Root-Selecting Fixed-Point Inversion for Rectified Flows via Trajectory Straightness

Semin Kim **Jihwan Yoon** **Seunghoon Hong**
KAIST
{seminkim, jihwan.yoon, seunghoon.hong}@kaist.ac.kr

Abstract

Finding the initial noise that generates a given data sample, known as inversion, is a key component for downstream applications such as training-free image editing. Existing fixed-point inversion methods improve inversion accuracy by formulating each inversion step as a fixed-point problem, but they lack a principled mechanism for selecting among multiple fixed-point solutions that can arise in practice. We observe that different selections induce different inversion trajectories, leading to substantial variation in reconstruction and editing quality. For rectified flows, we further find that this variation is closely associated with trajectory straightness, motivating straightness as a principled selection criterion. We propose SelFix, a fixed-point inversion method that selects fixed-point solutions inducing straighter inverse trajectories while retaining convergence to an exact inverse root under standard local assumptions. Experiments on FLUX.1-dev and PIE-Bench show that SelFix improves fixed-point inversion, achieving stronger real-image reconstruction and better source-preserving prompt-based editing than prior inversion baselines. The code is available at `https://github.com/seminkim/selfix`.

1 Introduction

Recent advances in text-to-image generative models have enabled diverse, high-quality image generation, with leading backbones increasingly shifting from diffusion models [25, 27, 28] to rectified flows [2, 9, 17, 20]. Accordingly, accurate inversion of rectified flows, which maps a real image back to an initial noise, has recently received increasing attention [7, 14, 34, 35]. Accurate inversion enables various applications that reuse the pretrained generative backbone, such as *training-free* prompt-based image editing [3, 4, 13, 22, 24, 35].

A central difficulty in inversion is discretization error. Naively integrating the learned dynamics in the reverse direction does not exactly invert the finite-step solver, especially when the learned trajectory is curved or the solver uses only a small number of steps. To reduce this error, previous methods often change the conditioning branch [21], optimize auxiliary variables [8, 22], or modify the sampler to couple inversion and reconstruction streams [33]. Among the various approaches, fixed-point inversion methods [10, 23, 30] have shown promise for more accurate inversion. Fixed-point inversion provides an alternative, principled approach; it keeps the chosen discrete solver fixed and formulates each inversion step as a local inverse equation induced by that solver step and solves it iteratively. When the iteration converges, the recovered point is an exact inverse of the chosen discrete forward solver, without requiring auxiliary-variable optimization or changes to the sampling procedure.

Existing fixed-point inversion methods focus on finding a single valid inverse point efficiently [23, 30]. Finding any single root is sufficient when the local inverse equation has a unique solution, or when all solutions lead to similar downstream behavior. In practice, however, we observe that the learned and discretized local inverse equation can admit multiple distinct approximate fixed-point roots. In this multi-root regime, achieving a small fixed-point residual alone is not enough. Different candidates

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

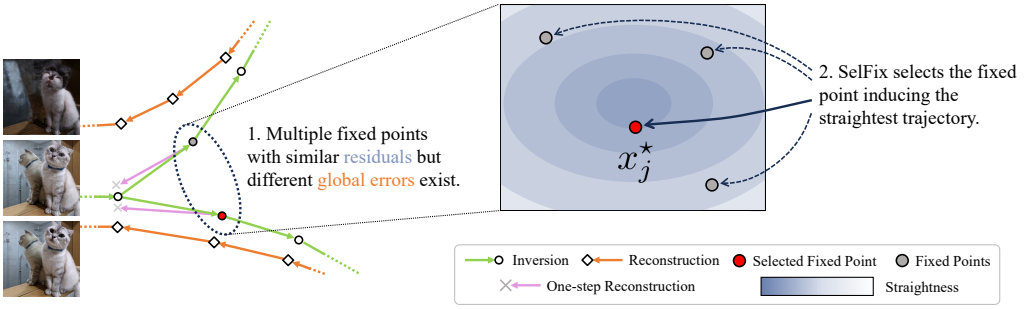


Figure 1: **Overview of SelFix.** Fixed-point inversion may produce multiple approximate fixed points with similar residuals but different global reconstruction errors. SelFix uses a causal straightness selector to guide fixed-point iteration toward straighter inverse trajectories, empirically improving reconstruction.

with comparable local residuals can lead to substantially different inversion trajectories, affecting downstream reconstruction or editing quality. We therefore view fixed-point inversion not only as solving a local inverse equation, but also as selecting a desirable solution among multiple valid candidates, as illustrated in Fig. 1.

For rectified flows in particular, a natural criterion for the selection problem is trajectory straightness. Rectified flows are designed so that straighter trajectories are easier to simulate with coarse discrete solvers [20], whereas curved trajectories make local solver approximations less reliable. Building on our empirical observation that reconstruction error is closely associated with inversion-path straightness, we propose **SelFix**, which **sel**ects a **fix**ed point that induces a straighter inversion trajectory, thereby improving rectified flow inversion. Viewing inversion as the selection of a particular fixed point, we derive a proxy for rectified flow's trajectory straightness [20] and use it as a criterion for selecting a root. Then, we design an iterative algorithm based on Halpern iteration [12], which blends the original fixed-point map with a straightness anchor derived from this criterion. We also prove that under standard assumptions, the proposed iteration converges to an exact fixed-point inverse that minimizes the straightness selector over the fixed-point set.

Our contributions are threefold. First, we formulate fixed-point inversion as root selection over the exact local inverse set, motivated by the observation that multiple approximate roots can have similar local residuals but substantially different global reconstruction errors. Second, we introduce a straightness-guided selector derived from rectified-flow trajectory straightness [20]. Third, we propose a theoretically grounded anchored fixed-point algorithm that selects a straighter inverse root while preserving exact fixed-point inversion asymptotically. Experiments on real-image reconstruction and prompt-based editing show that our method produces straighter inverse trajectories, yielding lower reconstruction error and better background preservation in image editing.

2 Preliminaries

2.1 Rectified Flow

Let $X_0 \sim p_{\text{data}}$ and $X_1 \sim p_{\text{noise}}$, where p_{noise} is typically a standard Gaussian distribution. Given a pair (X_0, X_1), rectified flow [20] learns a time-dependent velocity field $u_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ by matching the velocity of the linear interpolation

$$X_t = (1-t)X_0 + tX_1. \tag{1}$$

Since $\dot{X}_t = X_1 - X_0$, the standard conditional flow matching objective is

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_0,X_1}\left[\|u_\theta(X_t,t) - (X_1 - X_0)\|^2\right], \tag{2}$$

where $t \sim \text{Unif}[0,1]$. The learned velocity field induces the probability-flow ODE

$$\frac{dx_t}{dt} = u_\theta(x_t, t). \tag{3}$$

Data can be generated by numerically integrating Eq. (3) backward in time. Specifically, let $1 = t_T > t_{T-1} > \cdots > t_0 = 0$ be a discrete time grid, and let x_j denote the state at time t_j. Given a one-step numerical solver $f_{t_j \to t_{j-1}}$, generation amounts to the recurrence $x_{j-1} = f_{t_j \to t_{j-1}}(x_j)$, starting from random noise x_T and ending at the generated sample x_0. For the Euler solver in particular,

$$x_{j-1} = f_{t_j \to t_{j-1}}(x_j) = x_j - h_j u_\theta(x_j, t_j), \quad h_j = t_j - t_{j-1}. \tag{4}$$

A central property of rectified flow is that straight trajectories are easier to integrate with a small number of solver steps. For a continuous path $Z = \{Z_t\}_{t \in [0,1]}$, the trajectory straightness is commonly measured by:

$$S(Z) = \int_0^1 \mathbb{E}\left[||\dot{Z}_t - (Z_1 - Z_0)||^2\right] dt, \tag{5}$$

where $S(Z) = 0$ indicates perfectly straightened paths [20]. In practice, however, the learned velocity field may induce curved trajectories due both to averaging over many noise–data pairs and to model approximation error. Generation along such curved trajectories induces discretization error for solvers such as Euler, which rely on a constant-velocity approximation over each interval.

2.2 Inversion

Inversion is the problem of finding initial noise x_T that generates the given real data x_0. This is typically done by iteratively recovering the generation trajectory, starting from the data x_0. At the j-th iteration, the goal is to recover a state x_j, given the previously obtained states $x_{<j}$. With the definition of residual at time t_j as

$$r_j(x) := f_{t_j \to t_{j-1}}(x) - x_{j-1}, \tag{6}$$

the goal for each inversion step is to find x that satisfies zero residual:

$$r_j(x) = 0, \qquad \text{equivalently} \qquad f_{t_j \to t_{j-1}}(x) = x_{j-1}. \tag{7}$$

Solving Eq. (7) is generally challenging, since it requires inverting the solver f which itself depends on the learned velocity u_θ. The simplest strategy is to integrate Eq. (3) in the inverse direction with the same solver. For instance, for Euler solver, this gives $\hat{x}_j = x_{j-1} + h_j u_\theta(x_{j-1}, t_{j-1})$. This estimate is exact only in special cases, for example when the velocity remains constant across the steps, *i.e.*, $u_\theta(x_{j-1}, t_{j-1}) = u_\theta(x_j, t_j)$. In general, however, this assumption does not hold due to the non-straightness in the learned velocity (Sec. 2.1), incurring errors in the inversion process.

2.3 Fixed-Point Inversion

Fixed-point inversion improves upon direct reverse integration by explicitly solving the local inverse equation in Eq. (7). Define the fixed-point map

$$P_j(x) := x - r_j(x). \tag{8}$$

Then being the root of the fixed-point equation $x = P_j(x)$ is equivalent to being the root of $r_j(x) = 0$. Hence, the zero set of r_j is the same as the fixed-point set of P_j:

$$S_j = \{x : r_j(x) = 0\} = \mathrm{Fix}(P_j), \qquad \mathrm{Fix}(P_j) := \{x : P_j(x) = x\}. \tag{9}$$

Then the inversion can be done by running a fixed-point solving algorithm on P_j. One basic algorithm is applying Picard iteration:

$$x_j^{k+1} = P_j(x_j^k). \tag{10}$$

If Eq. (10) converges to some $x_j^\star$, then by construction $r_j(x_j^\star) = 0$, so the per-step inversion is solved exactly. Thus, if converged, fixed-point inversion gives exact inversion asymptotically, even with a coarse timestep discretization which induces large discretization error.

While empirically effective in practice, prior fixed-point inversion methods solve Eq. (9) for obtaining any single element of S_j. When $\mathrm{Fix}(P_j)$ contains multiple roots, Eq. (10) does not specify which exact inverse root should be preferred. In such a case, the selected limit simply depends on initialization or the local geometry of P_j. In the following section, we demonstrate that the learned model may admit multiple fixed points for Eq. (8), and selecting a fixed point that induces a straighter trajectory can improve the inversion.

3 Method

3.1 Fixed-Point Inversion as Root Selection

Existing fixed-point inversion methods seek an arbitrary element of S_j, which is sufficient when S_j is a singleton. When multiple roots exist, however, different choices of $x_j \in S_j$ can induce different inverse trajectories, even though they satisfy the same local reconstruction equation. Thus, we view fixed-point inversion not only as root finding, but also as root selection.

For the local convergence analysis, we restrict attention to a region of interest $C_j \subset \mathbb{R}^d$ where the fixed-point iteration is expected to operate, and define the local exact inverse root set

$$S_j^C := S_j \cap C_j = \mathrm{Fix}(P_j) \cap C_j. \tag{11}$$

We then formulate root-selecting inversion as

$$x_j^\star \in \arg\min_{x \in S_j^C} \phi_j(x), \tag{12}$$

where ϕ_j is a selection criterion. This formulation separates exactness from selection: the constraint $x \in S_j^C$ enforces exact local inversion, while ϕ_j specifies which exact root is preferred.

Empirical Existence of Multiple Fixed Points To show that the learned discrete inversion equation can admit multiple approximate fixed-point roots in practice, we analyze fixed-point inversion under different initializations. Specifically, we repeat fixed-point inversion on the same image with different random initializations, and track two quantities during the inner fixed-point iteration. The first is the mean fixed-point residual, computed as the average of $\|P_j(x) - x\|_2$ over differently initialized iterates. The second is the mean pairwise L_2 distance, computed between the iterates obtained from different initializations at the same timestep and inner iteration. As shown in Fig. 2, even near convergence where the mean fixed-point residual is small, differently initialized iterates tend to remain far from each other; their mean pairwise distance remains an order of magnitude larger than the mean fixed-point residual. Because the residual is small but the pairwise distance remains large, the converged iterates behave as distinct near-roots rather than variants of a single solution. This observation suggests that the learned discrete inversion equation can have multiple distinct approximate fixed-point roots in practice. We emphasize, however, that this multiplicity claim concerns the learned and discretized local inversion equation in Eq. (9); it does not claim non-injectivity of the ideal continuous-time ODE flow.

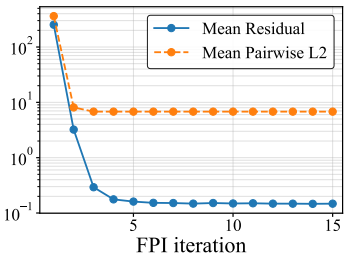
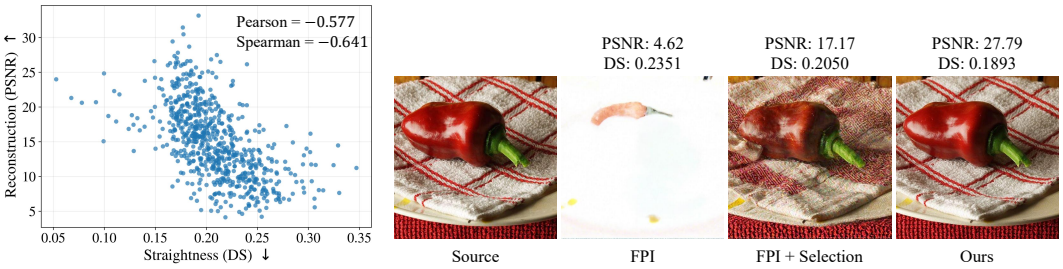


Figure 2: Comparison of average residual and average pairwise distance between different initializations. Analysis details, along with results across other backbones and datasets, are available in App. C.1.

3.2 Straightness Selector

Our root-selection view in Sec. 3.1 allows many possible choices of the root selector ϕ_j. To design a practical selector that improves the inversion, we first conjecture and validate that trajectory straightness is highly correlated with the failure of fixed-point inversion, motivating a straightness-based root selector.

Straightness and Inversion Failure If fixed-point inversion were solved to exact convergence at every step, a recovered state $x_j^\star$ would satisfy $r_j(x_j^\star) = 0$, so applying the same discrete generation solver would map it back to x_{j-1}. In practice, however, we only run a finite number of fixed-point iterations, and the obtained state $\hat{x}_j$ generally retains a nonzero residual $r_j(\hat{x}_j) \neq 0$. During reconstruction or editing, this residual becomes a local state mismatch that is recursively propagated by later generation steps. Notably, how severely such per-step errors accumulate depends on the local behavior of u_θ along the inverse trajectory, since the generation step re-evaluates u_θ at each replayed state (*e.g.*, Eq. (4)). We conjecture that curved trajectories are more susceptible to this accumulation because the local constant-velocity approximation of a discretized solver is less reliable, so small residuals can induce larger changes in subsequent velocity evaluations. Thus, among approximate



(a) Reconstruction PSNR versus discrete trajectory straightness.

(b) An image reconstruction result using straightness as a selection criterion.

Figure 3: Preliminary analysis regarding straightness. Straightness strongly correlates with reconstruction accuracy (left), and can be used as a selection criterion to enhance inversion (right).

fixed-point roots with comparable residuals, we prefer the one inducing a straighter inverse trajectory, as it is less likely to amplify residual errors during generation.

To empirically validate this claim, we define discretized straightness (DS), a sample-wise discrete counterpart of Eq. (5) that measures the straightness of an individual inversion trajectory:

$$DS(X) = \sum_{i=1}^{T} h_i ||v_i - \bar{v}||^2, \tag{13}$$

where $h_i = t_i - t_{i-1}$, $v_i = (x_i - x_{i-1})/h_i$, and $\bar{v} = \sum h_i v_i / \sum h_i$. Similar to the continuous counterpart, a perfectly straight path would yield $DS(X) = 0$.

In Fig. 3a, we plot reconstruction accuracy (PSNR) against the straightness measure (DS). The plot shows a strong correlation between them: straighter inverse trajectories tend to yield more accurate reconstruction. We also present an example in Fig. 3b showing that straightness can serve as a valid objective for root selection. The FPI+Selection variant, which performs best-of-N selection (details in App. C.2) to choose the straightest path among differently initialized states, can prevent the collapse of naive fixed-point iteration. These results suggest that straightness is a strong indicator of inversion failure and can also serve as a practical root-selection criterion. In later sections, we propose selecting such roots on the fly as the fixed-point iteration proceeds, which requires no multiple initializations and further improves reconstruction accuracy, as shown in Fig. 3b.

Straightness Root Selector Motivated by the previous observation, we construct a causal root selector inspired by the discretized rectified flow straightness in Eq. (13). The global mean velocity $\bar{v}$ depends on the entire trajectory and is unavailable during inversion. We therefore use the weighted average of previously observed velocities within a window of size W as a causal reference:

$$\bar{v}_{j-1}^{W} := \frac{\sum_{i=1}^{W} h_{j-i} v_{j-i}}{\sum_{i=1}^{W} h_{j-i}}. \tag{14}$$

Using this reference, we define an on-the-fly, per-step straightness proxy for root selection as follows.

Definition 1 (Straightness Root Selector)**.** *For a candidate current state x at time t_j, the straightness root selector is defined as*

$$\phi_{j,W}^{\mathrm{str}}(x) := h_j \left\| v_j(x) - \bar{v}_{j-1}^{W} \right\|^2 = \frac{1}{h_j} \|x - a_{j,W}\|^2, \tag{15}$$

which is a quadratic distance from the straightness anchor:

$$a_{j,W} := x_{j-1} + h_j \bar{v}_{j-1}^{W}. \tag{16}$$

Corollary. *Since the selector is a quadratic distance, the minimization problem in Eq.* (12) *reduces to*

$$x_j^{\star} \in \mathrm{proj}_{S_j^C}(a_{j,W}) := \arg\min_{x \in S_j^C} \|x - a_{j,W}\|^2 . \tag{17}$$

Although the form has changed, minimizing our root selector does not relax the exact inversion constraint. Moreover, the criterion requires no additional model forward passes, as it is computed directly from the previous inversion trajectory.

3.3 Root-Selecting Fixed-Point Inversion

The projection problem in Eq. (17) cannot be solved directly because the constraint set S_j^C is only available implicitly through the fixed-point equation $x = P_j(x)$. We therefore propose to apply the Halpern iteration [12]:

$$x_j^k = \alpha_k a_{j,W} + (1 - \alpha_k) P_j(x_j^{k-1}), \tag{18}$$

where $a_{j,W}$ is the straightness anchor defined in Eq. (16), and $\alpha_k \in (0, 1)$ is a sequence that satisfies:

$$\alpha_k \in (0,1), \qquad \alpha_k \to 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=2}^{\infty} |\alpha_k - \alpha_{k-1}| < \infty. \tag{19}$$

In our experiments, we consider the following family of schedules:

$$\alpha_k = \alpha_1 \frac{\delta}{(k-1) + \delta}, \qquad k \geq 1, \;\; 0 < \alpha_1 < 1, \;\; \delta > 0. \tag{20}$$

The schedule is controlled by two hyperparameters: α_1 sets the value at the initial iteration, and δ controls how fast the schedule decreases. The schedule satisfies the condition in Eq. (19) (App. D.3).

Under standard assumptions, the Halpern iteration converges strongly to the projection of the anchor $a_{j,W}$ onto the fixed-point set [36]. It works by blending two terms: the fixed-point term P_j enforces the original local inversion equation, while the anchor term $a_{j,W}$ biases the iteration toward a straighter trajectory. Since α_k vanishes in the limit, the anchor influences which root is selected, but the limiting point, when it exists, remains an exact inverse root.

We now state the guarantee under a standard local nonexpansiveness assumption commonly used in the fixed-point inversion literature [10, 30]:

Assumption 1 (Local Root-Selection setting)**.** *For each inversion step j, there exists a closed convex set $C_j \subset \mathbb{R}^d$ such that*

$$P_j(C_j) \subset C_j, \qquad a_{j,W}, x_j^0 \in C_j, \qquad S_j^C := \mathrm{Fix}(P_j) \cap C_j \neq \emptyset. \tag{21}$$

Moreover, P_j is nonexpansive on C_j:

$$\|P_j(x) - P_j(y)\| \leq \|x - y\|, \qquad \forall x, y \in C_j. \tag{22}$$

Theorem 1 (Straightness-minimizing fixed-point selection)**.** *Under Assumption 1 and with α_k satisfying Eq.* (19)*, the sequence generated by Eq.* (18) *converges to*

$$x_j^\star \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x). \tag{23}$$

Proof. The proof is given in App. D.4. □

Decoupled Momentum Although Eq. (18) gives the desired selection principle, we empirically find that smoothing the fixed-point correction improves finite-iteration behavior. Thus, we propose a decoupled momentum, where a momentum variable is applied only to the P_j term (which motivates the term *decoupled*), and the straightness anchor remains unchanged.

The decoupled momentum, with initialization set to $M_j^0 = x_j^0$, is:

$$M_j^k = \mu M_j^{k-1} + (1 - \mu) P_j(x_j^{k-1}), \tag{24}$$

$$x_j^k = \alpha_k a_{j,W} + (1 - \alpha_k) M_j^k, \tag{25}$$

where $\mu \in [0, 1)$ is the momentum coefficient that controls the strength of momentum; when $\mu = 0$, it exactly recovers the Halpern iteration in Eq. (18).

We posit that the decoupled momentum update yields better convergence by selectively smoothing only the update signal from $P_j(x_j^{k-1})$, which depends on iteration-varying model outputs whose

errors and noise can harm convergence. In contrast, the anchor $a_{j,W}$ remains the same throughout the iteration, so not averaging it into the momentum variable helps preserve the root-selection signal. In the finite-iteration regime, decoupled momentum can also help damp oscillations in the inner-loop updates by retaining a component of the previous iterate that counteracts sign-reversing fixed-point proposals. We provide a local analysis and empirical oscillation statistics in App. D.7.

Beyond this empirical gain, our decoupled momentum update also achieves exact convergence asymptotically. The complete inversion procedure, combining the straightness anchor, the anchored update, and decoupled momentum, is summarized in Algorithm 1 (App. B).

Theorem 2 (Momentum preserves the straightness-selected root)**.** *Under the same assumptions as Theorem 1, with α_k in Eq.* (20) *and with the momentum weight $\mu \in [0, 1)$, the sequence generated by Eqs.* (24)–(25) *satisfies*

$$x_j^k \to x_j^\star, \qquad x_j^\star \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x). \tag{26}$$

Proof. The proof is given in App. D.5. □

4 Related Work

Inversion Inversion is central to training-free image editing, but naive methods such as DDIM inversion [31] accumulate discretization error. Prior diffusion inversion methods reduce this error by modifying conditioning [21], optimizing auxiliary variables [8, 22], or coupling inversion and reconstruction streams [33]. Fixed-point inversion instead solves the discrete local inverse equation directly; ReNoise [10], AIDI [23], and GNRI [30] improve this process through iterative noising, acceleration, or Newton-style updates. These methods focus on obtaining a consistent inverse point, while SelFix addresses the complementary problem of selecting among distinct approximate roots.

Rectified flow inversion and trajectory straightness For rectified flow models, recent inversion methods such as RF-Inversion [29], RF-Solver [35], FireFlow [7], UniEdit-Flow [14], and PMI [34] reduce inversion error through tailored solvers, optimal-control formulations, or velocity priors. SelFix differs by preserving the fixed-point inverse equation and using trajectory straightness to select the preferred root. This is motivated by the design principle of rectified flow, where straighter trajectories are easier to simulate with coarse solvers [20]; related work improves straightness through coupling design, curvature or variance modeling, encoders, or test-time refinement [5, 11, 16, 18, 26, 32]. We instead use straightness as a causal root-selection criterion during inversion.

Fixed-point selection Halpern iteration is a classical fixed-point selection mechanism for non-expansive maps [1, 12, 36]. Our contribution is not a new general fixed-point algorithm, but an inversion-specific instantiation: we derive a rectified flow straightness anchor and show that the resulting anchored fixed-point update selects the corresponding exact local inverse root asymptotically.

5 Experiments

Following the convention [7], we validate the effectiveness of our method in real image reconstruction and editing tasks. For baselines, we consider methods using flow-tailored higher-order inversion solvers (RF-Solver [35] and FireFlow [7]), and methods using fixed-point formulation (ReNoise [10] and AIDI [23]). We also include naive baselines, one that simply integrates the flow in reverse direction (ReFlow) and one that does a naive fixed-point iteration (FPI). We use the FLUX.1-dev [2] model as a backbone, and the number of function evaluations (NFE) is matched across baselines for fair comparison. Additional experiment details are provided in App. E, along with further results using different backbones, editing strategies, or inversion baselines in App. F.

5.1 Image Reconstruction

Evaluation Metrics We report the reconstruction results on PIE-Bench [15] using standard reconstruction accuracy metrics. We also report inversion trajectory straightness in two ways: the discretized trajectory straightness DS in Eq. (13), and the accumulated selector value $\sum_j \phi_{j,W}^{\mathrm{str}}$, where $\phi_{j,W}^{\mathrm{str}}$ is the straightness proxy used by our root selector in Eq. (15).

Table 1: Quantitative image reconstruction results on PIE-Bench.

Method	LPIPS ↓	SSIM ↑	PSNR ↑	DS ↓	$\sum \phi^{\text{str}}_{j,W}$ ↓	NFE
ReFlow	0.2382	0.6965	19.54	0.181	0.176	166
Improved Solver						
RF-Solver	0.1997	0.7400	21.31	0.172	0.162	168
FireFlow	0.1037	0.8408	26.98	0.185	0.180	168
Fixed-Point Inversion						
ReNoise	0.1155	0.8159	22.97	0.177	0.157	165
AIDI	0.1533	0.7862	23.00	0.187	0.165	165
FPI	0.3059	0.6317	15.05	0.208	0.186	165
SelFix (Ours)	**0.0742**	**0.8714**	**28.29**	**0.170**	**0.151**	165

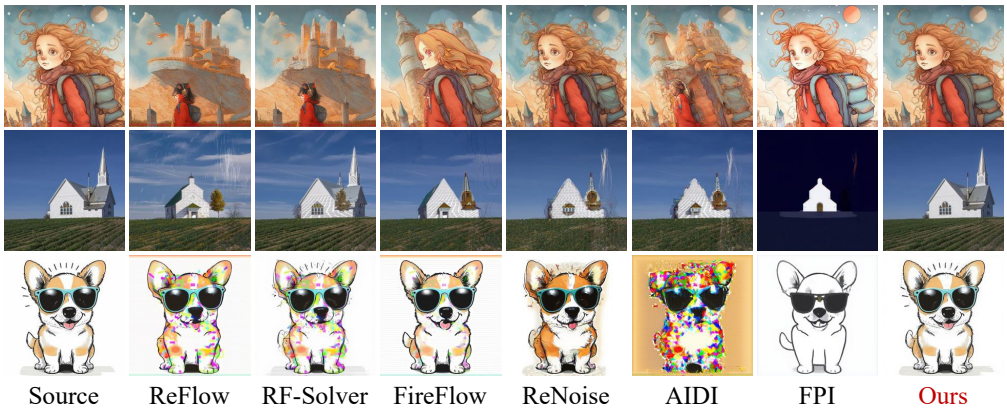


Figure 4: Qualitative real-image reconstruction results on PIE-Bench. More results are in App. F.4.

Results As shown in Tab. 1, SelFix achieves the best performance across all reconstruction metrics, outperforming both advanced fixed-point inversion methods and naive FPI. Qualitative results in Fig. 4 show the same trend: SelFix avoids the error accumulation that causes other fixed-point baselines to fail. This suggests that fixed-point inversion benefits from selecting among reachable local inverse roots, rather than simply applying unguided fixed-point iteration.

The straightness measurements further support this interpretation. SelFix obtains the lowest DS and the lowest accumulated proxy value $\sum_j \phi^{\text{str}}_{j,W}$ among all compared methods. In particular, compared with FPI, SelFix reduces $\sum_j \phi^{\text{str}}_{j,W}$ from 0.1855 to 0.1505, which is expected since the proposed update explicitly biases the fixed-point iteration toward roots close to the straightness anchor. This reduction is accompanied by a decrease in the trajectory-level straightness measure DS from 0.2080 to 0.1702. That is, although $\phi^{\text{str}}_{j,W}$ is a proxy, its reduction is well aligned with the observed decrease in DS. Taken together, the improvement in both reconstruction metrics and straightness metrics validates our method and supports the hypothesis that selecting a straighter inverse trajectory helps reduce error accumulation during reconstruction.

5.2 Image Editing

Evaluation We evaluate SelFix on the PIE-Bench [15] prompt-based image editing task. In this task, we first invert a source image into noise and then generate an edited image from the inverted noise using a target prompt. We use the same editing strategy across all inversion methods to ensure a fair comparison. We report Structure Distance (DINO self-similarity), background preservation (PSNR, LPIPS, and SSIM), and CLIP similarity on the whole image and edited region (Tab. 2). Since source preservation and target-prompt alignment exhibit a trade-off [7], we plot CLIP similarity against background PSNR in Fig. 5 to compare their preservation-alignment trade-offs. We further analyze this trade-off under different editing strategies in App. F.3.

Results As shown in Tab. 2, SelFix achieves the strongest source preservation among all compared methods. This is consistent with the reconstruction results: by selecting a straighter inverse trajectory, SelFix obtains a more faithful inverse noise, which provides a better starting point for preserving

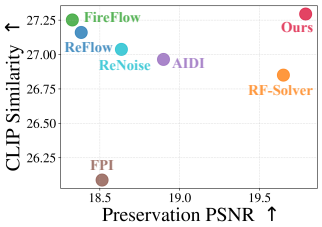


Figure 5: Source-preservation and prompt-alignment trade-off on PIE-Bench editing.

Table 2: Quantitative prompt-based editing results on PIE-Bench.

Method	Structure Dist. ↓	Background Preservation			CLIP Sim. ↑	
		PSNR ↑	LPIPS ↓	SSIM ↑	Whole	Edited
ReFlow	0.0657	18.38	0.2329	0.7204	27.16	23.85
RF-Solver	0.0525	19.65	0.1992	0.7527	26.85	23.64
FireFlow	0.0629	18.33	0.2218	0.7305	27.25	**23.98**
ReNoise	0.0530	18.64	0.1943	0.7449	27.04	23.71
AIDI	0.0505	18.90	0.1943	0.7581	26.96	23.64
FPI	0.0570	18.51	0.1830	0.7482	26.09	22.95
SelFix (Ours)	**0.0457**	**19.79**	**0.1767**	**0.7748**	**27.29**	23.87

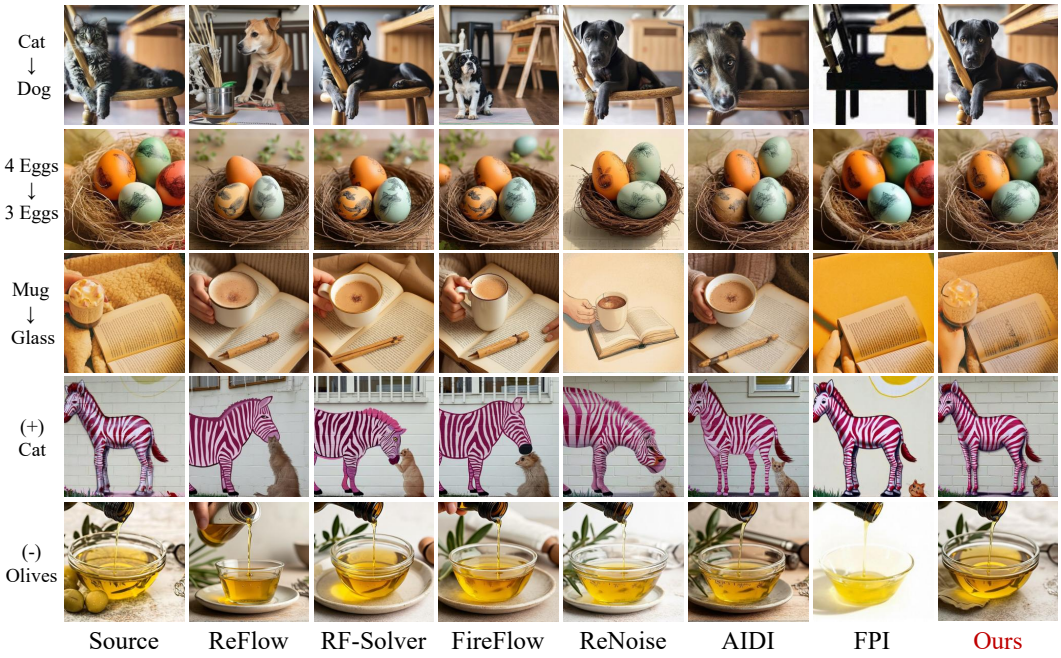


Figure 6: Qualitative prompt-based editing results on PIE-Bench. SelFix better preserves background while applying the requested semantic edit. We also provide more samples in App. F.5.

source content during editing. SelFix obtains the lowest structure distance and the best background-preservation scores, while remaining competitive in target-prompt alignment. Overall, it improves the preservation-editing trade-off, as shown in Fig. 5.

Qualitative results in Fig. 6 show the same trend. SelFix better preserves background details and global structure while still applying the requested edit. For example, in the second row, SelFix preserves the nest shape and background while correctly changing four eggs to three. In contrast, baselines either alter the global image structure or fail to render the requested number of eggs.

5.3 Ablation Study

Guiding Mechanism Tab. 3 compares SelFix with alternative ways of injecting the straightness anchor. Specifically, we compare SelFix with variants using either constant scheduling or gradient-based correction to minimize MSE to the anchor. As can be seen, all variants produce degraded results in terms of PSNR. While blending the straightness anchor with a large constant α or using a gradient-based update with a large learning rate can lead to even smaller DS, they bias the fixed-point equation too much and result in inaccurate inversion. Conversely, when hyperparameters are tuned to match DS across methods, we see that SelFix performs the best. We attribute this to SelFix's vanishing anchor schedule, which simultaneously guides the iteration toward a straighter trajectory and produces an accurate fixed point.

Table 3: Ablation on guiding mechanism.

Method	PSNR ↑	DS ↓
Iteration with constant α (0.5)	15.53	**0.127**
Iteration with constant α (0.0068)	27.86	0.170
Gradient descent (lr=0.1)	19.56	0.157
Gradient descent (lr=0.01)	27.81	0.170
SelFix (Ours)	**28.29**	0.170

Table 4: Ablation on the choice of selector anchor.

Selector Anchor	LPIPS ↓	SSIM ↑	PSNR ↑
Zero	0.6974	0.3395	13.04
Random	0.1779	0.7598	23.69
Straightness (Ours)	**0.0742**	**0.8714**	**28.29**

Table 5: Hyperparameter ablations.

(a) Momentum coefficient μ.

Decoupled?	μ	PSNR ↑	DS ↓
✓	0.75	26.93	**0.126**
✓	**0.5**	**28.29**	0.170
✓	0.25	23.71	0.180
×	0.5	27.54	0.170
–	0.0	16.77	0.204

(b) Decay δ.

δ	PSNR ↑	DS ↓
1	27.29	**0.1678**
1/2	27.60	0.1689
1/4	28.21	0.1700
1/8	**28.29**	0.1702
1/16	27.98	0.1700

(c) Window W.

W	PSNR ↑	DS ↓
1	**28.29**	0.1702
2	**28.29**	0.1701
3	28.26	0.1699
4	28.17	**0.1697**

Selector Choice We examine the importance of the straightness selector by replacing its anchor with zero or random anchors while keeping the remaining inversion procedure unchanged. As shown in Tab. 4, both alternatives substantially degrade reconstruction performance, with PSNR dropping from 28.29 to 13.04 and 23.69, respectively. These results indicate that the improvement of SelFix arises not simply from the anchored update itself, but from carefully choosing an anchor informed by trajectory straightness.

Hyperparameters Tab. 5 ablates the main hyperparameters of SelFix. Momentum is critical for finite-iteration convergence: removing it drops PSNR to 16.77, while $\mu = 0.5$ gives the best reconstruction. Smaller momentum insufficiently smooths the fixed-point correction, whereas larger momentum slows convergence and can leave the trajectory overly anchor-biased, yielding lower DS but worse PSNR. Decoupled momentum also outperforms coupled momentum, supporting our choice to smooth only the fixed-point term while applying the straightness anchor directly. For the anchor schedule, $\delta = 1/8$ performs best: too fast decay gives the anchor too little time to steer root selection, while too slow decay leaves a non-negligible anchor term near the end of the inner loop and can hinder accurate inversion.

Finally, short straightness windows work best. Although larger W slightly reduces DS, it relies on longer recovered trajectory segments that may already contain accumulated numerical or model errors. We therefore use the local setting $W = 1$ by default, which provides a reliable constant-velocity estimate and the best reconstruction accuracy.

Robustness to T2I Backbone and Dataset To examine the generality of SelFix beyond FLUX.1-dev and PIE-Bench, we evaluate it with a different backbone (Stable Diffusion 3.5 Large [9]) and on a different dataset (COCO [19]). SelFix consistently outperforms all compared inversion baselines in LPIPS and PSNR, and also achieves the best source preservation PSNR with comparable target CLIP similarity in the SD3.5 editing experiment, reproducing the trend observed with FLUX.1-dev and PIE-Bench. Results and corresponding experiment details are shown in App. F.1.

6 Conclusion

We presented SelFix, a straightness-guided fixed-point inversion method for rectified flows that reframes inversion as root selection over the exact local inverse set. By deriving a causal straightness anchor from the inverse trajectory and using a vanishing anchored iteration with decoupled momentum, SelFix selects a straighter exact inverse root while preserving the original fixed-point equation asymptotically. Experiments on reconstruction and prompt-based editing show that this selection principle improves reconstruction fidelity, trajectory straightness, and source preservation over solver-based and fixed-point baselines, suggesting that trajectory geometry is an effective criterion for resolving multi-root ambiguity in rectified flow inversion.

Acknowledgments and Disclosure of Funding

This work was in part supported by the National Research Foundation of Korea (RS-2024-00351212 and RS-2024-00436165), and the Institute of Information & communications Technology Planning & Evaluation (IITP) (RS-2024-00509279, RS-2022-II220926, RS-2022-II220959, RS-2026-25518317, and RS-2019-II190075).

References

[1] Heinz H Bauschke. 1996. The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space. *Journal of Mathematical Analysis and Applications*, 202(1):150–159.

[2] Black Forest Labs. 2024. Announcing Black Forest Labs. Blog post on Black Forest Labs website. Accessed: 2025-05-14.

[3] Manuel Brack, Felix Friedrich, Katharina Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinário Passos. 2024. LEDITS++: Limitless Image Editing Using Text-to-Image Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 8861–8870. IEEE.

[4] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. 2023. MasaCtrl: Tuning-Free Mutual Self-Attention Control for Consistent Image Synthesis and Editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22560–22570.

[5] Yeonwoo Cha, Jaehoon Yoo, Semin Kim, Yunseo Park, Jinhyeon Kwon, and Seunghoon Hong. 2026. Training-Free Refinement of Flow Matching with Divergence-based Sampling. *arXiv preprint arXiv:2604.04646*.

[6] A Cuntavepanit and B Panyanak. 2011. Strong convergence of modified Halpern iterations in CAT (0) spaces. *Fixed Point Theory and Applications*, 2011(1):869458.

[7] Yingying Deng, Xiangyu He, Changwang Mei, Peisong Wang, and Fan Tang. 2025. FireFlow: Fast Inversion of Rectified Flow for Image Semantic Editing. In *Forty-second International Conference on Machine Learning*.

[8] Wenkai Dong, Song Xue, Xiaoyue Duan, and Shumin Han. 2023. Prompt Tuning Inversion for Text-Driven Image Editing Using Diffusion Models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 7396–7406. IEEE.

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. 2024. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis.

[10] Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. 2024. Renoise: Real image inversion through iterative noising. In *European Conference on Computer Vision*, pages 395–413. Springer.

[11] Pengsheng Guo and Alex Schwing. 2025. Variational Rectified Flow Matching. In *Forty-second International Conference on Machine Learning*.

[12] Benjamin Halpern. 1967. Fixed points of nonexpanding maps. *Bulletin of the American Mathematical Society*, 73(6):957 – 961.

[13] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2023. Prompt-to-Prompt Image Editing with Cross-Attention Control. In *The Eleventh International Conference on Learning Representations*.

[14] Guanlong Jiao, Biqing Huang, Kuan-Chieh Jackson Wang, and Renjie Liao. 2026. UniEdit-Flow: Unleashing Inversion and Editing in the Era of Flow Models. In *The Fourteenth International Conference on Learning Representations*.

[15] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. 2024. PnP Inversion: Boosting Diffusion-based Editing with 3 Lines of Code. *International Conference on Learning Representations (ICLR)*.

[16] Semin Kim, Jaehoon Yoo, Jinwoo Kim, Yeonwoo Cha, Saehoon Kim, and Seunghoon Hong. 2024. Simulation-free training of neural odes on paired data. *Advances in Neural Information Processing Systems*, 37:60212–60236.

[17] Black Forest Labs. 2025. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`.

[18] Sangyun Lee, Beomsu Kim, and Jong Chul Ye. 2023. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pages 18957–18973. PMLR.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[20] Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[21] Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. 2023. Negative-prompt Inversion: Fast Image Inversion for Editing with Text-guided Diffusion Models. *CoRR*, abs/2305.16807.

[22] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2023. Null-text Inversion for Editing Real Images using Guided Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 6038–6047. IEEE.

[23] Zhihong Pan, Riccardo Gherardi, Xiufeng Xie, and Stephen Huang. 2023. Effective real image editing with accelerated iterative diffusion inversion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15912–15921.

[24] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. 2023. Zero-shot Image-to-Image Translation. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 11:1–11:11. ACM.

[25] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2024. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. In *The Twelfth International Conference on Learning Representations*.

[26] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T Chen. 2023. Multisample Flow Matching: Straightening Flows with Minibatch Couplings. *ICML 2023*.

[27] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125*.

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

[29] Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. 2024. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint arXiv:2410.10792*.

[30] Dvir Samuel, Barak Meiri, Haggai Maron, Yoad Tewel, Nir Darshan, Shai Avidan, Gal Chechik, and Rami Ben-Ari. 2025. Lightning-Fast Image Inversion and Editing for Text-to-Image Diffusion Models. In *The Thirteenth International Conference on Learning Representations*.

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising Diffusion Implicit Models. In *ICLR*. OpenReview.net.

[32] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. 2023. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*.

[33] Bram Wallace, Akash Gokul, and Nikhil Naik. 2023. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22532–22541.

[34] Chenru Wang, Beier Zhu, and Chi Zhang. 2026. Free Lunch for Stabilizing Rectified Flow Inversion. In *The Fourteenth International Conference on Learning Representations*.

[35] Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. 2025. Taming Rectified Flow for Inversion and Editing. In *International Conference on Machine Learning*, pages 64044–64058. PMLR.

[36] Rainer Wittmann. 1992. Approximation of fixed points of nonexpansive mappings. *Archiv der mathematik*, 58(5):486–491.

A Limitations and Broader Impacts

SelFix is designed as a root-selection mechanism for fixed-point inversion in rectified flows, and therefore inherits the limitations of the underlying generative backbone. For example, the demonstrated editing pipeline may still fail on challenging edits when the backbone model itself has limited generation or instruction-following capability. In addition, our theoretical guarantee is local and asymptotic: it assumes a nonempty local fixed-point set and local nonexpansiveness of the fixed-point map, while practical inversion uses only a finite number of inner iterations. Although we find that a small number of iterations, at most 10 in our experiments, is sufficient for reconstruction and editing, finite-iteration residuals cannot be fully eliminated in practice. A finite-distance counterpart is provided in App. D.6, which shows that an iterate within ε of the selected root attains a residual of at most 2ε and a straightness gap of order ε. Thus, the number of inner iterations may need to be adjusted depending on the compute budget and desired reconstruction accuracy. The theory should therefore be interpreted as characterizing the limiting root selected by the update, rather than guaranteeing exact convergence under every finite compute budget.

Broader Impacts and Safeguards SelFix improves inversion for rectified-flow text-to-image models, which may benefit reconstruction and source-preserving image editing in creative and assistive visual applications. At the same time, improved inversion and editing fidelity may inherit misuse risks from the underlying generative model, such as deceptive image manipulation or unauthorized modification of visual content. These risks are not unique to SelFix, as our method does not introduce a new generative backbone, training dataset, or user-facing deployment system. The method is evaluated using existing pretrained models and benchmarks, and the released code is limited to the inversion algorithm rather than a newly trained image generator. Therefore, safeguards for the underlying generative model, including responsible access policies, content filtering, provenance tracking, and usage restrictions, remain the primary mechanisms for mitigating potential misuse.

B Algorithm

Algorithm 1 states the complete SelFix inversion procedure. The outer loop over inversion steps j builds the straightness anchor $a_{j,W}$ from previously recovered velocities, and the inner fixed-point loop applies the anchored update with decoupled momentum to select the straightness-minimizing root.

C Details for Preliminary Analyses

C.1 Multiplicity of Fixed Points

For the analysis in Fig. 2, we use 16 different initializations, where each point is obtained by adding different Gaussian noise to the previous state x_{j-1}. The purpose of the analysis is to test whether the learned and discretized local inverse equation at a fixed inversion step admits multiple approximate fixed-point roots in practice. We use the FLUX.1-dev [2] model with $T = 40$ denoising steps, and conduct the analysis at the middle of denoising ($j = 20$). To prevent external error (*i.e.*, error from the latent autoencoder or a mismatch between the real image distribution and the model-generated image distribution), we conduct the analysis on a synthesized image. Generation uses the prompt *a corgi wearing a blue bandana sitting on a park path*, and inversion uses $K = 15$ iterations with Anderson acceleration to boost convergence.

At fixed-point iteration k, we report two quantities. The first is the mean fixed-point residual,

$$R_k := \frac{1}{N_{\text{init}}} \sum_{n=1}^{N_{\text{init}}} \left\| P_j(x_{j,n}^k) - x_{j,n}^k \right\|_2 . \tag{27}$$

The second is the mean pairwise distance among the current states,

$$D_k := \frac{2}{N_{\text{init}}(N_{\text{init}} - 1)} \sum_{1 \le n < m \le N_{\text{init}}} \left\| x_{j,n}^k - x_{j,m}^k \right\|_2 . \tag{28}$$

A small residual R_k means that each run is close to satisfying the same learned local inverse equation. If, at the same time, D_k remains much larger than R_k, then the converged states are not merely small

Algorithm 1 SelFix: Straightness-selected fixed-point inversion

Require: Image latent x_0; timesteps $0 = t_0 < t_1 < \cdots < t_N$ with $h_j = t_j - t_{j-1}$; velocity field $u_\theta(\cdot, t, c)$; conditioning c; fixed-point iterations K; window $W > 0$; Schedule $\{\alpha_k\}_{k=1}^{K}$ (Eq. (20)); momentum $\mu \in [0, 1)$.

Ensure: Inverted latent x_N and inversion trajectory $\{x_j\}_{j=0}^{N}$.

for $j = 1, \ldots, N$ **do** ▷ outer loop: inversion steps
  Define $r_j(z) \coloneqq f_{t_j \to t_{j-1}}(z) - x_{j-1}$ and $P_j(z) \coloneqq z - r_j(z)$. ▷ Eqs. (6), (8)
  (Euler reverse step: $P_j(z) = x_{j-1} + h_j\, u_\theta(z, t_j, c)$)
  $m_j \leftarrow \min\{W,\, j-1\}$.
  $z^0 \leftarrow x_{j-1}$.
  $M^0 \leftarrow z^0$.
  if $j \neq 1$ **then**
    $\bar{v}_{j-1}^{W} \leftarrow \big(\sum_{\ell=j-m_j}^{j-1} h_\ell\, \widehat{v}_\ell\big) / \big(\sum_{\ell=j-m_j}^{j-1} h_\ell\big)$. ▷ Eq. (14)
    $a_{j,W} \leftarrow x_{j-1} + h_j\, \bar{v}_{j-1}^{W}$. ▷ Eq. (16)
  end if
  for $k = 1, \ldots, K$ **do** ▷ inner loop: fixed-point iteration
    $q^k \leftarrow P_j(z^{k-1})$.
    $M^k \leftarrow \mu M^{k-1} + (1-\mu)\, q^k$. ▷ Eq. (24)
    if $j = 1$ **then**
      $z^k \leftarrow M^k$.
    else
      $z^k \leftarrow \alpha_k a_{j,W} + (1-\alpha_k) M^k$. ▷ Eq. (25)
    end if
  end for
  $x_j \leftarrow z^K$; $\widehat{v}_j \leftarrow (x_j - x_{j-1})/h_j$.
end for
return x_N and $\{x_j\}_{j=0}^{N}$.

Table 6: Extended analysis of approximate fixed-point multiplicity. Mean fixed-point residual and mean pairwise L_2 distance between iterates from different initializations, at inner fixed-point iterations 1, 8, and 15.

Backbone	Dataset	FPI iteration	Mean residual	Mean pairwise L_2 distance
FLUX.1-dev	PIE-Bench	1	251.03	362.20
		8	0.1759	6.963
		15	0.1638	6.956
SD3.5 Large	PIE-Bench	1	251.39	364.41
		8	2.596	36.61
		15	1.580	36.70
FLUX.1-dev	COCO	1	251.08	362.11
		8	0.1733	5.738
		15	0.1617	5.731

numerical perturbations of a single point. Instead, they are distinct approximate fixed-point roots of the same learned and discretized local equation. This is the empirical phenomenon shown in Fig. 2.

Extended Analysis Across Backbones and Datasets To strengthen the observation in Fig. 2, we extend the analysis from a single illustrative example to the first 20 PIE-Bench prompts with both FLUX.1-dev and SD3.5 Large, and further repeat it on COCO with FLUX.1-dev. For each input, we run the same local fixed-point iteration from multiple initializations and measure the mean residual and mean pairwise distance between the resulting iterates. The results are shown in Tab. 6.

Across all three settings, the fixed-point residual decreases by more than two orders of magnitude, whereas the pairwise distance between solutions remains large and nearly unchanged in the late iterations. Thus, the converged states are distinct approximate solutions of the same learned, discretized local inverse equation, indicating that the observed approximate-root multiplicity is not specific to a single example, backbone, or dataset.

C.2 Greedy Multi-Start Straightness Selection

This section describes the FPI+Selection sanity check shown in Fig. 3b. This is not the proposed method, but a diagnostic experiment used to test whether selecting a straighter fixed-point trajectory can improve reconstruction.

The FPI+Selection variant performs best-of-N selection at each denoising step. Given the previously selected states, we generate $N_{\text{cand}} = 32$ candidate inverse states by running independent fixed-point iterations from differently perturbed initializations, using the same fixed-point budget as the naive FPI baseline. The perturbation distribution is

$$x^0_{\text{cand},n} = x^0_{\text{base}} + \sigma_{\text{cand}}\epsilon_n, \qquad \epsilon_n \sim \mathcal{N}(0, I), \qquad n = 1, \dots, N_{\text{cand}}. \tag{29}$$

The perturbation scale σ_{cand} is set to one, and this injection is applied at every denoising step, at the start of the inner fixed-point iteration. After obtaining the candidate states, we evaluate the causal straightness proxy $\phi^{\text{str}}_{j,W}$ in Eq. (15) for each candidate and select the one with the smallest value. The selected state is then used to proceed to the next denoising step. The resulting algorithm is a zero-order greedy search, provided as empirical motivation for root selection. Although inefficient and imperfect, this search illustrates that, among multiple feasible fixed-point inversion paths, choosing a straighter path can help produce a better reconstruction.

D Proofs and Mathematical Details

D.1 Notation and Standard Convergence Facts

All proof arguments are local to one inversion step. Fix $j \in \{1, \dots, T\}$. To avoid conflict with the total number of timesteps T, we write

$$P := P_j, \qquad C := C_j, \qquad S := S_j^C = \text{Fix}(P_j) \cap C_j, \qquad a := a_{j,W}. \tag{30}$$

The step index j is omitted whenever this causes no ambiguity. Under Assumption 1, C is closed and convex, $P(C) \subset C$, P is nonexpansive on C, and S is nonempty. Since the fixed-point set of a nonexpansive map on a closed convex subset of a Hilbert space is closed and convex, S is closed and convex. Hence the metric projection $\text{proj}_S(a)$ is well-defined and unique.

We use the standard projection characterization: for a nonempty closed convex set S and $q = \text{proj}_S(a)$,

$$\langle a - q, y - q\rangle \le 0, \qquad \forall y \in S. \tag{31}$$

All convergence is in the Euclidean norm of the latent space. The full inverse trajectory is obtained by applying the same per-step argument sequentially for $j = 1, \dots, T$.

For convenience, we also record the two standard convergence theorems used in the proofs below.

Theorem 3 (Halpern iteration; Bauschke 1996, Sec. 3, Thm. 3.1, single-map case)**.** *Let H be a real Hilbert space, $C \subset H$ a nonempty closed convex set, and $T : C \to C$ a nonexpansive map with $\text{Fix}(T) \neq \emptyset$. Let $a, x^0 \in C$, and let $(\lambda_k) \subset [0, 1]$ satisfy*

$$\lambda_k \to 0, \qquad \sum_{k=1}^{\infty} \lambda_k = \infty, \qquad \sum_{k=1}^{\infty} |\lambda_{k+1} - \lambda_k| < \infty. \tag{32}$$

Then the sequence defined by

$$x^{k+1} = \lambda_{k+1} a + (1 - \lambda_{k+1}) T(x^k) \tag{33}$$

converges strongly to $\text{proj}_{\text{Fix}(T)}(a)$ [1].

Theorem 4 (Modified Halpern iteration; Cuntavepanit–Panyanak 2011, Sec. 3, Thm. 3.2)**.** *Let C be a nonempty closed convex subset of a complete CAT(0) space, let $T : C \to C$ be nonexpansive with $F(T) \neq \emptyset$, and let $u \in C$, $x^1 \in C$. Let $(\lambda_n), (\tau_n) \subset (0, 1)$ satisfy*

$$\lambda_n \to 0, \qquad \sum_{n=1}^{\infty} \lambda_n = \infty, \qquad 0 < \liminf_{n\to\infty} \tau_n \le \limsup_{n\to\infty} \tau_n < 1. \tag{34}$$

Then the sequence defined by

$$x^{n+1} = \tau_n x^n \oplus (1 - \tau_n)\big(\lambda_n u \oplus (1 - \lambda_n) T x^n\big) \tag{35}$$

converges to the point of $F(T)$ nearest to u [6]. In a Euclidean or Hilbert space, $\oplus$ denotes the usual convex combination, $F(T) = \text{Fix}(T)$, and the nearest point is $\text{proj}_{\text{Fix}(T)}(u)$.

D.2 Algebra of the Straightness Selector

This subsection expands the straightness selector used in Sec. 3. The window size W denotes the number of previous velocities used to estimate the local mean velocity. For steps where the full window is available,

$$\bar{v}_{j-1}^{W} := \frac{\sum_{i=1}^{W} h_{j-i} v_{j-i}}{\sum_{i=1}^{W} h_{j-i}}, \qquad v_{j-i} := \frac{x_{j-i} - x_{j-i-1}}{h_{j-i}}. \tag{36}$$

For early inversion steps where not enough previous velocities are available, the implementation includes as many velocities as possible from the stored trajectories.

For a candidate current state x at time t_j, the candidate velocity is

$$v_j(x) := \frac{x - x_{j-1}}{h_j}. \tag{37}$$

The causal local straightness selector is

$$\phi_{j,W}^{\mathrm{str}}(x) := h_j \left\| v_j(x) - \bar{v}_{j-1}^{W} \right\|_2^2. \tag{38}$$

Substituting $v_j(x)$ gives

$$\phi_{j,W}^{\mathrm{str}}(x) = h_j \left\| \frac{x - x_{j-1}}{h_j} - \bar{v}_{j-1}^{W} \right\|_2^2 \tag{39}$$

$$= \frac{1}{h_j} \left\| x - \left(x_{j-1} + h_j \bar{v}_{j-1}^{W} \right) \right\|_2^2. \tag{40}$$

Thus the straightness anchor is

$$a_{j,W} := x_{j-1} + h_j \bar{v}_{j-1}^{W}, \tag{41}$$

and the selector is exactly

$$\phi_{j,W}^{\mathrm{str}}(x) = \frac{1}{h_j} \left\| x - a_{j,W} \right\|_2^2. \tag{42}$$

Since $h_j > 0$, minimizing $\phi_{j,W}^{\mathrm{str}}$ over any set is equivalent to minimizing the squared distance to $a_{j,W}$ over the same set. In particular,

$$\arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x) = \arg\min_{x \in S_j^C} \left\| x - a_{j,W} \right\|_2^2 = \left\{ \mathrm{proj}_{S_j^C}(a_{j,W}) \right\}. \tag{43}$$

This identity is the algebraic link between causal straightness minimization and projection-selected fixed-point inversion.

D.3 Verification of the Implemented Anchor Schedule

In this work, we use the following anchor schedule, identical to Eq. (20):

$$\alpha_k = \alpha_1 \frac{\delta}{(k-1) + \delta}, \qquad k \geq 1, \;\; 0 < \alpha_1 < 1, \;\; \delta > 0. \tag{44}$$

Below we show the schedule satisfies the Halpern conditions in Eq. (48). We can easily see $\alpha_k \in (0, 1)$ and $\alpha_k \to 0$. Moreover,

$$\sum_{k=1}^{\infty} \alpha_k = \alpha_1 \delta \sum_{k=1}^{\infty} \frac{1}{(k-1) + \delta} = \infty, \tag{45}$$

because the sum is harmonic up to a finite shift. Finally, (α_k) is monotone decreasing, so

$$\sum_{k=2}^{\infty} |\alpha_k - \alpha_{k-1}| = \alpha_1 - \lim_{k \to \infty} \alpha_k = \alpha_1 < \infty. \tag{46}$$

D.4 Proof of Theorem 1

Theorem (Restatement of Theorem 1)**.** *Under Assumption 1 and with α_k satisfying Eq.* (19)*, the anchored fixed-point iteration*

$$x_j^k = \alpha_k a_{j,W} + (1-\alpha_k) P_j(x_j^{k-1})$$

converges to

$$x_j^\star \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x).$$

Proof. Fix an inversion step j and use the shorthand in Eq. (30). Under Assumption 1, C is closed and convex, $P(C) \subset C$, P is nonexpansive on C, $a, x^0 \in C$, and $S = \mathrm{Fix}(P) \cap C$ is nonempty. Hence $\mathrm{proj}_S(a)$ is well-defined and unique.

The anchored fixed-point iteration in the main text can be written as

$$x^k = \alpha_k a + (1-\alpha_k) P(x^{k-1}), \qquad k \geq 1, \tag{47}$$

where the schedule (α_k) satisfies

$$\alpha_k \in (0,1), \qquad \alpha_k \to 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=2}^{\infty} |\alpha_k - \alpha_{k-1}| < \infty \tag{48}$$

by Eq. (19). Applying Theorem 3 with $T = P$, anchor $a = a_{j,W}$, and $\lambda_k = \alpha_k$, we obtain

$$x^k \to q := \mathrm{proj}_S(a). \tag{49}$$

By Eq. (42),

$$\phi_{j,W}^{\mathrm{str}}(x) = \frac{1}{h_j} \|x - a_{j,W}\|_2^2, \tag{50}$$

so minimizing $\phi_{j,W}^{\mathrm{str}}$ over S_j^C is identical to projecting $a_{j,W}$ onto S_j^C. Hence

$$q = \mathrm{proj}_{S_j^C}(a_{j,W}) \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x). \tag{51}$$

The limit is therefore an exact local inverse root selected by the causal straightness criterion. This proves Theorem 1. □

D.5 Proof of Theorem 2

Theorem (Restatement of Theorem 2)**.** *Under the same assumptions as Theorem 1, with α_k in Eq.* (20) *and momentum weight $\mu \in [0,1)$, the decoupled-momentum iteration*

$$M_j^k = \mu M_j^{k-1} + (1-\mu) P_j(x_j^{k-1}),$$
$$x_j^k = \alpha_k a_{j,W} + (1-\alpha_k) M_j^k,$$

initialized with $M_j^0 = x_j^0$, satisfies

$$x_j^k \to x_j^\star, \qquad x_j^\star \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x).$$

Proof. The strategy is to eliminate the momentum variable for $k \geq 1$ using the anchored relation $x^k = \alpha_k a + (1-\alpha_k) M^k$, recast the resulting one-state recursion in the modified-Halpern form of Theorem 4, and verify its coefficient hypotheses. The case $\mu = 0$ reduces to Theorem 1, so we focus on $0 < \mu < 1$.

Fix j and write $P := P_j$, $C := C_j$, $S := S_j^C$, $a := a_{j,W}$. The implemented recursion is

$$M^{k+1} = \mu M^k + (1-\mu) P(x^k), \qquad x^k = \alpha_k a + (1-\alpha_k) M^k, \qquad M^0 = x^0 \in C, \tag{52}$$

with $\mu \in [0,1)$. Since C is convex, $a \in C$, and $P(C) \subset C$, induction gives $M^k, x^k \in C$ for all k.

For $k \geq 1$, the anchored relation inverts to

$$M^k = \frac{x^k - \alpha_k a}{1 - \alpha_k}, \qquad k \geq 1. \tag{53}$$

Substituting Eq. (53) into the momentum and anchor updates gives

$$x^{k+1} = \eta_k a + \rho_k x^k + \sigma_k P(x^k), \qquad k \geq 1, \tag{54}$$

with

$$\rho_k = \frac{\mu(1 - \alpha_{k+1})}{1 - \alpha_k}, \quad \sigma_k = (1 - \mu)(1 - \alpha_{k+1}), \quad \eta_k = \alpha_{k+1} - \frac{\mu(1 - \alpha_{k+1})\alpha_k}{1 - \alpha_k}, \tag{55}$$

summing to $\eta_k + \rho_k + \sigma_k = 1$. Setting $b_k := \rho_k$ and $\gamma_k := \eta_k/(1 - \rho_k)$, this regroups as

$$x^{k+1} = b_k\, x^k + (1 - b_k)\big(\gamma_k a + (1 - \gamma_k)P(x^k)\big), \tag{56}$$

which matches Theorem 4 with $\tau_n \leftarrow b_k$, $\lambda_n \leftarrow \gamma_k$, anchor $u = a$, and map $T = P$ in Euclidean space.

It remains to verify that $b_k, \gamma_k \in (0, 1)$ for all sufficiently large k and that they satisfy the hypotheses of Theorem 4. The explicit schedule $\alpha_k = \alpha_1 \delta/((k - 1) + \delta)$ with $0 < \alpha_1 < 1$ and $\delta > 0$ has three structural properties that make this verification immediate:

(i) α_k is strictly decreasing with $0 < \alpha_k \leq \alpha_1 < 1$ for all $k \geq 1$, and thus $1 - \alpha_k \geq 1 - \alpha_1 > 0$;

(ii) $\alpha_k \to 0$;

(iii) $\alpha_{k+1}/\alpha_k = (k - 1 + \delta)/(k + \delta) \to 1$.

Combining (ii) and (iii),

$$b_k = \mu\, \frac{1 - \alpha_{k+1}}{1 - \alpha_k} \;\longrightarrow\; \mu, \qquad \frac{\eta_k}{\alpha_k} = \frac{\alpha_{k+1}}{\alpha_k} - \mu\, \frac{1 - \alpha_{k+1}}{1 - \alpha_k} \;\longrightarrow\; 1 - \mu. \tag{57}$$

Both limits lie strictly inside $(0, 1)$ because $\mu \in (0, 1)$, so $b_k \in (0, 1)$ and $\eta_k > 0$ hold for all sufficiently large k. Together with $\sigma_k = (1 - \mu)(1 - \alpha_{k+1}) > 0$ (always) and $1 - \rho_k = \eta_k + \sigma_k$, this also gives $\gamma_k = \eta_k/(1 - \rho_k) \in (0, 1)$ eventually. Discarding the finite prefix we have $b_k, \gamma_k \in (0, 1)$ for all sufficiently large k. Moreover,

$$\frac{\gamma_k}{\alpha_k} = \frac{\eta_k/\alpha_k}{1 - \rho_k} \;\longrightarrow\; \frac{1 - \mu}{1 - \mu} = 1, \tag{58}$$

so $\gamma_k \sim \alpha_k$. Therefore $0 < \liminf_k b_k = \limsup_k b_k = \mu < 1$, $\gamma_k \to 0$, and $\sum_k \gamma_k = \infty$ since $\sum_k \alpha_k = \infty$ (App. D.3).

Theorem 4 applied to Eq. (56) therefore gives $x^k \to q := \mathrm{proj}_S(a)$, and Eq. (53) together with $\alpha_k \to 0$ gives $M^k \to q$ as well. The quadratic selector identity Eq. (42) then yields

$$q = \mathrm{proj}_{S_j^C}(a_{j,W}) \in \arg\min_{x \in S_j^C} \phi_{j,W}^{\mathrm{str}}(x), \tag{59}$$

completing the proof. □

D.6 Finite-Distance Bound for Approximate Iterates

While our convergence guarantees are asymptotic, in practice the fixed-point iteration runs for only finitely many steps, and the recovered iterate retains a nonzero residual. The following remark shows that, even in this finite regime, the distance from the iterate to the selected fixed point controls both the residual and the gap to the straightness optimum.

Remark (Finite-distance bound for approximate iterates)**.** *Let q_j denote the selected fixed point, and let x_j^k be a finite inner iterate lying in the local region where P_j is nonexpansive. Define*

$$d_k := \|x_j^k - q_j\|, \qquad R_j := \|q_j - a_{j,W}\|. \tag{60}$$

Since $q_j \in \mathrm{Fix}(P_j)$, the residual satisfies

$$\|r_j(x_j^k)\| = \|P_j(x_j^k) - x_j^k\| \le \|P_j(x_j^k) - P_j(q_j)\| + \|x_j^k - q_j\| \le 2d_k. \tag{61}$$

Moreover, for the straightness selector

$$\phi_{j,W}^{\mathrm{str}}(x) = \frac{1}{h_j}\|x - a_{j,W}\|^2, \tag{62}$$

we have

$$\phi_{j,W}^{\mathrm{str}}(x_j^k) - \phi_{j,W}^{\mathrm{str}}(q_j) = \frac{1}{h_j}\left(2\langle q_j - a_{j,W},\, x_j^k - q_j\rangle + \|x_j^k - q_j\|^2\right), \tag{63}$$

and therefore

$$\phi_{j,W}^{\mathrm{str}}(x_j^k) \le \min_{s \in S_j^C} \phi_{j,W}^{\mathrm{str}}(s) + \frac{2R_j d_k + d_k^2}{h_j}. \tag{64}$$

In particular, if $d_k \le \varepsilon$, then

$$\|r_j(x_j^k)\| \le 2\varepsilon, \qquad \phi_{j,W}^{\mathrm{str}}(x_j^k) \le \min_{s \in S_j^C} \phi_{j,W}^{\mathrm{str}}(s) + \frac{2R_j\varepsilon + \varepsilon^2}{h_j}. \tag{65}$$

Hence finite iterates sufficiently close to the selected fixed point are both approximately feasible and near-optimal in straightness.

D.7 Finite-iteration Role of Decoupled Momentum

We first provide an intuitive local explanation of how decoupled momentum can suppress oscillatory modes in the finite-iteration regime. The key intuition is that momentum retains a component of the current iterate, which can partially cancel a fixed-point proposal pointing in the opposite direction.

For one inversion step, let P be the local fixed-point map, a the fixed straightness anchor, and x^k the k-th iterate. Eliminating the momentum variable from Eqs. (24)–(25) gives

$$x^{k+1} = \eta_k a + \rho_k x^k + \sigma_k P(x^k), \tag{66}$$

where $\rho_k = \frac{\mu(1-\alpha_{k+1})}{1-\alpha_k}, \sigma_k = (1-\mu)(1-\alpha_{k+1}), \eta_k = 1 - \rho_k - \sigma_k$.

Let q be a fixed point satisfying $P(q) = q$, and define the current error and fixed-point proposal relative to q as $e_k := x^k - q$ and $p_k := P(x^k) - q$. Then,

$$e_{k+1} = \eta_k(a - q) + \rho_k e_k + \sigma_k p_k, \tag{67}$$

where the term $\rho_k e_k$ is present only when momentum is used ($\mu > 0$). Therefore, when p_k and e_k point in opposing directions, $\rho_k e_k$ can partially cancel the $\sigma_k p_k$ term, reducing the oscillatory error-dependent component.

To illustrate this mechanism, consider a local oscillatory case in which $p_k = -e_k$. Without momentum, the error-dependent component becomes

$$\rho_k e_k + \sigma_k p_k = -(1 - \alpha_{k+1})e_k. \tag{68}$$

With $\mu = 0.5$, it becomes

$$\rho_k e_k + \sigma_k p_k = (1 - \alpha_{k+1})\frac{\alpha_k}{2(1-\alpha_k)}e_k. \tag{69}$$

Thus, momentum removes the sign reversal of this component and reduces its magnitude to a fraction

$$\frac{\alpha_k}{2(1-\alpha_k)} \le \frac{1}{2} \tag{70}$$

of the no-momentum magnitude, since our schedule satisfies $\alpha_k \le 1/2$. This illustrates how momentum can suppress sign-alternating error propagation near a fixed point.

Table 7: Oscillation analysis for non-momentum and momentum updates.

Metric	**Definition**	$\mu = 0$	$\mu = 0.5$
Update flip rate	$\mathbb{E}[\mathbf{1}(c_k < 0)]$	0.968	0.336 (-65.3%)
Negative cosine magnitude	$\mathbb{E}[\max(0, -c_k)]$	0.699	0.082 (-88.3%)
Last-iteration residual RMS	$\mathrm{RMS}(P(x^{K-1}) - x^{K-1})$	0.00792	0.00370 (-53.2%)

Table 8: Default hyperparameters for the main reconstruction experiments.

Quantity	**Value**
Backbone	FLUX.1-dev
Number of timesteps T	15
Base solver	Euler
Fixed-point iterations K	10
Previous-velocity window size W	1
Momentum coefficient μ	0.5
Initial anchor weight α_1	0.5
Decay parameter δ	$1/8$
Guidance scale	1

Numerical Results On top of the intuitive explanation, we empirically examine whether the predicted reduction in oscillatory behavior appears in the actual inner-loop iterates. Let $d_k := x^{k+1} - x^k$, and define the cosine similarity between consecutive movements as

$$c_k := \frac{\langle d_k, d_{k-1} \rangle}{\|d_k\|_2 \|d_{k-1}\|_2}. \tag{71}$$

Here, $c_k < 0$ indicates that two consecutive updates point in opposing directions.

As shown in Tab. 7, decoupled momentum substantially reduces both the frequency and aggregate magnitude of back-and-forth updates, accompanied by a 53.2% lower finite-iteration residual. These results are consistent with the local oscillation-damping mechanism described above.

E Experimental Details

The default hyperparameters used in the main reconstruction and editing experiments can be found in Tab. 8 and Tab. 9, respectively.

Baselines In our experiments, we compare SelFix with ReFlow, RF-Solver, FireFlow, ReNoise, AIDI, and naive FPI. For fair comparison, we approximately match the total NFE across inversion methods by increasing the number of denoising steps for ReFlow, RF-Solver, and FireFlow, as shown in Tab. 10. In reconstruction, fixed-point inversion methods, including SelFix, use 165 NFEs: 150 for inversion with 15 steps and 10 fixed-point iterations, followed by 15 NFEs for generation. We therefore use 83 steps per direction for ReFlow, 42 steps per direction for RF-Solver, and 83 steps per direction for FireFlow, resulting in 166, 168, and 168 total NFEs, respectively. In editing, fixed-point inversion methods use 75 NFEs: 60 for inversion with 15 steps and 4 iterations, followed by 15 NFEs for generation. We accordingly use 38 steps per direction for ReFlow and FireFlow, and 19 steps per direction for RF-Solver, resulting in 76, 78, and 76 total NFEs, respectively.

To implement ReNoise on the Flux backbone, we try various settings for the accumulation weighting scheme and the editability enhancement step, and report the optimal setting. Specifically, we compare the weighting strategy proposed in the original paper against a uniform weighting scheme. For the editability enhancement step, we try the hyperparameters of the original paper (λ_{pair}: 10, $\lambda_{patch-KL}$: 0.055), but obtain the best result by disabling it. For the editing experiment, which uses a 4-iteration setting, the original paper provides no predefined configuration, so we adopt the optimal setting from the reconstruction experiment. When implementing AIDI, we use the suggested `AIDI_E` variant.

Editing Pipeline For all inversion methods, the downstream editing procedure is kept fixed so that the comparison isolates the inversion method. Our implementation uses value feature injection: the

Table 9: Default hyperparameters for the main editing experiments.

Quantity	Value
Backbone	FLUX.1-dev
Number of timesteps T	15
Base solver	Euler
Fixed-point iterations K	4
Previous-velocity window size W	1
Momentum coefficient μ	0.1
Initial anchor weight α_1	0.5
Decay parameter δ	2
Guidance scale	2

Table 10: NFE budget used for inversion and generation. For fixed-point inversion methods, the inversion cost is computed as number of inversion steps times number of fixed-point iterations.

Experiment	Method	Inversion	Generation	Extra	Total NFE
Reconstruction	Fixed-point inversions	$15 \times 10 = 150$	15	0	165
	ReFlow	83	83	0	166
	RF-Solver	$42 \times 2 = 84$	$42 \times 2 = 84$	0	168
	FireFlow	83	83	2	168
Editing	Fixed-point inversions	$15 \times 4 = 60$	15	0	75
	ReFlow	38	38	0	76
	RF-Solver	$19 \times 2 = 38$	$19 \times 2 = 38$	0	76
	FireFlow	38	38	2	78

Table 11: Hyperparameter settings for the editing task.

Argument	Value
`-start_layer_index`	0
`-end_layer_index`	37
`-inject`	1
`-reuse_v`	1
`-editing_strategy`	`replace_v`
`-qkv_ratio`	`1.0,1.0,1.0`

inverted source trajectory is used as the source trajectory, and the target prompt is used during the editing pass. Feature injection settings are shared across all inversion baselines. We use the official repository of FireFlow [7] to implement the feature injection, with hyperparameters set as in Tab. 11.

E.1 Compute and Runtime

All experiments are conducted in a GPU cloud environment, with each experiment running on a single H200 GPU. When using SelFix for reconstruction and editing tasks, processing a single image takes roughly 11 seconds and 5.5 seconds, respectively.

F Additional Experimental Results

F.1 Robustness to T2I Backbone and Dataset

To examine the generality of SelFix beyond FLUX.1-dev and PIE-Bench, we evaluate it with a different backbone (Stable Diffusion 3.5 Large) and on a different dataset (COCO [19]). For SD3.5 Large, we use the same hyperparameters as the FLUX.1-dev experiments unless otherwise noted. In reconstruction, all hyperparameters follow Tab. 8. In editing, we follow Tab. 9, except that we use μ=0.3 and δ=1, and a guidance scale of 1 for inversion and 3.5 for generation. The NFE budget is matched across methods as in Tab. 10. The results are reported in Tab. 12.

Table 12: Generality of SelFix across backbones and datasets. All methods are compared under matched NFE, using the same baselines as in the main experiments.

(a) PIE-Bench recon. (SD3.5)

Method	LPIPS ↓	PSNR ↑
ReFlow	0.3071	18.60
RF-Solver	0.1426	23.80
FireFlow	0.0910	27.06
ReNoise	0.1335	23.56
AIDI	0.2040	23.12
FPI	0.3253	14.52
SelFix	**0.0561**	**29.42**

(b) PIE-Bench edit. (SD3.5)

Method	CLIP ↑	PSNR ↑
ReFlow	28.21	14.75
RF-Solver	28.22	16.13
FireFlow	28.18	15.04
ReNoise	28.26	15.83
AIDI	28.23	16.90
FPI	27.96	16.46
SelFix	**28.34**	**17.21**

(c) COCO recon. (FLUX.1-dev)

Method	LPIPS ↓	PSNR ↑
ReFlow	0.2748	18.68
RF-Solver	0.2556	19.49
FireFlow	0.1523	23.75
ReNoise	0.1337	21.92
AIDI	0.1590	22.45
FPI	0.3588	13.76
SelFix	**0.0913**	**26.15**

Table 13: Comparison with additional inversion baselines on PIE-Bench reconstruction using FLUX.1-dev.

Method	**LPIPS ↓**	**PSNR ↑**
UniEdit-Flow	0.0971	27.17
ReFlow + PMI	0.2388	19.53
FireFlow + PMI	0.1027	27.05
FPI + PMI	0.3068	15.04
SelFix (Ours)	**0.0742**	**28.29**

For the COCO dataset, we follow the convention of ReNoise [10], using the `val2017` split images whose width and height both exceed 420 pixels (N=3866), paired with the first caption of each image as the prompt. Images are converted to RGB and resized to 512×512 with bicubic interpolation without preserving the aspect ratio. We use the same hyperparameters as the FLUX.1-dev reconstruction setting in Tab. 8.

F.2 Comparison with Additional Baselines

We provide additional comparisons with UniEdit-Flow [14] and PMI [34]. We follow the same evaluation protocol as the main reconstruction experiment, with approximately matched NFE budgets for fair comparison. Since PMI is an auxiliary correction applied to a base inversion method, we evaluate it with three representative methods: ReFlow, FireFlow, and FPI, corresponding to naive reverse integration, an improved solver-based method, and fixed-point inversion, respectively. As shown in Tab. 13, SelFix outperforms all additional baselines in both LPIPS and PSNR.

F.3 Preservation-Alignment Trade-off

SelFix improves the inversion process rather than the downstream editing mechanism. In the main experiments, we use the same editing strategy across all inversion methods to isolate the effect of inversion. More accurate inversion provides a more faithful source trajectory, which primarily benefits source preservation. In contrast, target-prompt alignment depends largely on the downstream editing strategy. Since this strategy is fixed across methods, we expect SelFix to improve source preservation while maintaining comparable target-prompt alignment.

Source preservation and target-prompt alignment exhibit a trade-off [7], as stronger preservation can restrict the changes needed to follow the target prompt. As shown in Fig. 5, SelFix achieves the best source preservation while maintaining comparable target-prompt alignment among the compared methods. To further examine this trade-off, we evaluate SelFix with two additional feature-injection strategies from FireFlow [7], Add Q and Add K, while keeping the inversion process unchanged. The results are reported in Tab. 14.

Consistent with FireFlow, Add Q and Add K improve target-prompt alignment compared to the default Replace V strategy, but substantially reduce source preservation. Specifically, Add Q increases whole-image CLIP similarity from 27.29 to 27.71 while reducing background PSNR from 19.79 to

Table 14: SelFix under different editing strategies on PIE-Bench with FLUX.1-dev. Only the downstream feature-injection strategy is varied; the SelFix inversion remains unchanged.

Editing strategy	Preservation PSNR ↑	CLIP Sim. (Whole) ↑
Add Q	15.34	**27.71**
Add K	15.62	27.59
Replace V (Default)	**19.79**	27.29

15.34. These results illustrate that the preservation-alignment trade-off can be adjusted through the downstream editing strategy without changing SelFix inversion.

F.4 Additional Reconstruction Qualitative Results

We provide more reconstruction examples from PIE-Bench, expanding Fig. 4. The results are in Fig. 7.

F.5 Additional Editing Qualitative Results

We provide more prompt-based editing examples from PIE-Bench, expanding Fig. 6. The results are in Fig. 8.

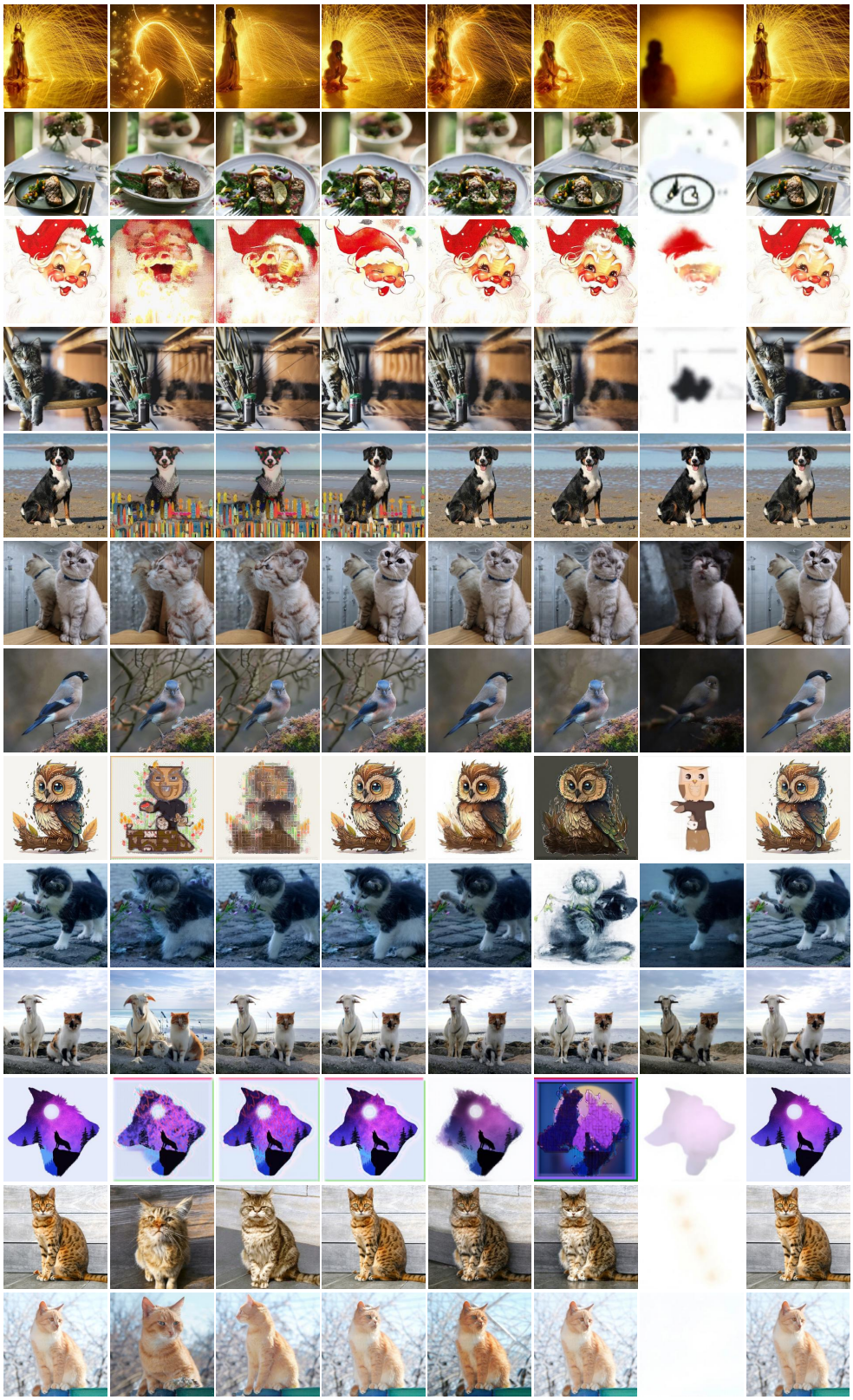


Figure 7: Additional qualitative results for the image reconstruction task.

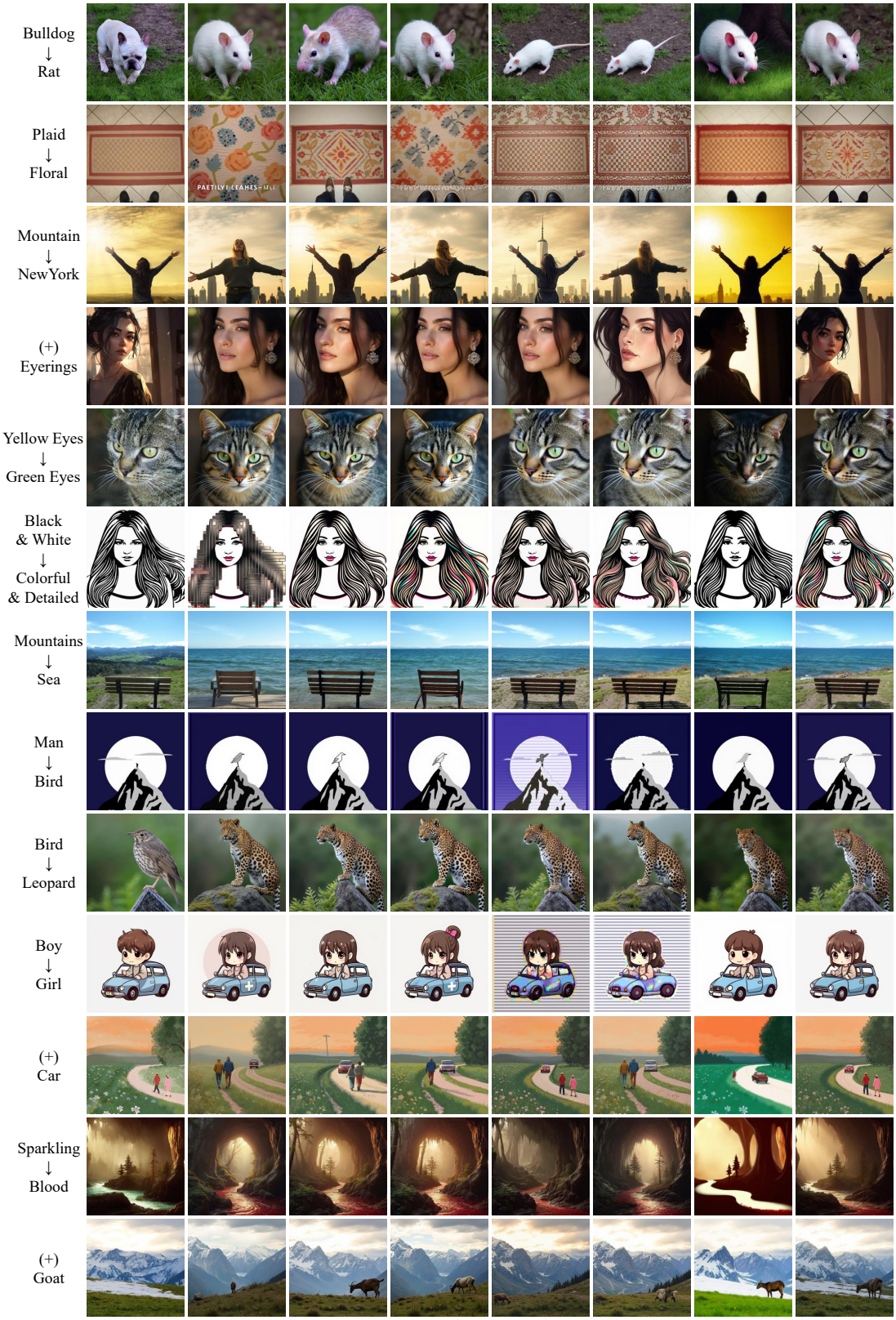


Figure 8: Additional qualitative results for the prompt-based image editing task.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in App. A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the full set of assumptions and proof for theoretical analysis.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the information needed to reproduce our work.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide open access to the code and data, along with the instructions to faithfully reproduce our experiment.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the details are specified in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our inversion framework is deterministic and is a test-time method, where introducing variability is unnatural.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computational cost in App. E.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential societal impacts, including safeguards, in App. A.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We discuss potential societal impacts, including safeguards, in App. A.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited all models, datasets and codebases we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our implementation on GitHub with detailed documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: There was no LLM usage for important, original, or non-standard component of the core methods in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.