{v}_{\text{LA}} \leftarrow \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{0})$
    $\hat{\mathbf{x}}_1 \leftarrow \text{EstimateSample}(\mathbf{x}_t, t, \mathbf{v}_{\text{LA}})$
  else *// Shortcut: Prior-Step Reuse*
    $\hat{\mathbf{x}}_1 \leftarrow \hat{\mathbf{x}}_1^{\text{prev}}$
  $\mathbf{s}_t \leftarrow \text{Feedback}(\hat{\mathbf{x}}_1, \mathbf{y})$
  $\mathbf{v}_{\text{ref}} \leftarrow \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{s}_t)$ *// Pass 2: Refinement*
  $\hat{\mathbf{x}}_1^{\text{prev}} \leftarrow \text{EstimateSample}(\mathbf{x}_t, t_i, \mathbf{v}_{\text{ref}})$ *// Cache Update*
  $\mathbf{x}_{t+1} \leftarrow \text{Step}(\mathbf{x}_t, \mathbf{v}_{\text{ref}}, t)$ *// Integration*
return $\mathbf{x}_N$

A.1 2D Toy Experiment

Ground-Truth Distribution. The target distribution is a 2D Archimedean spiral with finite thickness, defined by the curve

$$r = a\theta, \quad \theta \sim \text{Uniform}(0, 2\pi), \quad a = 2.0$$

The finite thickness of is modeled by an isotropic Gaussian spread around each point on the curve ($\sigma = 0.12$).

Each point is assigned a quadrant label $c \in \{0, 1, 2, 3\}$ based on which quarter-turn of the spiral it falls in ($c = \lfloor \theta/(\pi/2) \rfloor$).

Model Architecture. The generative model is an MLP with 3 linear layers, hidden dimension 64, and SiLU activations between layers. The input is a concatenation of the noisy sample $x_t \in \mathbb{R}^2$ with a time embedding. The scalar $t \in [0, 1]$ is lifted to a 16-dimensional sinusoidal embedding (Vaswani et al., 2017). The class condition is encoded via a learned embedding table that maps the quadrant label c to a vector of the hidden dimension, which is added to the time embedding before being passed to the main network.

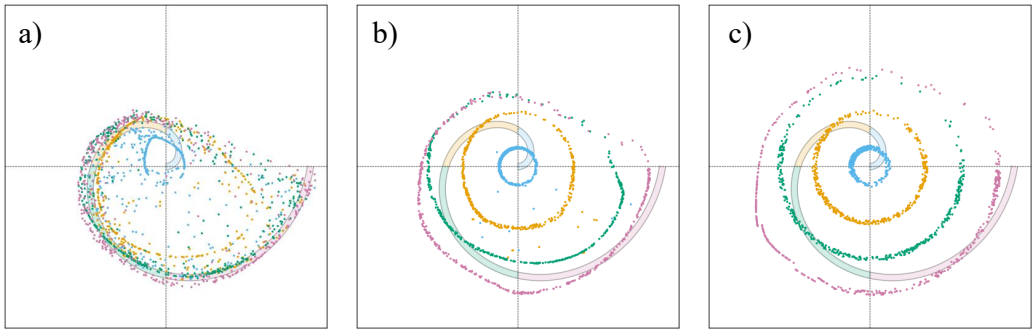


Figure 7: **Effect of guidance scale on conditional generation.** We compare guidance scales of a) 0.5, b) 2.0 (used in Fig. 1), and c) 4.0. Small scale (a) fails to enforce the condition, while large guidance (c) pushes samples off the data manifold entirely. We therefore selected an intermediate scale that provides a reasonable trade-off between fidelity and plausibility.

Training. All models are trained with the standard conditional flow matching objective (Lipman et al., 2022) for 50,000 steps with a batch size of 1024 and a learning rate of 3×10^{-4} (AdamW optimizer). Training samples are generated on the fly at each iteration. Prior to training, a z-score normalizer is fitted on 200,000 spiral samples (per-coordinate mean and standard deviation). All inputs and model outputs are normalized using this transformation.

Feedback and Guidance objective The spiral is divided into four quadrants, each assigned a non-overlapping arc-length band. For quadrant c, the target radius interval is:

$$r^c_{\min} = c\pi, \qquad r^c_{\max} = (c+1)\pi$$

The loss penalises points outside this band:

$$\mathcal{L}(x, c) = \text{softplus}(r^c_{\min} - r) + \text{softplus}(r - r^c_{\max}), \quad r = |x|_2$$

This is zero (up to softplus smoothing) when $r \in [r^c_{\min}, r^c_{\max}]$, and grows linearly outside.

Guidance Scale. As illustrated in Fig. 1, Training-Free Guidance tends to push the final samples off the data manifold. We explore different guidance scales in Fig. 7. Small guidance scales fail to enforce the conditioning signal, while large guidance scales lead to samples that deviate from the target manifold.

A.2 Image-to-Image Translation

We use the official Stable Diffusion 3.5 checkpoint implemented in the diffusers (von Platen et al., 2022) library and adapt their reference script to add and train the ControlNets [1].

The model is conditioned on $\mathbf{c} = (\mathbf{y}, \mathbf{c}_{\text{text}})$ where $\mathbf{y}$ is the image-shaped condition (e.g., edges) and $\mathbf{c}_{\text{text}}$ is the text condition. In the following, we omit the dependence on the text condition $\mathbf{c}_{\text{text}}$ for clarity.

Zero-order. To inject zero-order information, we expand the input convolution of the ControlNet and concatenate the residual to the condition along the channel dimension. Specifically, the ControlNet module CN computes the condition in the unguided pass from the image condition $\mathbf{y}$ as

$$\hat{\mathbf{c}} = \text{CN}([\mathbf{y}, \mathbf{0}]), \quad \mathbf{y}, \mathbf{0} \in \mathbb{R}^{3\times H\times W}$$

where $[\cdot]$ denotes concatenation along the channel dimension.

[1] https://github.com/huggingface/diffusers/blob/main/examples/controlnet/train_controlnet_sd3.py

We then compute the residual via

$$\begin{aligned}\mathbf{v}_{\text{LA}} &= \mathbf{v}_\theta(\mathbf{x}_t, t, \hat{\mathbf{c}}) \\ \hat{\mathbf{x}}_1 &= \text{EstimateSample}(\mathbf{x}_t, t, \mathbf{v}_{\text{LA}}) \\ \tilde{\mathbf{s}}_t &= |\mathbf{y} - \mathcal{H}(\hat{\mathbf{x}}_1)|\end{aligned}$$

where $\mathcal{H}$ is the composition of the VAE decoder and the pixel-space forward operator. We normalize the feedback signal using

$$\mathbf{s}_t = \tilde{\mathbf{s}}_t / \max(|\tilde{\mathbf{s}}_t|)$$

Finally, the ControlNet encodes the guidance signal $\tilde{\mathbf{c}} = \text{CN}([\mathbf{y}, \mathbf{s}_t])$ and the refined velocity is

$$\mathbf{v}_{\text{ref}} = \mathbf{v}_\theta(\mathbf{x}_t, t, \tilde{\mathbf{c}}).$$

First-order. For our first-order variant, we double the number of input channels of the input convolution of the DiT to inject the feedback signal. In the unguided pass, the denoising network $\mathbf{v}_\theta$ predicts

$$\mathbf{v}_{\text{LA}} = \mathbf{v}_\theta([\mathbf{x}_t, \mathbf{0}], t, \tilde{\mathbf{c}}), \quad \mathbf{x}_t, \mathbf{0} \in \mathbb{R}^{16\times h\times w}$$

where h, w are the height and width of the VAE-encoded image latents and $\tilde{\mathbf{c}} = \text{CN}(\mathbf{y})$.

We then compute the guidance signal $\tilde{\mathbf{s}}_t = \nabla_{\mathbf{x}_t}\mathcal{L}$ where

$$\begin{aligned}\hat{\mathbf{x}}_1 &= \text{EstimateSample}(\mathbf{x}_t, t, \mathbf{v}_{\text{LA}}) \\ \mathcal{L} &= \text{MSE}(\mathbf{y}, \mathcal{H}(\hat{\mathbf{x}}_1)))\end{aligned}$$

We standardize the feedback signal

$$\mathbf{s}_t = (\tilde{\mathbf{s}}_t - \text{mean}(\tilde{\mathbf{s}}_t)) \,/\, \text{std}(\tilde{\mathbf{s}}_t)$$

before computing the refined velocity as

$$\mathbf{v}_{\text{ref}} = \mathbf{v}_\theta([\mathbf{x}_t, \mathbf{s}_t], t, \tilde{\mathbf{c}}).$$

Inspired by Patel et al. (2025), we additionally explore a variant that uses $\tilde{\mathbf{s}}_t = \nabla_{\hat{\mathbf{x}}_1}\mathcal{L}$ which avoids differentiating through the denoising network $\mathbf{v}_\theta$.

Training We center crop and resize images of the Unsplash-25K (Ali et al., 2023) dataset (20k training and 5k test images) to $1,024^2$ resolution and create text conditions for images using Florence2-Large (Xiao et al., 2024). Training uses batch size 16 for 5 epochs, resulting in a total of $6,250$ optimizer steps, using the AdamW (Loshchilov & Hutter, 2017) optimizer with learning rate 10^{-5}, weight decay 0.01, and 500 steps of learning rate warmup. Similar to Zhang et al. (2023), the weights corresponding to the additional channels, used to inject feedback information, are zero-initialized such that the network is slowly adapted to make use of feedback during training. We drop the feedback condition during training with probability $p_{un} = 0.1$ (See ablation at Table 16).

We only train newly added parameters, specifically the ControlNet, while keeping weights of the base model frozen. In the first-order and combined variants, we additionally unfreeze the input convolution to the DiT (Peebles & Xie, 2023) to allow the model to make use of the feedback signal that is concatenated to the input latent.

All methods use NVIDIA A100 GPUs for training where zero-order experiments take ≈ 51 GPU hours, first-order experiments take ≈ 61 GPU hours, and Standard FT and FT+$\mathcal{L}_{\text{align}}$ runs take ≈ 38 and ≈ 48 GPU hours, respectively. For the training-compute-matched comparison in Table 13, the Standard FT and FT+$\mathcal{L}_{\text{align}}$ baselines are trained for 10 instead of 5 epochs with all other hyperparameters unchanged (including the learning-rate schedule, stretched to the longer run), and the final checkpoint is evaluated.

Inference. We use 40 Euler steps with full two-pass execution at every step ($t_{\text{thresh}} = 1$). No CFG is applied in the main text comparisons (guidance scale 1.0), consistent with all other evaluated methods; for CFG results, see extended tables in Section D.

Table 5: Ablation of $t_{\min}$ parameter in $\mathcal{L}_{\text{align}}$ baseline on depth-to-rgb task.

	Fidelity		Plausibility
$t_{\min}$	MAE ↓	$\delta_{1.25}$ ↑	FID ↓
0.8	0.0837	0.7930	17.57
0.5	0.0799	0.7854	21.31

Baselines. *Standard FT.* Baseline fine-tuning variants use the vanilla ControlNet setup with zero-convolutions to slowly expose the network to the condition during training.

FT + $\mathcal{L}_{align}$. The variants with $\mathcal{L}_{\text{align}}$ adopt the set up in Li et al. (2024) with loss weight $\lambda_{\text{align}} = 0.5$ for the consistency objective in addition to the standard flow matching loss. We use the same forward operator used in our other experiments to compute the consistency loss. Further, similarly to Li et al. (2024), the loss is only applied on timesteps $t > t_{\min}$ such that the consistency loss is not computed on low-quality point estimates during the initial denoising steps. While rectified flow models produce stable estimates relatively early in the denoising chain, we find in Table 5 that higher $t_{\min}$ leads to better results, similar to the findings in Li et al. (2024).

IT Guidance. Inference-time guidance methods directly update the evolving sample using gradient descent on the measurement-space distance function $\mathcal{L} = \text{MSE}(\mathcal{H}(\hat{\mathbf{x}}_1), \mathbf{y})$. We follow the FlowChef approach (Patel et al., 2025) to avoid differentiating through the denoising model by computing the gradient w.r.t. $\hat{\mathbf{x}}_1$ instead of $\mathbf{x}_t$.

Additional inference-time guidance baselines. Beyond FlowChef, we additionally compare against FlowDPS (Kim et al., 2025), LatentDAPS (Zhang et al., 2025), and FlowLPS (Park & Ye, 2025) on the super-resolution task (Table 9). All three are run on the same SD3.5 backbone without any fine-tuning. For each method we grid-search its main hyperparameters on a 20-image subset of the test set and report the configuration that minimizes full-reference LPIPS against the ground-truth image, i.e. each method's own best-case operating point for perceptual quality. For FlowDPS we sweep the step size $\in \{5, 10, 15, 30, 45, 60\}$ (paper default 15), the guidance scale $\in \{2.0, 3.0, 4.5\}$, and the number of sampling steps $\in \{28, 40\}$; the selected configuration is step size 30, guidance scale 2.0, and 28 steps. For LatentDAPS we sweep the learning rate $\in \{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}\}$, the guidance scale $\in \{2.0, 4.5\}$, and the annealing schedule $\in \{28{\times}2, 28{\times}5, 50{\times}2, 50{\times}5\}$ (annealing steps $\times$ ODE evaluations per step); the selected configuration is learning rate 10^{-5}, guidance scale 4.5, and a $50{\times}2$ schedule. For FlowLPS we sweep the proximal learning rate $\in \{0.05, 0.1, 0.25, 0.5\}$ and the number of Langevin steps per reverse step $\in \{0, 4, 7\}$ at the paper-default 40 sampling steps and guidance scale 2.0; the selected configuration is proximal learning rate 0.1 with 7 Langevin steps.

Higher-order samplers. In Tables 9 to 12 we additionally evaluate the Standard FT and Standard FT + $\mathcal{L}_{\text{align}}$ baselines with the RF-Solver (Wang et al., 2024) (second-order Taylor expansion of the rectified-flow ODE with a half-step finite-difference estimate of the velocity derivative) and the ABM-Solver (Ma et al., 2025) (Adams–Bashforth predictor with Adams–Moulton corrector in PECE mode; since SD3.5's shifted σ grid is non-uniform we recompute the coefficients for the actual grid and ramp the order from 1 to 4 over the first steps). Both solvers use 2 function evaluations per sampling step; at 40 solver steps they match FlowBender's 80-NFE two-pass budget.

Reward-based post-training. As detailed in Section D.1, we additionally compare against three post-training reward-based baselines: DDPO (Black et al., 2024), DRaFT (Clark et al., 2024), and AlignProp (Prabhudesai et al., 2024), three methods that fine-tune the Standard FT baseline ControlNet using a reward derived from the same consistency signal that underlies our feedback; the reward is the negative per-sample MSE between the forward operator applied to the decoded sample and the input condition. All three train rank-32 LoRA adapters ($\alpha = 32$) on the attention query, key, value, and output projections of the frozen ControlNet, initialized from the Standard FT checkpoints, with the SD3.5 transformer and VAE kept frozen. DDPO treats sampling as a multi-step MDP and optimizes it with PPO; since our backbone uses deterministic flow-matching sampling, we convert it to an equivalent SDE following Flow-GRPO (Liu et al., 2025a) (noise level $\in \{0.7, 1.0\}$) to obtain tractable per-step log-probabilities. Following the reference implementation, each round samples 256 trajectories of 40 steps and performs 2 PPO updates at an effective batch size of 128 with clip range 10^{-3}, for a budget of 200 rounds. DRaFT and AlignProp, instead, backpropagate the reward

directly through a truncated 20-step denoising chain. DRaFT-K differentiates through the last K steps – we evaluate both DRaFT-5 and the paper's best DRaFT-1-LV variant (with two additional low-variance re-noised gradient samples). AlignProp resamples the truncation point uniformly per batch. Both are trained for up to 1,000 steps. For all three methods we sweep the learning rate $\in \{3 \cdot 10^{-4}, 10^{-4}, 2 \cdot 10^{-5}\}$ and keep all checkpoints, evaluating every 50 optimizer steps (DRaFT, AlignProp) or every 10 rounds (DDPO), since the consistency reward is a distortion objective that starts to blur samples (i.e. reward-hack) within a few hundred steps, visualized in Fig. 12. Because DDPO only requires a black-box scalar reward, it is the only one of the three applicable to JPEG restoration, whose compression operator is non-differentiable (Table 11). For super-resolution and depth we report the checkpoint with the best LPIPS compared to the GT, used as a proxy to catch reward hacking before it shows up in FID; in Table 8 we report both the best-reward and the best-LPIPS checkpoint for edge-to-RGB to make the reward-hacking failure mode explicit.

A.3 3D Mesh Texturing

Data Preparation. We follow the original TRELLIS-2 data preparation pipeline (Xiang et al., 2025) with two notable modifications. First, conditioning images are rendered using TRELLIS-2's differentiable PBR renderer rather than Blender. This ensures operator consistency: the ground-truth observation $\mathbf{y}$ in the dataset is produced by the exact same forward operator $\mathcal{H}$ that computes the feedback term during training and inference. Second, we use a fixed lighting for all assets to aid convergence in our relatively small-data regime.

Architecture. We adapt the TRELLIS-2 texture flow model, a sparse DiT operating on 32-channel PBR texture latents, by attaching LoRA adapters (rank 128, $\alpha = 128$, dropout 0.05) to all linear layers. To inject the first-order gradient feedback $\mathbf{s}_t$, we expand the model's input projection from $2d$ to $3d$ channels, where $d = 32$ is the latent dimension, zero-initializing the additional block so the network initially behaves identically to the unguided baseline. In the look-ahead pass the gradient slot is set to $\mathbf{0}$; the refinement pass receives $\mathbf{s}_t$.

Feedback Signal. We use the first-order variant with gradients $\tilde{\mathbf{s}}_t$ taken with respect to either $\mathbf{x}_t$ or the clean estimate $\hat{\mathbf{x}}_1$. The forward operator $\mathcal{H}$ is the composition of the frozen TRELLIS-2 PBR texture decoder and the differentiable split-sum PBR renderer (nvdiffrec (Munkberg et al., 2022)), and $\mathcal{L}$ is the sum-reduced MSE between the rendered output and the conditioning image. The signal is normalized per-sample by its standard deviation,

$$\mathbf{s}_t = \tilde{\mathbf{s}}_t \;/\; \mathrm{std}(\tilde{\mathbf{s}}_t),$$

Training. We fine-tune the LoRA layers and the expanded input layer on 7,500 Objaverse assets, filtered by aesthetic score ≥ 4.5 and a maximum token count of 8,192 sparse voxels. Training runs for 25,000 steps with a batch size of 16. We use AdamW (Loshchilov & Hutter, 2017) with learning rate 10^{-4}, weight decay 0.01, and $\beta = (0.9, 0.95)$. Training uses bfloat16 automatic mixed precision. Gradients are clipped adaptively at the 95^{th} percentile with maximum norm 1.0. The null-feedback probability is $p_{\text{un}} = 0.1$. Image conditioning uses a DINOv3-ViT-L/16 feature extractor (Siméoni et al., 2025) at resolution 512. Training takes approximately 34 hours on 4 NVIDIA RTX PRO 6000 GPUs.

Inference. We use the default TRELLIS-2 sampler: 12 Euler steps with full two-pass execution at every step ($t_{\text{thresh}} = 1$). No CFG is applied in the main text comparisons (guidance scale 1.0), consistent with all other evaluated methods.

Baselines. *Standard FT* uses identical LoRA architecture (rank 128, $\alpha = 128$, dropout 0.05) and the same training configuration as our method, using the standard flow matching objective. CFG dropout (for results shown in Tables 17, 18) is set to 0.1.

FT + $\mathcal{L}_{align}$ extends Standard FT with an additional render consistency loss, following the ControlNet++ (Li et al., 2024) protocol. The loss weight is $\lambda_{\text{align}} = 5 \times 10^{-3}$ and the loss is applied only for $t > 0.3$, where the clean-signal estimate is sufficiently reliable; all other hyperparameters are shared with Standard FT.

IT Guidance (FlowChef (Patel et al., 2025)) is applied directly to the pretrained TRELLIS-2 model without any fine-tuning. At each of the 12 ODE steps, one SGD update with learning rate $\lambda = 0.1$ is applied to the noisy latent $\mathbf{x}_t$ using the MSE render loss gradient, following the official implementation defaults.

Results for baselines with additional configurations, including number of sampling steps, CFG, consistency loss weight λ_{align}, and guidance learning rate λ can be seen in Tables 17, 18.

B Extended Discussion of Closely Related Methods

We expand here on the two methods that share the most with FlowBender at a mechanistic level — both supply a network with information derived from a forward operator — and explain where the designs diverge.

InvFusion. InvFusion (Elata et al., 2025) integrates the forward operator directly into the network architecture through a "Feature Degradation Layer", which applies the operator to the internal activations of each denoiser block and compares the result against the measurement. The architectural choice comes at a cost in generality, however. The layer maps activations back to feature space through the pseudo-inverse $\mathbf{H}^\dagger$, which restricts it to linear operators, and it requires the network's deep feature resolution to match the input image dimension, which ties it to pixel-space models. These constraints exclude the settings we study: because we fine-tune latent-space backbones, which dominate current high-resolution generation, the effective forward operator composes the decoder with the pixel-space measurement process, and is non-linear even when the latter is linear; the latent activations likewise do not match the measurement resolution the layer requires. InvFusion is therefore not applicable to our tasks, and we compare the two designs conceptually. FlowBender instead computes the feedback signal externally, at the estimated clean sample $\hat{\mathbf{x}}_1$, and injects it as an additional conditioning input. Our method, therefore, extends correction to latent-space models and to complex, non-linear, and even non-differentiable operators. It also makes the approach architecture-agnostic rather than a specific network design: we demonstrate it with both ControlNet and LoRA adapters, across image and 3D modalities.

Simulator Feedback for Posterior Inference. Holzschuh & Thuerey (2024) present a flow-matching framework for simulation-based inference that shares a high-level principle with our approach: using feedback from a forward operator to inform correction during flow-based sampling. It operates, however, in a markedly different setting, regressing a small set of physical simulation parameters rather than performing high-dimensional visual or geometric generation. We detail the differences across three axes: the method itself (conceptual differences), the setting each was designed for (scale and task), and — as we verify empirically — the resulting behavior when the two designs are compared directly in our domain.

Conceptual differences. At a high level, the design of Holzschuh & Thuerey (2024) reflects an implicit assumption that the necessary correction is small in magnitude and only accurate enough to exploit late in the sampling trajectory. This assumption is reflected in three coupled design choices: the pretrained flow network is kept frozen while a separate, small network ($\approx 10\%$ of parameters) is trained, its output added as a residual correction on top of the frozen prediction; this correction is restricted to the cleanest 20% of the trajectory, where the one-step estimate is trusted; and the small network is trained only after the base model has converged, for a comparatively small number of additional iterations. FlowBender makes no such assumptions about the scale, form, or timing of the needed adjustment. We instead train a single, full-capacity network to consume feedback as a first-class input and to produce the corrected velocity outright, across the entire trajectory, jointly with the base flow-matching objective. Nothing constrains this correction to be small, additive, or confined to late timesteps — a distinction we verify empirically in Section 5.4, where we show that $\approx 80\%$ of the correction's energy is orthogonal to the feedback gradient, indicating a genuinely non-linear use of the signal rather than a well-tuned residual nudge.

Scale and task. Holzschuh & Thuerey (2024) target low-dimensional simulation-based inference (17–23 scalar parameters) with small MLPs, validated against baselines on scientific inference benchmarks. FlowBender targets high-dimensional conditional generation — image latents and 3D texture latents — and is evaluated on perceptual and geometric fidelity and plausibility (FID) across

a broad range of tasks and operator types: super-resolution, depth- and edge-to-RGB translation, JPEG restoration, and 3D mesh texturing.

Empirical comparison. To test whether this architectural distinction matters in practice rather than being cosmetic, we implemented two variants in the spirit of Holzschuh & Thuerey (2024) within our 3D texturing setting; an exact reproduction is not possible, as their code was not available at the time of writing. Both variants match the reported $\approx 10\%$-of-parameters budget relative to the full texturing transformer, and differ only in architecture: one is a generic small residual network as originally proposed, the other uses DiT-style transformer blocks at the same parameter count, in line with our base network (TRELLIS-2), to rule out that the choice of building block, rather than the training regime, explains any gap. Both are trained on top of our fully fine-tuned Standard FT model for an additional 10K iterations, gated to the cleanest 20% of the trajectory, using the same feedback signal as our first-order variant. Against the Standard FT baseline (M.PSNR 21.91, FID 8.74), both variants recover only a marginal gain (M.PSNR 22.2–22.3, FID 8.6–8.7; a ≈ 1.3–1.8% relative M.PSNR improvement and $\approx 1\%$ relative FID improvement), compared to our method's M.PSNR 26.39 and FID 6.64 ($\approx 20\%$ and $\approx 24\%$ relative improvement, respectively). Since the two variants perform comparably despite the architectural change, the gap is not attributable to the specific choice of building block for the residual module, but to the joint, full-trajectory training regime itself: simply attaching a small residual-correction head to an already fine-tuned network does not recover FlowBender's gains in this setting.

C Inference-Time Cost

We present a breakdown of the inference cost of FlowBender across its variants, compared against Standard FT and IT Guidance (FlowChef), on the super-resolution task. We report per-image wall-clock time on an NVIDIA A100-80GB GPU (bf16, 1024^2 resolution, batch size 1), decomposed into denoiser (DiT + ControlNet) cost, forward-operator $\mathcal{H}$ cost, and VAE cost (decode of $\hat{\mathbf{x}}_1$, its backward pass where applicable, and the residual/condition encode).

Table 6 compares our variants to Standard FT. The shortcut (Section 4.5) substantially reduces inference cost, saving a denoiser evaluation at every step but the first ($33.8 \rightarrow 20.5$ s for the zero-order variant). The VAE, however, carries a significant cost of its own that the shortcut largely retains, which is what gives Standard FT its advantage at matched NFEs. This VAE cost follows from our choice to fine-tune a latent-space backbone, the current state of the art, and is not inherent to FlowBender: the method applies identically to pixel-space diffusion and flow models, where the overhead would not arise. The VAE backward pass required by the first-order variant is itself noticeable, and accounts for the entire gap between the zero- and first-order variants (4.5 s in both the two-pass and shortcut settings). The forward operator is negligible here, as it is a downsampling operator; neural-network operators such as depth and edge extractors are heavier, but even in the heaviest case we tested (depth) the operator accounts for roughly 3–10% of total inference time.

Table 7 compares FlowChef, representative of the IT Guidance methods, to our shortcut variants. FlowChef also requires a VAE decode and backward pass to compute its guidance term. Our first-order shortcut is on par with it (25.0 vs. 24.1 s), while our zero-order shortcut, which does not backpropagate through the decoder, is meaningfully faster (20.5 s).

D Additional Results

D.1 Image-to-Image Translation

In this section we supplement the main paper results with additional configurations, baselines, and hyperparameters; implementation details are given in Appendix A.2. The full, extended quantitative comparison appears in Tables 9 to 12, where the configurations that appear in the main text tables are marked with $(*)$. Additional qualitative comparisons are provided for JPEG restoration in Fig. 9, depth-to-RGB in Fig. 10, and super-resolution in Fig. 11.

Inference-Time Guidance. Methods in this family modify a pre-trained model's sampling trajectory by adding a *guidance* term (see Eq. 1), closely related in form to our first-order feedback variants. A wide variety of such methods have been proposed for both diffusion and flow match-

Table 6: **Per-image inference cost decomposition (super-resolution).** Wall-clock time on a single A100-80GB. "Two-pass" is the full feedback regime ($2N$ NFE); "shortcut" reuses the previous step's $\hat{\mathbf{x}}_1$ for all but the first step ($N{+}1$ NFE, $t_{\text{thresh}} = 0$). The shortcut removes roughly half the denoiser cost, while the VAE cost is largely retained.

Method	NFE	Total (s)	Denoiser (s)	Fwd. op. (s)	VAE (s)	Other (s)
Standard FT	40	10.5	10.3	0.00	0.2	0.1
Standard FT + CFG	80	20.1	20.0	0.00	0.3	0.0
Zero-order, two-pass	80	33.8	20.7	0.00	12.9	0.3
Zero-order, shortcut	41	20.5	10.6	0.00	9.7	0.3
First-order (w.r.t. $\hat{\mathbf{x}}_1$), two-pass	80	38.3	20.7	0.02	17.4	0.4
First-order (w.r.t. $\hat{\mathbf{x}}_1$), shortcut	41	25.0	10.6	0.02	14.2	0.4

Table 7: **Per-image inference cost comparison with FlowChef (super-resolution).** The FlowChef baseline adds no denoiser evaluations compared to Standard FT, but pays a VAE decode-and-backward, leaving it on par with our first-order shortcut and slower than our zero-order shortcut.

Method	NFE	Total (s)	Denoiser (s)	Fwd. op. (s)	VAE (s)	Other (s)
Standard FT	40	10.5	10.3	0.00	0.2	0.1
FlowChef	40	24.1	10.2	0.03	13.9	0.2
Zero-order, shortcut	41	20.5	10.6	0.00	9.7	0.3
First-order (w.r.t. $\hat{\mathbf{x}}_1$), shortcut	41	25.0	10.6	0.02	14.2	0.4

ing models. We selected FlowChef (Patel et al., 2025) as our primary representative baseline for its generality (applicable to any differentiable operator, not just linear), latent-space compatibility, open-source implementation, and simplicity.

While FlowChef is operator-general in principle, and shows reasonable performance on simple forward operators such as super-resolution, we find that it does not generalize in practice to complex forward operators such as neural networks, as evidenced by the poor results in Tables 10 and 12. To rule out insufficient tuning, we performed a dense sweep over key hyperparameters, including learning rate (guidance strength) $\lambda \in \{1\times10^{-5}, 2.5\times10^{-5}, 5\times10^{-5}, 7.5\times10^{-5}, 1\times10^{-4}, 2.5\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}, 1\times10^{-2}, 2\times10^{-2}, 5\times10^{-2}, 1\times10^{-1}, 5\times10^{-1}, 1.0\}$, the percentage of steps applying guidance $(80\%, 100\%)$, and total number of steps $(40, 80, 100)$. No setting yielded both satisfactory visual quality and adherence to the conditioning for the Edge and Depth tasks; we provide qualitative examples of these failures in Fig. 8.

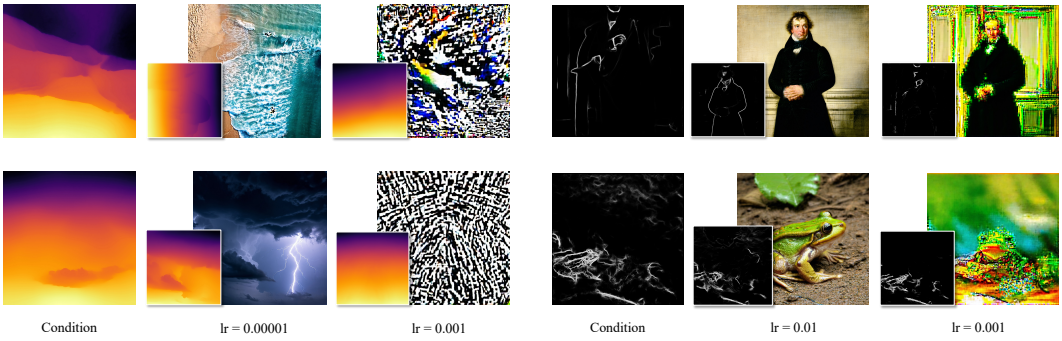
We further compare against three additional recent guidance-based frameworks, FlowDPS (Kim et al., 2025), LatentDAPS (Zhang et al., 2025), and FlowLPS (Park & Ye, 2025), on the super-resolution task, which is a linear inverse problem. Implementation details are available in Appendix A.2, and results in Table 9. The trade-off is not specific to FlowChef: all three reproduce the same pattern, each reaching high measurement-space fidelity only at a cost to plausibility. LatentDAPS attains the highest measurement fidelity among the guidance baselines, but at roughly 585 s per image, compared to $\approx$20–40 for our variants (Section C); FlowLPS and FlowDPS require $\approx$234 and $\approx$40, respectively, on the same hardware.

Overall, guidance methods can reach arbitrarily high fidelity by increasing guidance strength, but this comes at a heavy plausibility cost. These methods exhibit a strict trade-off between plausibility and fidelity across all tasks depending on hyperparameter choices. The tradeoff is not an artifact of any single baseline choice — it reproduces across four independently-designed guidance methods — but an inherent property of the guidance paradigm itself. FlowBender achieves strong fidelity and plausibility simultaneously rather than trading one for the other.

Reward-Based Fine-Tuning. These methods improve measurement fidelity by modifying the training objective, replacing or supplementing the diffusion/FM loss with an explicit measurement-matching reward term. We use ControlNet++ (Li et al., 2024), which adds such a term during training, as the representative baseline in this family, as it was originally proposed in a similar context:

Table 8: **Checkpoint selection for reward fine-tuning baselines (edge-to-RGB).** Each reward method is reported under two checkpoint-selection criteria. Selecting by best reward maximizes fidelity but collapses plausibility (reward hacking); selecting by best LPIPS preserves plausibility but does not improve fidelity over *Standard FT*. FlowBender results are repeated from Table 12 for reference.

	Fidelity		Plausibility
Method	Edge MAE ↓	Edge MSE ↓	FID ↓
Standard FT	$0.0533_{\pm 0.0286}$	$0.0137_{\pm 0.0087}$	13.98
Checkpoint selected by best reward			
AlignProp	$0.0286_{\pm 0.0170}$	$0.0042_{\pm 0.0030}$	68.48
DRaFT-1-LV	$0.0376_{\pm 0.0225}$	$0.0079_{\pm 0.0058}$	46.50
DRaFT-5	$0.0291_{\pm 0.0170}$	$0.0045_{\pm 0.0032}$	51.79
DDPO	$0.0516_{\pm 0.0283}$	$0.0128_{\pm 0.0087}$	18.19
Checkpoint selected by best LPIPS			
AlignProp	$0.0526_{\pm 0.0281}$	$0.0133_{\pm 0.0085}$	14.00
DRaFT-1-LV	$0.0535_{\pm 0.0285}$	$0.0137_{\pm 0.0087}$	13.97
DRaFT-5	$0.0525_{\pm 0.0281}$	$0.0133_{\pm 0.0085}$	14.01
DDPO	$0.0535_{\pm 0.0285}$	$0.0137_{\pm 0.0087}$	13.99
FlowBender (Combined w.r.t. $\hat{\mathbf{x}}_1$)	$0.0408_{\pm 0.0232}$	$0.0085_{\pm 0.0060}$	13.68



(a) Depth (b) Edge

Figure 8: **Qualitative examples of failures of FlowChef for neural-network forward operators.** As the adherence to the condition improves, shown as insets, the visual quality degrades significantly.

improving measurement fidelity in conditional generation. Other reward-based fine-tuning methods have been proposed as post-training procedures, typically targeting objectives such as human preference; we present comparisons to three popular approaches, DDPO (Black et al., 2024), DRaFT (Clark et al., 2024), and AlignProp (Prabhudesai et al., 2024), using a measurement-matching reward. The full results appear in Tables 9 to 12, and implementation details are available in Appendix A.2. These baselines exhibit a trade-off similar to the guidance approaches: the modified objective can improve fidelity, but at a heavy cost to plausibility, a failure mode known as *reward hacking* (see Fig. 12). As can be seen in Table 8, stopping training early preserves plausibility but yields no fidelity gain. In contrast, FlowBender remains faithful to the original FM loss and improves fidelity and plausibility simultaneously.

Increased Test-Time Compute. In most experiments in the main paper we fix the number of sampling steps N, so our full two-pass method incurs twice the network evaluations. In Section 5.3 we show that the inference-time shortcut preserves much of this performance at a cost of only $N+1$ evaluations, while still outperforming the baselines. Here we examine the opposite end: whether granting the baselines the same additional compute through standard means closes the gap. It does

Table 9: **Super resolution results.** Asterisks (*) denote base configurations compared in the main text. Shaded rows indicate FlowBender variants.

	Fidelity			Plausibility
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Standard FT (*)	$34.35_{\pm 2.78}$	$96.88_{\pm 4.32}$	$0.83_{\pm 0.43}$	3.93
+ CFG ($w = 3.0$)	$34.65_{\pm 2.94}$	$97.23_{\pm 3.29}$	$0.91_{\pm 0.49}$	4.09
+ CFG ($w = 5.0$)	$34.07_{\pm 3.02}$	$97.17_{\pm 2.76}$	$1.15_{\pm 0.68}$	4.57
+ 2× steps	$34.46_{\pm 2.74}$	$96.93_{\pm 3.84}$	$0.86_{\pm 0.44}$	4.07
+ ABM-Solver	$34.16_{\pm 2.71}$	$96.56_{\pm 3.99}$	$0.98_{\pm 0.51}$	4.71
+ RF-Solver	$34.27_{\pm 2.69}$	$96.71_{\pm 3.80}$	$0.95_{\pm 0.49}$	4.51
Standard FT + $\mathcal{L}_{\text{align}}$ (*)	$35.21_{\pm 2.80}$	$97.53_{\pm 2.96}$	$0.79_{\pm 0.36}$	4.11
+ 2× steps	$35.11_{\pm 2.81}$	$97.43_{\pm 2.74}$	$0.83_{\pm 0.39}$	4.10
+ ABM-Solver	$34.82_{\pm 2.78}$	$97.09_{\pm 2.81}$	$0.94_{\pm 0.46}$	4.50
+ RF-Solver	$34.89_{\pm 2.81}$	$97.16_{\pm 2.78}$	$0.91_{\pm 0.44}$	4.40
IT Guidance ($\lambda = 0.1$)	$38.30_{\pm 2.80}$	$97.20_{\pm 1.91}$	$1.33_{\pm 1.46}$	20.93
IT Guidance ($\lambda = 0.5$) (*)	$43.02_{\pm 4.85}$	$98.29_{\pm 3.19}$	$0.65_{\pm 2.24}$	18.96
IT Guidance ($\lambda = 0.5$, 2× steps)	$46.05_{\pm 4.91}$	$98.95_{\pm 2.72}$	$0.27_{\pm 1.18}$	18.87
FlowDPS	$37.58_{\pm 2.82}$	$96.64_{\pm 1.99}$	$2.03_{\pm 2.03}$	13.63
LatentDAPS	$54.75_{\pm 2.14}$	$99.67_{\pm 2.32}$	$0.02_{\pm 0.03}$	10.83
FlowLPS	$44.23_{\pm 5.07}$	$99.44_{\pm 0.66}$	$0.38_{\pm 0.72}$	8.14
AlignProp	$37.37_{\pm 3.19}$	$98.64_{\pm 1.63}$	$0.50_{\pm 0.25}$	4.24
DRaFT-1-LV	$36.24_{\pm 3.04}$	$98.11_{\pm 2.08}$	$0.67_{\pm 0.32}$	3.96
DRaFT-5	$37.24_{\pm 3.24}$	$98.69_{\pm 1.63}$	$0.50_{\pm 0.24}$	4.24
DDPO	$34.75_{\pm 2.75}$	$97.16_{\pm 3.57}$	$0.83_{\pm 0.42}$	3.99
Zero-order (*)	$39.25_{\pm 2.46}$	$98.18_{\pm 4.06}$	$0.21_{\pm 0.11}$	3.36
+ CFG ($w = 3.0$)	$45.25_{\pm 1.35}$	$99.08_{\pm 1.68}$	$0.13_{\pm 0.15}$	3.58
+ CFG ($w = 5.0$)	$42.61_{\pm 1.61}$	$98.70_{\pm 1.10}$	$0.34_{\pm 0.38}$	4.20
First-order (w.r.t. $\mathbf{x}_t$) (*)	$36.27_{\pm 3.95}$	$97.10_{\pm 4.69}$	$0.66_{\pm 0.39}$	4.30
First-order (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$39.21_{\pm 3.85}$	$97.64_{\pm 5.01}$	$0.43_{\pm 0.30}$	3.83
+ CFG ($w = 3.0$)	$45.14_{\pm 2.98}$	$98.95_{\pm 2.50}$	$0.17_{\pm 0.17}$	4.74
+ CFG ($w = 5.0$)	$44.07_{\pm 2.11}$	$98.96_{\pm 2.04}$	$0.33_{\pm 0.55}$	6.45
Combined (w.r.t. $\mathbf{x}_t$) (*)	$39.95_{\pm 2.77}$	$98.25_{\pm 4.18}$	$0.21_{\pm 0.12}$	3.40
+ CFG ($w = 3.0$)	$43.77_{\pm 1.40}$	$98.79_{\pm 2.58}$	$0.15_{\pm 0.17}$	3.62
+ CFG ($w = 5.0$)	$41.43_{\pm 1.57}$	$98.25_{\pm 2.10}$	$0.42_{\pm 0.49}$	4.39
Combined (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$39.77_{\pm 2.83}$	$98.24_{\pm 3.94}$	$0.23_{\pm 0.13}$	3.39

Table 10: **Depth-to-RGB generation results.** Asterisks (*) denote base configurations compared in the main text. Shaded rows indicate FlowBender variants.

	Fidelity		Plausibility
Method	MAE ↓	$\delta_{1.25}$ ↑	FID ↓
Standard FT (*)	$0.0847_{\pm 0.0593}$	$0.7886_{\pm 0.1930}$	18.21
+ CFG ($w = 3.0$)	$0.0814_{\pm 0.0394}$	$0.8003_{\pm 0.1331}$	14.90
+ CFG ($w = 5.0$)	$0.0823_{\pm 0.0388}$	$0.7959_{\pm 0.1333}$	15.77
+ 2× steps	$0.0844_{\pm 0.0415}$	$0.7907_{\pm 0.1381}$	16.74
+ ABM-Solver	$0.0838_{\pm 0.0585}$	$0.7894_{\pm 0.1917}$	14.97
+ RF-Solver	$0.0825_{\pm 0.0574}$	$0.7929_{\pm 0.1888}$	14.13
FT + $\mathcal{L}_{\text{align}}$ (*)	$0.0837_{\pm 0.0600}$	$0.7930_{\pm 0.1926}$	17.57
+ ABM-Solver	$0.0847_{\pm 0.0607}$	$0.7897_{\pm 0.1946}$	18.04
+ RF-Solver	$0.0833_{\pm 0.0580}$	$0.7930_{\pm 0.1910}$	16.85
IT Guidance ($\lambda = 0.0001$)	$0.1617_{\pm 0.0787}$	$0.5919_{\pm 0.1986}$	23.42
IT Guidance ($\lambda = 0.001$) (*)	$0.0866_{\pm 0.0360}$	$0.7592_{\pm 0.1284}$	223.54
IT Guidance ($\lambda = 0.01$)	$0.1385_{\pm 0.0501}$	$0.6145_{\pm 0.1378}$	328.57
DRaFT-1-LV	$0.0824_{\pm 0.0580}$	$0.7952_{\pm 0.1893}$	15.11
DRaFT-5	$0.0737_{\pm 0.0467}$	$0.8178_{\pm 0.1709}$	18.29
AlignProp	$0.0737_{\pm 0.0485}$	$0.8206_{\pm 0.1734}$	17.43
Zero-order (*)	$0.0763_{\pm 0.0563}$	$0.8188_{\pm 0.1843}$	15.70
First-order (w.r.t. $\mathbf{x}_t$) (*)	$0.0747_{\pm 0.0535}$	$0.8268_{\pm 0.1726}$	14.57
+ CFG ($w = 3.0$)	$0.0616_{\pm 0.0465}$	$0.8705_{\pm 0.1474}$	14.62
+ CFG ($w = 5.0$)	$0.0534_{\pm 0.0422}$	$0.8961_{\pm 0.1280}$	14.76
First-order (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$0.0818_{\pm 0.0582}$	$0.7973_{\pm 0.1933}$	15.89
Combined (w.r.t. $\mathbf{x}_t$) (*)	$0.0829_{\pm 0.0610}$	$0.7949_{\pm 0.1953}$	15.93
Combined (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$0.0783_{\pm 0.0606}$	$0.8175_{\pm 0.1878}$	15.39

Table 11: **JPEG restoration results.** Asterisks (*) denote base configurations compared in the main text. Shaded rows indicate FlowBender variants.

	Fidelity			Plausibility
Method	PSNR	SSIM	LPIPS	FID
Standard FT (*)	$26.29_{\pm 2.91}$	$79.45_{\pm 10.59}$	$22.24_{\pm 5.44}$	4.35
+ CFG. ($w = 3.0$)	$26.56_{\pm 3.04}$	$80.32_{\pm 10.18}$	$21.62_{\pm 5.48}$	4.36
+ CFG. ($w = 5.0$)	$26.58_{\pm 3.10}$	$80.85_{\pm 9.89}$	$21.44_{\pm 5.47}$	4.62
+ 2× steps	$26.30_{\pm 3.00}$	$79.17_{\pm 10.54}$	$22.59_{\pm 5.58}$	4.30
+ ABM-Solver	$26.09_{\pm 3.01}$	$78.19_{\pm 10.92}$	$23.50_{\pm 5.70}$	4.79
+ RF-Solver	$26.14_{\pm 3.03}$	$78.41_{\pm 10.77}$	$23.17_{\pm 5.64}$	4.60
DDPO	$25.32_{\pm 2.67}$	$75.01_{\pm 11.89}$	$25.36_{\pm 5.57}$	6.89
Zero-order (*)	$28.86_{\pm 4.00}$	$83.13_{\pm 9.64}$	$16.33_{\pm 4.59}$	3.80
+ CFG. ($w = 3.0$)	$29.79_{\pm 4.21}$	$85.58_{\pm 8.50}$	$14.72_{\pm 4.93}$	3.85
+ CFG. ($w = 5.0$)	$29.66_{\pm 4.07}$	$85.83_{\pm 8.02}$	$16.19_{\pm 5.95}$	4.58

Table 12: **Edge-to-RGB generation results.** Asterisks (*) denote base configurations compared in the main text. Shaded rows indicate FlowBender variants.

	Fidelity		Plausibility
Method	Edge MAE ↓	Edge MSE ↓	FID ↓
Standard FT (*)	$0.0533_{\pm 0.0286}$	$0.0137_{\pm 0.0087}$	13.98
+ CFG ($w = 3.0$)	$0.0500_{\pm 0.0286}$	$0.0137_{\pm 0.0092}$	12.41
+ CFG ($w = 5.0$)	$0.0513_{\pm 0.0296}$	$0.0150_{\pm 0.0101}$	13.64
$2\times$ steps	$0.0512_{\pm 0.0285}$	$0.0134_{\pm 0.0087}$	13.17
+ ABM-Solver	$0.0532_{\pm 0.0298}$	$0.0143_{\pm 0.0094}$	13.88
+ RF-Solver	$0.0516_{\pm 0.0291}$	$0.0137_{\pm 0.0089}$	12.84
Standard FT + $\mathcal{L}_{\text{align}}$ (*)	$0.0501_{\pm 0.0277}$	$0.0128_{\pm 0.0087}$	14.47
+ ABM-Solver	$0.0495_{\pm 0.0286}$	$0.0132_{\pm 0.0093}$	15.43
+ RF-Solver	$0.0477_{\pm 0.0280}$	$0.0126_{\pm 0.0089}$	13.99
IT Guidance ($\lambda = 0.001$)	$0.0620_{\pm 0.0359}$	$0.0247_{\pm 0.0167}$	23.97
IT Guidance ($\lambda = 0.01$) (*)	$0.0416_{\pm 0.0235}$	$0.0129_{\pm 0.0093}$	97.65
AlignProp	$0.0526_{\pm 0.0281}$	$0.0133_{\pm 0.0085}$	14.00
DRaFT-1-LV	$0.0535_{\pm 0.0285}$	$0.0137_{\pm 0.0087}$	13.97
DRaFT-5	$0.0525_{\pm 0.0281}$	$0.0133_{\pm 0.0085}$	14.01
DDPO	$0.0535_{\pm 0.0285}$	$0.0137_{\pm 0.0087}$	13.99
Zero-order (*)	$0.0456_{\pm 0.0253}$	$0.0104_{\pm 0.0072}$	14.81
First-order (w.r.t. $\mathbf{x}_t$) (*)	$0.0460_{\pm 0.0265}$	$0.0111_{\pm 0.0077}$	14.29
First-order (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$0.0435_{\pm 0.0294}$	$0.0123_{\pm 0.0089}$	14.97
Combined (w.r.t. $\mathbf{x}_t$) (*)	$0.0435_{\pm 0.0253}$	$0.0097_{\pm 0.0072}$	13.91
Combined (w.r.t. $\hat{\mathbf{x}}_1$) (*)	$0.0408_{\pm 0.0232}$	$0.0085_{\pm 0.0060}$	13.68
+ CFG ($w = 3.0$)	$0.0322_{\pm 0.0201}$	$0.0057_{\pm 0.0053}$	14.62
+ CFG ($w = 5.0$)	$0.0320_{\pm 0.0220}$	$0.0062_{\pm 0.0071}$	17.97

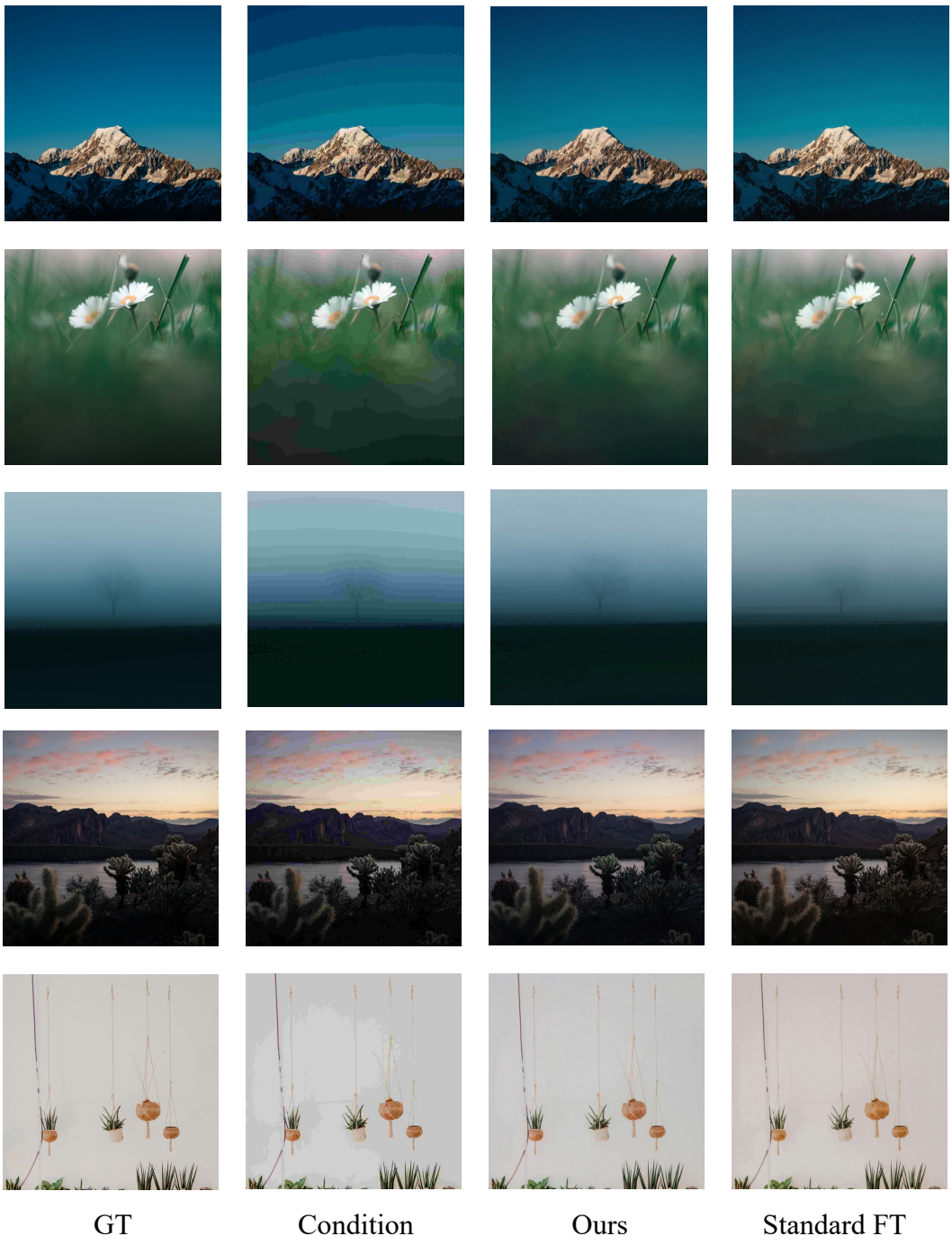


Figure 9: **Qualitative comparison for the JPEG restoration task.** Our method reduces color banding quantization artifacts (rows 1-3) and color shifts (rows 4-5).

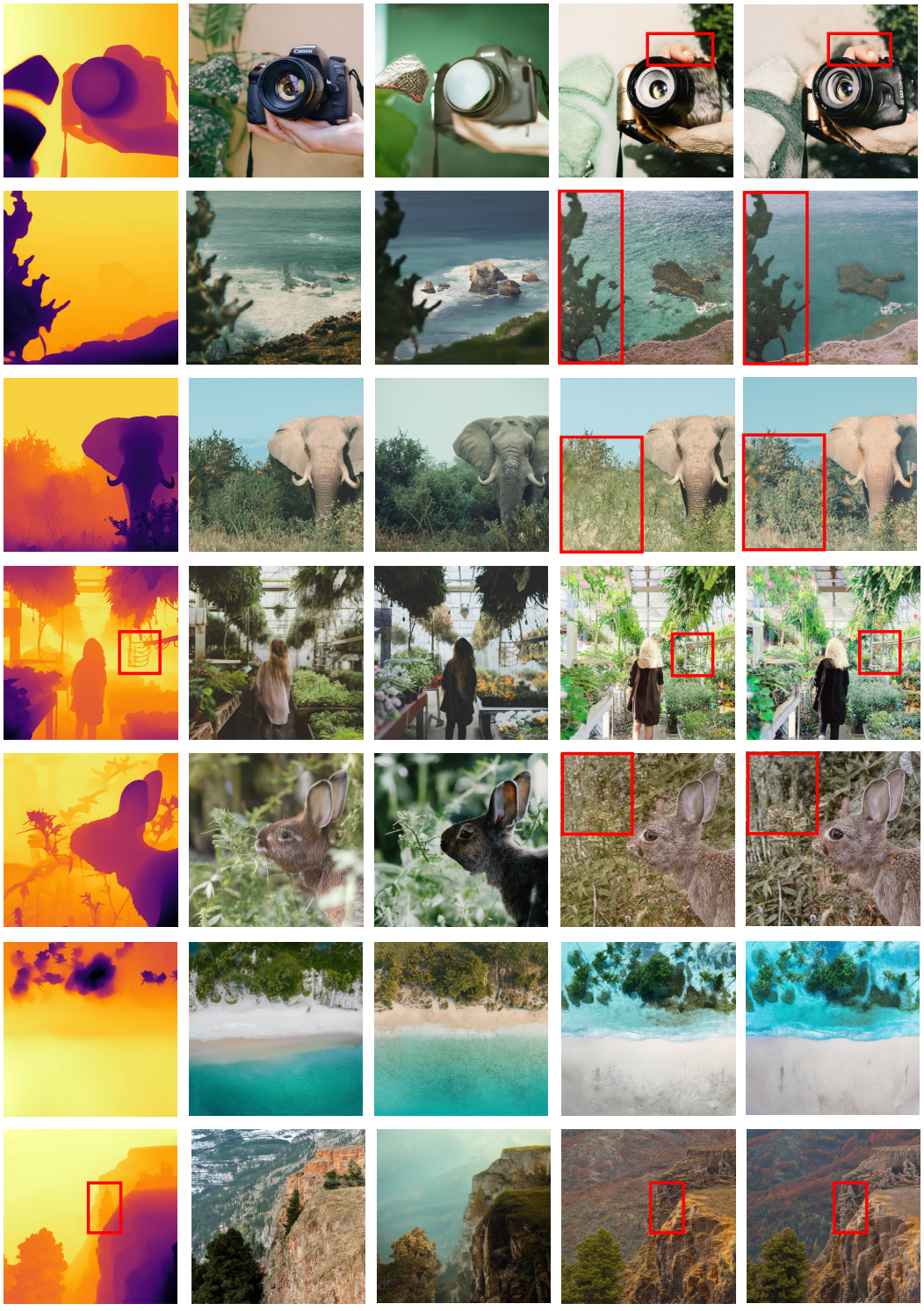


Figure 10: Qualitative comparison for the depth-to-RGB task. Red boxes highlight deviations from the conditioning image, which FlowBender avoids while maintaining visual quality.

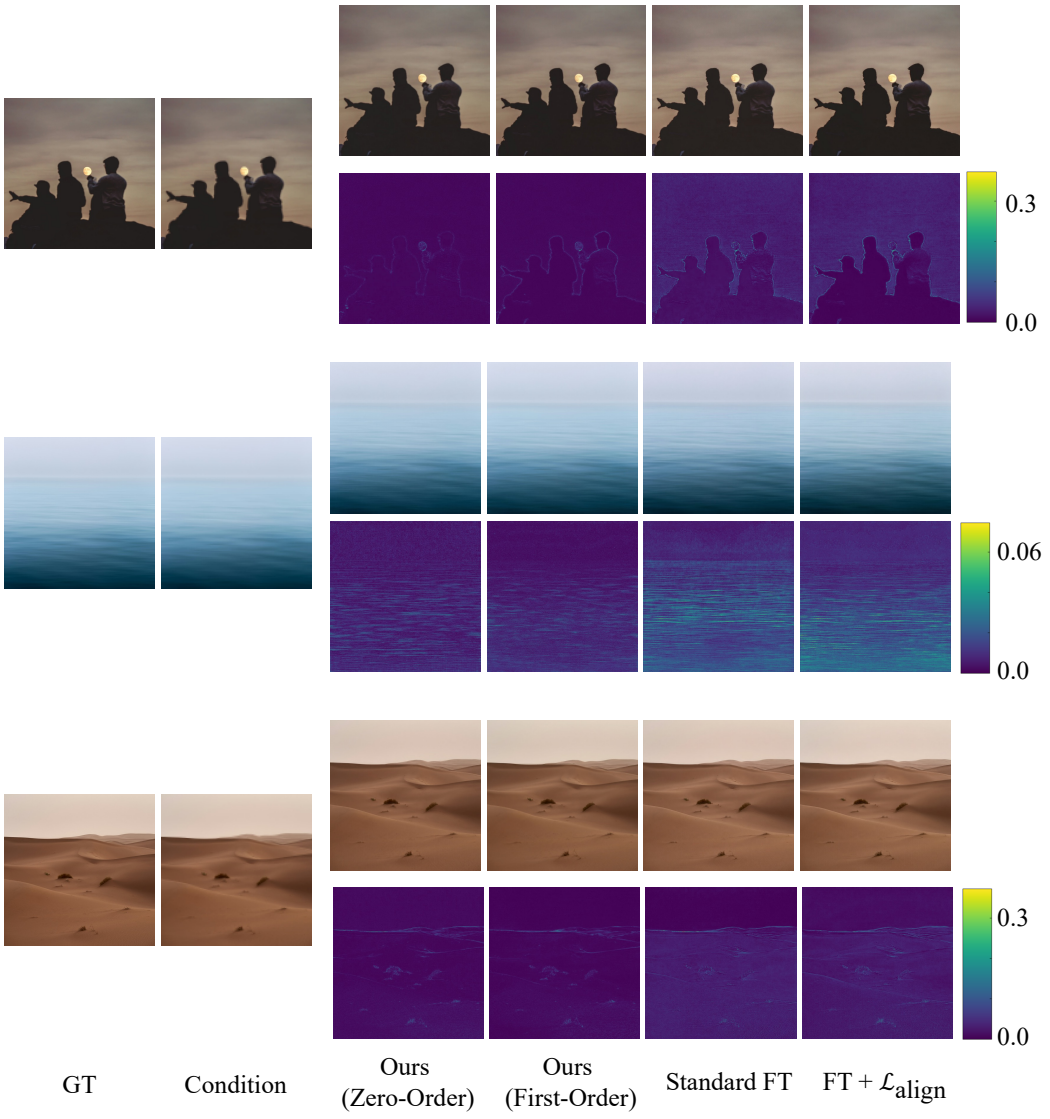


Figure 11: **Qualitative comparison for the super-resolution task.** Error maps show per-pixel MAE.

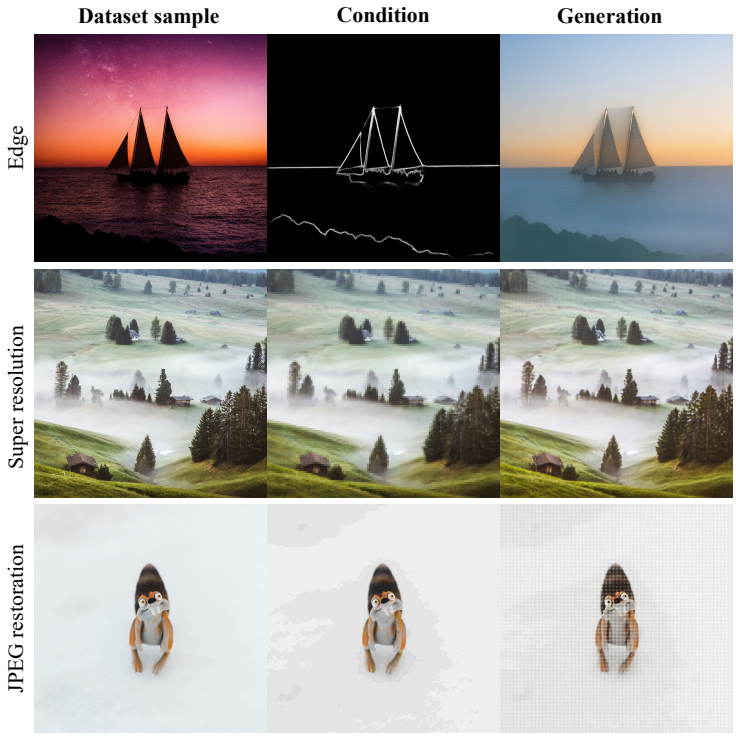


Figure 12: **Examples of reward hacking in DDPO (Black et al., 2024) under extended training.** The models sacrifice plausibility (perceptual quality) to maximize reward, in this case measurement fidelity.

not. Results appear in Tables 9 to 12. First, simply doubling the sampling budget ($2 \times N$ NFEs for N sampling steps) yields negligible improvements in fidelity metrics. Second, spending the same budget on a higher-order ODE solver (RF-Solver (Wang et al., 2024), ABM-Solver (Ma et al., 2025)) does not recover the gap either: it slightly hurts every fidelity metric on super-resolution and JPEG restoration, and on depth and edge it mostly improves FID rather than condition fidelity. Third, applying CFG (Ho & Salimans, 2022) (again $2 \times N$ NFEs) often degrades plausibility (FID) without meaningfully bridging the fidelity gap. These results indicate that open-loop failures cannot be resolved by standard test-time enhancements. In contrast, our CFG scheme (Section 4.5), which incurs no cost beyond the same $2 \times N$ budget, often boosts FlowBender's fidelity: as shown in Table 9, guidance with $w = 3.0$ improves our zero-order variant's PSNR by 6.0 dB, compared to a negligible 0.3 dB for the baseline.

Increased Training Budget. While the inference-time shortcut (Section 4.5) alleviates the increased cost at sampling time, our two-pass scheme still increases training-time compute relative to single-pass supervised baselines: roughly 1.27–$1.6\times$ for the first-order variant and 1.06–$1.34\times$ for the zero-order variant. To contextualize this overhead, we ran an additional control on the super-resolution task, doubling the supervised baselines' training budget in epochs so as to substantially exceed FlowBender's training compute. As shown in Table 13, extended training improves the baselines to some degree but does not close the gap: FlowBender still outperforms them on every fidelity and plausibility metric, at a lower compute budget than the extended baselines require. The advantage therefore does not follow simply from compute spent. Open-loop training plateaus well short of FlowBender even at nearly double our cost, which we take as evidence that the two-pass training overhead is a reasonable trade for the resulting gains.

D.2 Additional Ablations

We present here additional ablations on the super-resolution task.

Table 13: **Additional training budget for supervised baselines (super resolution).** Doubling the supervised baselines' training budget to exceed FlowBender's training compute improves them to some degree but does not close the gap to our method.

		Fidelity			Plausibility
Method	Compute (A100 h)	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Standard FT (5 epochs)	38	34.35	96.88	0.83	3.93
Standard FT (10 epochs)	76	36.55	97.57	0.56	3.69
Standard FT + $\mathcal{L}_{\text{align}}$ (5 epochs)	48	35.21	97.53	0.79	4.11
Standard FT + $\mathcal{L}_{\text{align}}$ (10 epochs)	96	35.64	97.30	0.58	3.67
Zero-order (*)	51	39.25	98.18	0.21	3.36

Table 14: **Restricted feedback window (super resolution, first-order w.r.t. $\hat{\mathbf{x}}_1$ variant).** Feedback is only provided in the cleanest w-fraction of the trajectory; $w = 1.0$ recovers the default, full-trajectory setting. Performance degrades monotonically as the window narrows, indicating that early-step feedback is useful.

	Fidelity			Plausibility
Window w	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
0.2 (cleanest 20%)	32.11	96.46	1.83	5.01
0.5	34.87	97.07	0.82	4.15
0.8	36.34	97.17	0.62	4.04
1.0 (default, full trajectory)	39.21	97.64	0.43	3.83

Restricting the feedback window. Since our feedback variants are based on the single-step prediction $\hat{\mathbf{x}}_1(\mathbf{x}_t)$, a natural question is how informative the feedback is in the noisy, low-t regime, and whether it should be restricted to the cleaner steps. As shown in Table 14, limiting feedback to a window of the cleanest denoising steps degrades performance monotonically as the window narrows, indicating that noisy early-step feedback is useful rather than merely tolerable. Conceptually, this can be understood as the model learning whether and how to use the feedback at each timestep in order to obtain the best velocity prediction.

Multi-step look-ahead estimate. We test whether obtaining $\hat{\mathbf{x}}_1$, from which the feedback is computed, via multiple Euler steps rather than a single-step prediction improves results. Beyond the obvious computational cost, Table 15 suggests that multi-step prediction generally *hurts* both fidelity and plausibility, for the first- and zero-order variants alike. We hypothesize that this is because the feedback computed at $\hat{\mathbf{x}}_1$ becomes a progressively worse proxy for the current state $\mathbf{x}_t$ of the trajectory.

Null Feedback Probability. The parameter p_{un} allocates the training budget between unguided and feedback-aware iterations. Conceptually, a non-zero p_{un} supports the model's ability to generate reliable look-ahead estimates, which serve as the foundation for the feedback signal. However, increasing this probability limits the iterations available for learning the second-pass refinement. Table 16 shows that $p_{un} = 0.1$ yields the best results across all metrics, providing the optimal balance for the closed-loop system.

D.3 3D Mesh Texturing

We present extended 3D texturing results in Tables 17 and 18, additional qualitative comparisons in Figure 13, and multi-view visualizations in Figure 14.

The patterns observed in image-to-image tasks (Appendix D.1) persist in the 3D domain (Tables 17 and 18). For open-loop baselines, standard enhancements like doubling sampling steps or applying CFG prove suboptimal or even detrimental; for example, applying CFG ($w = 3.0$) to Standard FT on Toys4K leads to a 1.57 dB drop in M.PSNR and a significant plausibility collapse (FID in-

Table 15: **Multi-step look-ahead (super resolution).** Obtaining the look-ahead estimate using 2 or 3 Euler steps instead of the default single step, for both the first-order and zero-order feedback variants. More steps generally degrade both fidelity and plausibility.

		Fidelity			Plausibility
Variant	Passes	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
First-order (w.r.t. $\hat{\mathbf{x}}_1$)	1 (default)	39.21	97.64	0.43	3.83
First-order (w.r.t. $\hat{\mathbf{x}}_1$)	2	38.42	97.87	0.61	4.42
First-order (w.r.t. $\hat{\mathbf{x}}_1$)	3	36.42	96.57	0.82	4.31
Zero-order	1 (default)	39.25	98.18	0.21	3.36
Zero-order	2	38.94	97.96	0.34	3.80
Zero-order	3	39.34	98.02	0.32	3.85

Table 16: **Ablation of first-pass training percentage p_{un} (super resolution).**

	Fidelity			Plausibility
p_{un}	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
0.0	37.38	97.41	0.49	3.89
0.1	39.21	97.64	0.43	3.83
0.2	38.55	97.57	0.48	3.93
0.3	37.55	97.24	0.53	3.92

creases from 8.87 to 10.30). While our proposed CFG scheme for FlowBender (Sec. 4.5) provides further fidelity and plausibility gains—such as a 1.04 dB M.PSNR and 0.65 FID improvement on Toys4K—we emphasize that FlowBender without any explicit guidance already significantly outperforms baselines using doubled sampling budgets or CFG. This indicates that while FlowBender can effectively leverage optional user-controlled test-time guidance, its primary strength lies in the learned correction policy which internalizes the feedback signal.

Table 17: **Extended Quantitative Results for 3D Texturing (Objaverse).** Asterisks (*) denote configurations included in the main text comparisons. Shaded rows indicate FlowBender variants.

	Fidelity				**Plausibility**				
Method	M. PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	MV-M.PSNR ↑	MV-SSIM ↑	MV-LPIPS ↓	MV-CLIP ↑	FID ↓
IT Guidance (*)	25.86±3.7	0.986±0.015	0.0191±0.0250	0.973±0.029	22.80±5.7	0.989±0.011	0.0123±0.0113	0.968±0.024	9.10
+ 2× steps	27.05±3.5	0.988±0.013	0.0174±0.0231	0.978±0.026	23.47±5.9	0.990±0.011	0.0122±0.0114	0.969±0.026	8.70
+ $\lambda = 0.5$	26.77±3.2	0.987±0.015	0.0218±0.0276	0.970±0.035	23.04±6.7	0.989±0.011	0.0136±0.0130	0.966±0.025	10.11
Standard FT (*)	21.91±3.8	0.981±0.020	0.0189±0.0178	0.970±0.022	23.05±6.3	0.990±0.012	0.0115±0.0114	0.970±0.022	8.74
+ CFG ($w = 3$)	20.41±4.4	0.980±0.023	0.0217±0.0215	0.961±0.030	21.06±5.5	0.988±0.015	0.0134±0.0136	0.963±0.024	10.11
+ CFG ($w = 5$)	19.45±4.8	0.975±0.034	0.0255±0.0273	0.952±0.042	19.83±5.0	0.986±0.020	0.0158±0.0167	0.955±0.029	12.53
+ 2× steps	21.68±3.9	0.980±0.021	0.0194±0.0184	0.970±0.022	22.77±6.5	0.989±0.012	0.0120±0.0117	0.969±0.022	8.81
FT + $\mathcal{L}_{\text{align}}$									
$\lambda_{\text{align}} = 10^{-3}$	22.49±3.8	0.983±0.017	0.0182±0.0162	0.971±0.021	22.89±6.4	0.990±0.011	0.0116±0.0110	0.970±0.021	8.79
$\lambda_{\text{align}} = 5 \times 10^{-3}$ (*)	22.63±3.6	0.983±0.018	0.0182±0.0160	0.970±0.023	22.48±5.3	0.990±0.011	0.0117±0.0110	0.969±0.021	8.94
$\lambda_{\text{align}} = 10^{-2}$	21.75±4.2	0.981±0.018	0.0193±0.0157	0.969±0.024	21.73±6.0	0.988±0.012	0.0127±0.0112	0.966±0.023	9.64
Ours (w.r.t. $\mathbf{x}_t$) (*)	25.26±3.9	0.985±0.017	0.0157±0.0152	0.979±0.017	25.91±7.2	0.991±0.011	0.0105±0.0109	0.974±0.021	7.41
Ours (w.r.t. $\hat{\mathbf{x}}_1$) (*)	26.39±3.8	0.987±0.016	0.0140±0.0142	0.983±0.014	26.53±7.2	0.992±0.010	0.0098±0.0101	0.977±0.019	6.64
+ CFG ($w = 3$)	27.02±5.1	0.989±0.013	0.0124±0.0128	0.986±0.017	25.85±7.2	0.992±0.010	0.0096±0.0100	0.978±0.018	6.25
+ CFG ($w = 5$)	26.35±5.6	0.989±0.012	0.0139±0.0153	0.983±0.023	24.43±7.2	0.991±0.011	0.0109±0.0111	0.973±0.022	7.35

Table 18: **Extended Quantitative Results for 3D Texturing (Toys4K).** Asterisks (*) denote base configurations compared in the main text. Shaded rows indicate FlowBender variants.

	Fidelity				Plausibility				
Method	M. PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	MV-M.PSNR ↑	MV-SSIM ↑	MV-LPIPS ↓	MV-CLIP ↑	FID ↓
IT Guidance (*)	26.82±3.2	0.990±0.009	0.0137±0.0136	0.982±0.017	23.63±4.2	0.992±0.008	0.0112±0.0111	0.972±0.020	8.43
+ 2× steps	27.95±3.2	0.991±0.008	0.0127±0.0125	0.984±0.016	24.42±4.7	0.992±0.009	0.0111±0.0110	0.973±0.020	7.90
+ $\lambda = 0.5$	28.02±3.4	0.991±0.009	0.0163±0.0195	0.975±0.030	23.94±5.2	0.991±0.008	0.0120±0.0116	0.970±0.020	9.00
Standard FT (*)	23.00±4.2	0.986±0.014	0.0160±0.0164	0.975±0.024	24.38±6.9	0.991±0.010	0.0114±0.0118	0.972±0.021	8.87
+ CFG ($w = 3$)	21.43±4.4	0.985±0.015	0.0182±0.0177	0.968±0.023	21.26±4.9	0.990±0.010	0.0130±0.0124	0.965±0.023	10.30
+ CFG ($w = 5$)	20.55±4.6	0.984±0.016	0.0210±0.0214	0.963±0.025	20.32±4.9	0.989±0.011	0.0151±0.0149	0.960±0.027	12.41
+ 2× steps	22.66±4.3	0.986±0.015	0.0163±0.0161	0.975±0.024	23.04±6.3	0.990±0.010	0.0118±0.0118	0.971±0.020	8.75
FT + $\mathcal{L}_{\text{align}}$									
$\lambda_{\text{align}} = 10^{-3}$	23.01±4.1	0.986±0.014	0.0162±0.0173	0.975±0.023	23.39±5.3	0.991±0.010	0.0114±0.0124	0.972±0.022	7.99
$\lambda_{\text{align}} = 5 \times 10^{-3}$ (*)	23.05±4.0	0.986±0.014	0.0159±0.0162	0.976±0.023	23.24±5.0	0.991±0.009	0.0111±0.0112	0.973±0.022	8.30
$\lambda_{\text{align}} = 10^{-2}$	22.79±4.2	0.986±0.015	0.0168±0.0169	0.975±0.024	23.16±5.7	0.990±0.010	0.0119±0.0116	0.971±0.022	8.60
Ours (w.r.t. $\mathbf{x}_t$) (*)	26.54±4.2	0.990±0.011	0.0129±0.0140	0.980±0.024	27.20±8.1	0.993±0.008	0.0097±0.0104	0.977±0.020	7.20
Ours (w.r.t. $\hat{\mathbf{x}}_1$) (*)	27.26±4.4	0.991±0.010	0.0116±0.0125	0.984±0.022	27.21±7.5	0.993±0.008	0.0092±0.0100	0.978±0.021	6.66
+ CFG ($w = 3$)	28.30±4.0	0.993±0.006	0.0099±0.0102	0.990±0.010	26.18±6.7	0.993±0.008	0.0093±0.0102	0.979±0.018	6.01
+ CFG ($w = 5$)	27.44±4.8	0.992±0.007	0.0114±0.0120	0.986±0.017	24.26±6.7	0.992±0.008	0.0109±0.0120	0.974±0.022	7.53

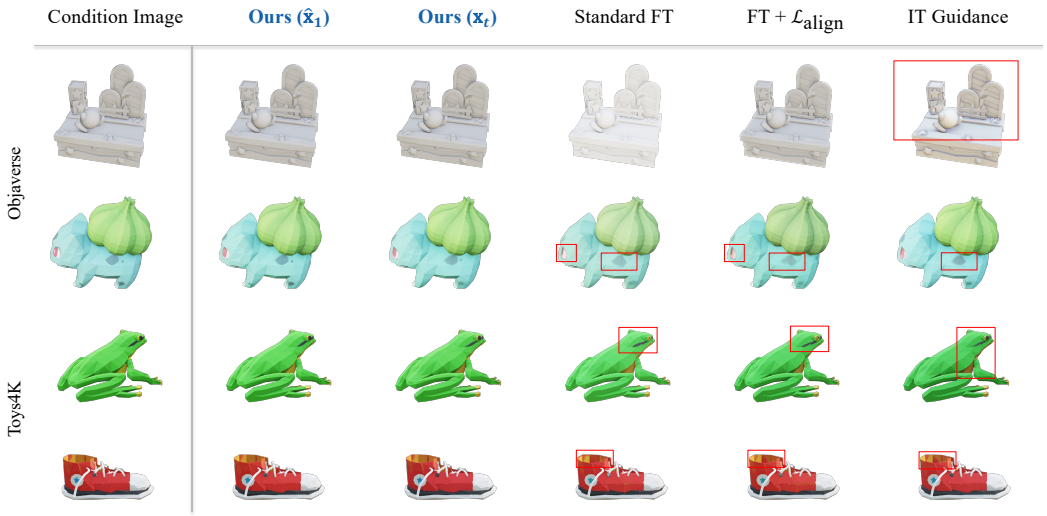


Figure 13: Additional 3D Mesh Texturing Results (Objaverse, Toys4K). Red boxes highlight deviations from the conditioning image, which FlowBender avoids while maintaining visual quality.

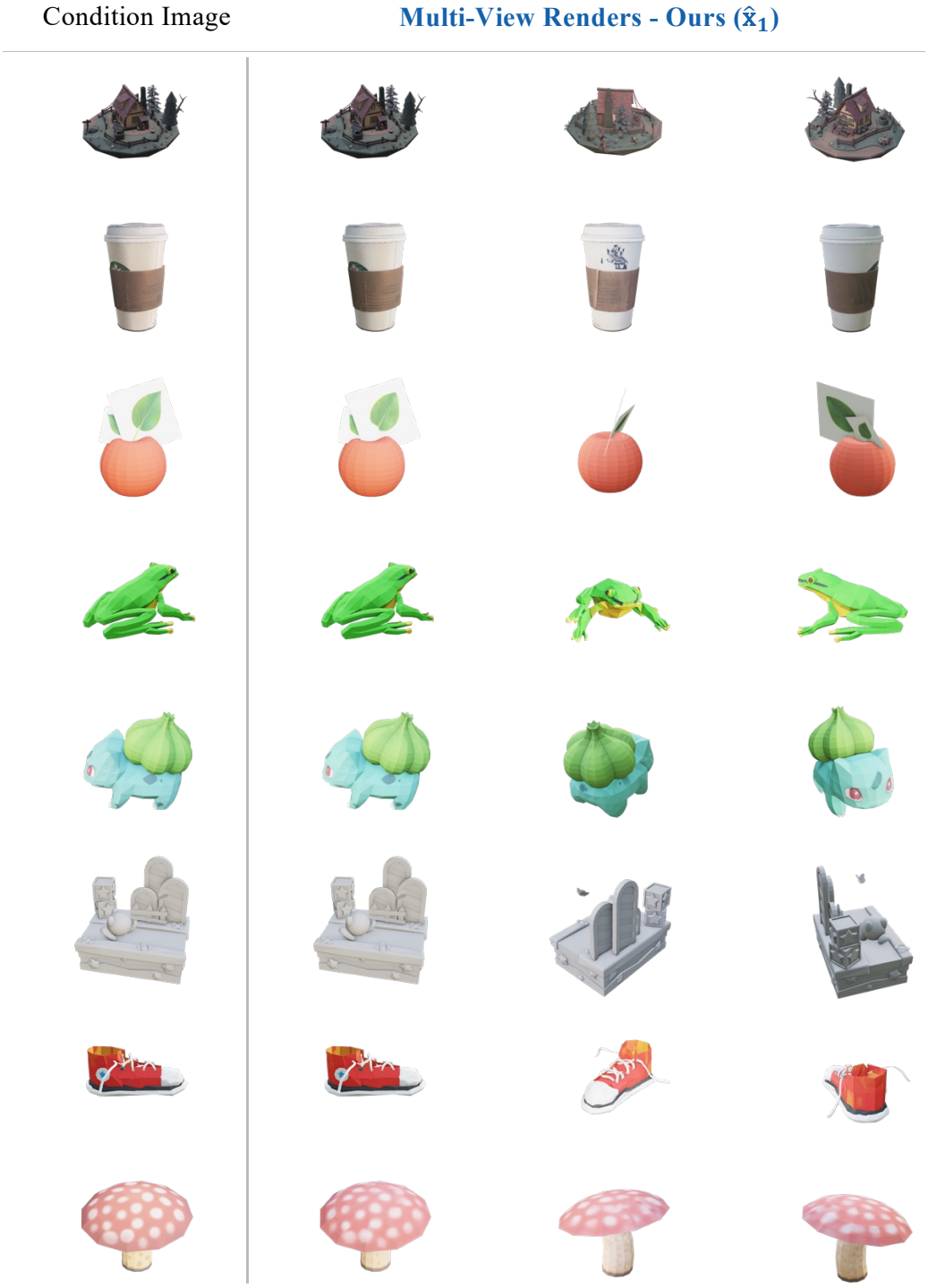


Figure 14: Multi-view visualizations of textured objects (Objaverse, Toys4K).

Table 19: **3D texturing under operator shift (elevated lighting, halved irradiance).** Models evaluated with the light source raised from 20° to 60° and irradiance halved compared to training. FlowBender remains the best on all three metrics.

	Fidelity		Plausibility
Method	M.PSNR ↑	LPIPS ↓	FID ↓
Standard FT	24.26	0.0163	9.50
Standard FT + $\mathcal{L}_{\text{align}}$	23.22	0.0171	10.00
IT Guidance	29.63	0.0154	9.58
Ours (First-order, w.r.t. $\hat{\mathbf{x}}_1$)	31.71	0.0122	7.18

Table 20: **Depth-to-RGB under operator shift (MiDaS depth estimator).** Models trained with Depth Anything V2, evaluated with MiDaS. Our variants maintain fidelity advantage over baselines.

	Fidelity		Plausibility
Method	MAE ↓	$\delta_{1.25}$ ↑	FID ↓
Standard FT	0.1130	0.6366	27.94
Standard FT + $\mathcal{L}_{\text{align}}$	0.1120	0.6340	28.44
IT Guidance	0.1349	0.5692	336.38
Ours (First-order, w.r.t. $\mathbf{x}_t$)	0.0958	0.6807	22.75
Ours (First-order, w.r.t. $\hat{\mathbf{x}}_1$)	0.1074	0.6528	28.19
Ours (Zero-order)	0.1077	0.6498	25.14

D.4 Robustness to Operator Shift

We tested trained FlowBender models under a mismatched forward operator at inference time: for super-resolution, bilinear downsampling (trained with average pooling); for depth, the MiDaS (Ranftl et al., 2022) depth extractor (trained with Depth Anything V2); for texturing, a shifted lighting configuration for the renderer (light source elevated from 20° to 60°, irradiance halved relative to training).

As shown in Tables 19 to 21, FlowBender generally retains its advantage under operator shift. In texturing it remains best on all three metrics, with a $\approx$7.5 dB M.PSNR gain over *Standard FT* and $\approx$2 dB over FlowChef (*IT Guidance*); in depth, all variants improve MAE and $\delta_{1.25}$ over baselines. In super resolution the picture is more complex: our zero-order variant gains modestly in measurement fidelity over the supervised baselines but at slightly higher FID, while FlowChef attains the best measurement fidelity at a large plausibility cost (roughly twice the FID of *Standard FT*). FlowChef, being training-free, is largely unaffected by the shift itself, and its behavior mirrors the main setting: decent performance on the closed-form downsampling operator, collapsing on the neural-network operator (FID 336.38 on depth), and in both cases still bound by the fidelity–plausibility trade-off. We take these results as evidence that FlowBender's learned correction generalizes to operator shift rather than overfitting to the exact training-time operator.

Table 21: **Super resolution under operator shift (bilinear downsampling).** Models trained with average-pooling downsampling, evaluated with bilinear downsampling.

	Fidelity			Plausibility
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Standard FT	32.39	93.65	7.53	14.04
Standard FT + $\mathcal{L}_{\text{align}}$	32.68	94.12	7.88	14.68
IT Guidance	39.38	97.02	1.40	27.16
Ours (Zero-order)	33.44	93.55	7.17	15.90
Ours (First-order, w.r.t. $\mathbf{x}_t$)	32.62	92.58	9.08	17.43
Ours (First-order, w.r.t. $\hat{\mathbf{x}}_1$)	33.43	92.86	8.62	17.19