FlowSteer: Conditioning Flow Field for Consistent Image Restoration Supplementary Materials

Tharindu Wickremasinghe[1], Chenyang Qi[2], Harshana Weligampola[1], Zhengzhong Tu[3], Stanley H. Chan[1]
[1]Purdue University [2]HKUST [3]Texas A&M University

Contents 1

A Noise Sensitivity of the Flow Model 1
A.1. Noise-robust fidelity update with diffusion 1
A.2 Can this scheme extend to flow models? 2

B More Details on the Flow Model 3
B.1. Feature sharing 3
B.2. Text prompts 3

C Details on the Degradation Models 3

D Design Details for $\{\lambda_i\}$ 3
D.1. A Two-Step Schedule 3
D.2 Effect of the Fidelity update 4
D.3 Final padding 4

E Limitations and Future Directions 4
E.1. Empirical schedule 4
E.2. Artifacts from explicit conditioning 4

F More Visual Results 4
F.1. More visual results on restoration tasks 4
F.2. Plug-and-Play(PnP) on pre-trained Flow models . . 4

G More Quantitative Results 5
G.1. Inversion-free models 5

A. Noise Sensitivity of the Flow Model

In Section 3, we describe that the flow model is sensitive to noisy intermediate projections. This raises the question: "Are there noise-robust algorithms from diffusion models, that can be reinterpreted for the flow-model scheme?" To the best of our knowledge, there are no such algorithms that can be directly translated to the flow scheme. The main reason is that a diffusion model is trained with a noise schedule $\{\sigma_t\}$ for the time schedule $t = t_N, ...t_0$. This acts as an inherent buffer of randomness at each step, and is used to design a weight λ_t in fidelity update steps. To illustrate this, we present the robust version of Algorithm 1 noise and describe why it cannot be applied directly to the flow model.

A.1. Noise-robust fidelity update with diffusion

Fidelity update step. Assume a linear forward model with additive noise $\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma_y^2\mathbf{I})$. The DDNM-style fidelity update[20] for a noisy measurement would update line 4 in Algorithm 1 as follows.

$$\begin{aligned}\widehat{\mathbf{x}}_{0|t} &= \mathbf{A}^\dagger\mathbf{y} + (\mathbf{I} - \mathbf{A}^\dagger\mathbf{A})\,\mathbf{x}_{0|t} \\ &= \mathbf{A}^\dagger(\mathbf{A}\mathbf{x} + \boldsymbol{\eta}) + (\mathbf{I} - \mathbf{A}^\dagger\mathbf{A})\,\mathbf{x}_{0|t} \\ &= \mathbf{x}_{0|t} - \mathbf{A}^\dagger\big(\mathbf{A}\mathbf{x}_{0|t} - \mathbf{y}\big) + \mathbf{A}^\dagger\boldsymbol{\eta}, \quad \boldsymbol{\eta}\sim\mathcal{N}(\mathbf{0}, \sigma_y^2\mathbf{I}),\end{aligned} \tag{5}$$

which makes explicit the extra noise term $\mathbf{A}^\dagger\boldsymbol{\eta}$.

Projection back/ Posterior sampling step. As in DDPM [10] or DDIM [17], a diffusion model samples/project back to the reconstruction path at time step $t-1$. This is line 5 of Algorithm 1:

$$\mathbf{x}_{t-1} \sim \mathcal{N}\Big(\boldsymbol{\mu}_t(\mathbf{x}_t, \widehat{\mathbf{x}}_{0|t}),\, \sigma_t^2\,\mathbf{I}\Big).$$

With the re-parametrization trick,

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_t(\mathbf{x}_t, \widehat{\mathbf{x}}_{0|t}) + \sigma_t\,\boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{6}$$

The posterior mean above is

$$\boldsymbol{\mu}_t(\mathbf{x}_t, \widehat{\mathbf{x}}_{0|t}) = \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}}_{=:\,a_t}\,\widehat{\mathbf{x}}_{0|t} \;+\; \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,\mathbf{x}_t. \tag{7}$$

Damped correction(λ_t) and variance matching(γ_t). Following the "project-and-correct" view, Wang *et al*. [20] propose to parametrize and dampen the update weight $1 \to \lambda_t$ for the data-fidelity (null-space) update, and adjust the diffusion noise by $\sigma_t \to \gamma_t$. This changes lines 4 and 5 of Algorithm 1 to the following:

$$\widehat{\mathbf{x}}_{0|t} = \mathbf{x}_{0|t} - \lambda_t\,\mathbf{A}^\dagger\big(\mathbf{A}\mathbf{x}_{0|t} - \mathbf{y}\big), \tag{8}$$

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_t(\mathbf{x}_t, \widehat{\mathbf{x}}_{0|t}) + \gamma_t\,\boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{9}$$

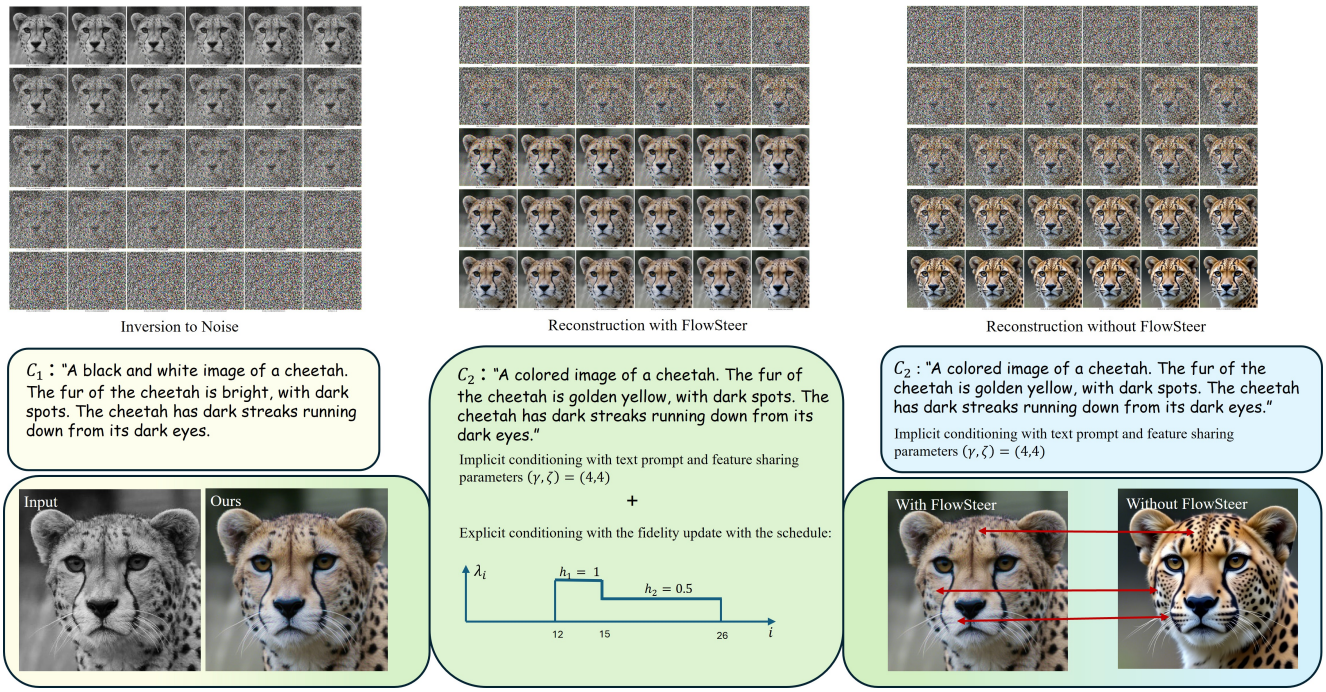

Figure 9. FlowSteer–an overview. Left: The input degraded image is inverted to noise. The source prompt C_1 guides the inversion flow path. Middle: The FlowSteer reconstruction path is conditioned by the target caption C_2, feature sharing parameterized by (γ, ζ), and the fidelity update scheduled by λ_i. Right: The FlowSteer output is compared against our baseline flow model without FlowSteer. The implicit conditioning without FlowSteer generates unnecessary hallucinations. (Zoom in for a better view).

Since $\widehat{\mathbf{x}}_{0|t}$ contains $\mathbf{A}^\dagger\boldsymbol{\eta}$, the sampling mean (7) injects an additional noise term $a_t\lambda_t\mathbf{A}^\dagger\boldsymbol{\eta}$ with covariance $a_t^2\lambda_t^2\sigma_y^2\,\mathbf{A}^\dagger\mathbf{A}^{\dagger *}$. For the simple linear operators used in the linear tasks selected, they are treated as isotropic and set γ_t to preserve the target posterior variance. The two principles that guide the adaptive calculation of the new parameters are as follows:

i) Variance should be preserved in each step t,

$$\gamma_t^2 \;=\; \max\Big(0,\; \sigma_t^2 \;-\; a_t^2\,\lambda_t^2\,\sigma_y^2\Big). \tag{10}$$

ii) λ_t should be as close as possible to 1,

$$\lambda_t \;=\; \begin{cases} 1, & \sigma_t \geq a_t\,\sigma_y, \\ \dfrac{\sigma_t}{a_t\,\sigma_y}, & \sigma_t < a_t\,\sigma_y\;. \end{cases} \tag{11}$$

Interpretation: a_t is the coefficient on $\widehat{\mathbf{x}}_{0|t}$ in the posterior mean, λ_t controls the strength of the fidelity (null-space) correction, σ_y is the measurement-noise std., and γ_t is the residual diffusion noise after accounting for injected measurement noise.

A.2. Can this scheme extend to flow models?

Reframe equation 10 as $\gamma_t^2 \;=\; \sigma_t^2 \;-\; \big(a_t\,\lambda_t\,\sigma_y\big)^2$ leads to the following condition for γ_t:

$$\gamma_t \;=\; \begin{cases} \sigma_t^2 \;-\; \big(a_t\,\sigma_y\big)^2, & \sigma_t \geq a_t\,\sigma_y, \\ 0, & \sigma_t < a_t\,\sigma_y\;. \end{cases} \tag{12}$$

Since flow models do not have a noise schedule σ_t, it is equivalent to setting $\sigma_t = 0$. This would imply the second case $(\sigma_t < a_t\,\sigma_y)$ of equations 11 and 12, leading to $\gamma_t = 0$ and

$$\lambda_t = \frac{\sigma_t}{a_t\,\sigma_y} = 0. \tag{13}$$

Thus, in a flow model with no noise schedule $\sigma_t \equiv 0$ the noise-aware formulation collapses to $\gamma_t = 0$ and $\lambda_t = 0$—i.e., no fidelity correction. Moreover, the core principle of "preserving the variance in each timestep t" does not apply, since a flow model does not have an inherent variance schedule through σ_t. This shows that the diffusion-style noise-robust update does not transfer directly to flows. Empirically, we find that fidelity updates in flow models are sensitive to measurement noise and, if used at all, must be applied sparingly with careful, task-dependent tuning.

B. More Details on the Flow Model

FlowSteer, and all related baselines were implemented on an NVIDIA A100 GPU with VRAM 80GB. Resources were only required for model inference.

B.1. Feature sharing

The Flux-dev [2] model is our base-line image editing model, that we adapt for image restoration. Both the inversion path and the reconstruction path has $N = 30$ steps. The velocity prediction $v_\theta(.)$ in each step is modeled through a Diffusion Transformer (DiT) [6] block. The diffusion transformer has "double-block" layers and "single block" layers, out of which the "single block" layers are used for feature sharing. The attention maps that are calculated in the last ζ inversion steps (input to noise) and correspond to the first ζ reconstruction steps (noise to image). During reconstruction, the cached attention maps are used for the first ζ steps, creating an implicit conditioning that the reconstructed image should preserve some qualities of the original image. There are similar approaches in literature of caching attention maps or just the Values or Key-Query pairs to drive an edited image/video to be faithful to an input image. [3, 4, 7–9, 11, 12, 16, 19, 21]. In our design, we cache the complete attention map. However, after experimenting with different feature sharing schemes and grid searches through hyperparameters as described in section 3, we find that such implicit conditioning is insufficient for enforcing the pixel-level fidelity required for image restoration tasks.

B.2. Text prompts

As described in section 3, there are two types of prompts that implicitly condition the flow field. During inversion, the source prompt C_1 is used to describe the degraded image. During reconstruction, the target prompt C_2 is used to describe the target image features. The prompts are manually selected by the user. The table 4 shows the sample prompts used for each of the restoration tasks.

C. Details on the Degradation Models

Colorization. The forward operator $\mathbf{A}$ maps RGB to an achromatic image by averaging channels and repeating the result across three channels. For $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ and pixel p,

$$(\mathbf{A}\mathbf{x})_c(p) = \tfrac{1}{3}\sum_{k=1}^{3} x_k(p), \qquad c \in \{1,2,3\}.$$

The Moore–Penrose pseudoinverse coincides with $\mathbf{A}$, i.e.

$$\mathbf{A}^\dagger \mathbf{y} = \mathcal{R}\left(\tfrac{1}{3}\sum_{k=1}^{3} y_k\right),$$

where $\mathcal{R}(\cdot)$ replicates a single channel three times. Hence, for any $\mathbf{y} \in \mathrm{range}(\mathbf{A})$ (three identical channels), $\mathbf{A}^\dagger\mathbf{y} = \mathbf{y}$ and $\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}$.

Deblurring. The forward operator is a circular Gaussian blur $\mathbf{A}$ (convolution with kernel h). We use the Tikhonov-regularized (Wiener-type) pseudoinverse parameterized by λ_{W},

$$\mathbf{A}^\dagger_{\lambda_{\mathrm{W}}} \triangleq (\mathbf{A}^*\mathbf{A} + \lambda_{\mathrm{W}}\mathbf{I})^{-1}\mathbf{A}^*,$$

where $\mathbf{A}^*$ denotes the adjoint of $\mathbf{A}$. We set $\lambda_{\mathrm{W}} = 0.1$ in all our experiments. The forward model can be implemented using Fourier transforms as $\mathcal{F}\{\mathbf{y}\} = \mathcal{F}\{h\}\,\mathcal{F}\{\mathbf{x}\}$, (or equivalently, as the convolution $\mathbf{y} = h * \mathbf{x}$). The inversion is implemented in python code in the following form,

$$\mathbf{A}^\dagger_{\lambda_{\mathrm{W}}}\mathbf{y} = \mathcal{F}^{-1}\left[\frac{\overline{\mathcal{F}\{h\}}}{|\mathcal{F}\{h\}|^2 + \lambda_{\mathrm{W}}}\,\mathcal{F}\{\mathbf{y}\}\right].$$

$\overline{\mathcal{F}\{h\}}$ is the element-wise complex conjugate of $\mathcal{F}\{h\}$. For a real, symmetric Gaussian kernel h, $\overline{\mathcal{F}\{h\}} = \mathcal{F}\{h\}$.

Super-resolution ($\times 4$). The forward operator $\mathbf{A}$ downsamples an RGB image by average–pooling over non-overlapping 4×4 blocks (per channel) and decimating by a factor of 4 along height and width: $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^{3\times H/4\times W/4}$ with

$$y_c(u,v) = \tfrac{1}{16}\sum_{i,j=0}^{3} x_c(4u+i,\, 4v+j).$$

As a practical pseudo-inverse we use a right-inverse that restores the original spatial size by *patch replication* (nearest-neighbor up-sampling):

$$(\mathbf{A}^\dagger\mathbf{y})_c(i,j) = \mathbf{y}_c\big(\lfloor i/4\rfloor,\, \lfloor j/4\rfloor\big),$$

such that $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}$ on $\mathbb{R}^{3\times H/4\times W/4}$.

Denoising. We model denoising with the identity forward operator $\mathbf{A} = \mathbf{I}$, so the measurement is simply $\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta} = \mathbf{x} + \boldsymbol{\eta}$. Because $\mathbf{I}$ is self-adjoint and full-rank, its Moore–Penrose pseudoinverse is itself: $\mathbf{A}^\dagger = \mathbf{I}$.

D. Design Details for $\{\lambda_i\}$

D.1. A Two-Step Schedule

In section 5.3 we describe the effect of the schedule $\{\lambda_i\}$. Empirically we observe that a single step-design as mentioned in table 2 (left) is sufficient for general reconstructions. However, the reconstruction quality can be improved by having a scheduler that gradually reduces the strength of the fidelity update. This empirical observation is shown in table 2 (right).

The following is the python-style implementation of a two-step scheduler, with the parameters $i_{\text{start}}, i_{\text{step}}, i_{\text{end}}, h_1, h_2$. Between $i_{\text{start}}, i_{\text{step}}$ the value of λ_i is h_1, and between $i_{\text{step}}, i_{\text{end}}$ the value of λ_i is h_2.

Listing 1. Step-shaped λ_i schedule used in SteerFlow.

```python
def make_lambda_step_schedule(
    timesteps,
    *, start, step, end,
    h_1=1.0, h_2=0.5,
    final_pad=1,
):
    N = len(timesteps) - 1
    lam = np.zeros(N, dtype=np.float32)
    if N <= 0:
        return lam

    i_0 = to_index(start, N)
    i_1 = to_index(step,  N)
    # make end exclusive
    i_2 = to_index(end,   N) + 1

    # order & clamp
    i_start, i_step = min(i_0, i1), max(i0, i1)
    i_step, i_end = min(i1, i2), max(i1, i2)
    i_start = np.clip(i_start, 0, N)
    i_step = np.clip(i_step, 0, N)
    i_end = np.clip(i_end, 0, N)

    if i_start < i_step:
        lam[i_start:i_step] = h_1
    if i_step < i_end:
        lam[i_step:i_end] = h_2

    if 0 < final_pad < N:
        lam[-final_pad:] = 0.0

    # keep a peak=1 if padding nuked it
    if lam.max() <= 0 and N - final_pad - 1 >=
        0:
        lam[N - final_pad - 1] = 1.0
    else:
        lam /= max(1.0, float(lam.max()))
    return lam
```

D.2. Effect of the Fidelity update

The effect of the fidelity update is visually shown in the figures 10 and 11. If the update is started too late (i_{start} is too late), then the reconstruction will have artifacts from the hallucinated details. (figure 10). If the update is started too early(i_{start} is too early), then the desired level of hallucinations (such as rich colors for colorization) have not yet formed. This makes the reconstruction have over smoothness, and loose color and texture details. (figure 11).

D.3. Final padding

A final padding parameter is added to enforce that at least a few steps of the reconstruction path does not end with a fidelity update. This is another empirical design decision, where we observe that having some steps without fidelity updates helps to smoothen out some of the noise artifacts. However, if the final padding is too high, the image can get oversmoothed. This is effectively the same as having a fidelity update too early on in the reconstruction path, as shown in figure 11.

E. Limitations and Future Directions

E.1. Empirical schedule

The Flow Steer schedule recommendations are given through empirical fine-tuning. For the four restoration tasks together, we recommend a one-step schedule in Table 2(left). For each task separately, we recommend a two-step schedule in Table 2(right). Even with this task-specific schedule recommendation, there are cases where the schedule does not work for every image in the dataset. This can be seen in some of the failure cases for the current SteerFlow scheme as shown in Figure 12.

A future directions would be to extend this schedule to be adaptable for each individual image, so that manual tuning can be avoided. The key is to estimate the noise injected by the fidelity update and trigger the update at the step whose model noise budget best matches it, while adjusting the update strength $\{\lambda_t\}$ accordingly. This would lead to an image-adaptive scheduler, and will not have to depend on heuristics.

E.2. Artifacts from explicit conditioning

The explicit conditioning of FlowSteer relies on Pseudoinverse operators. This has the inherent limitation of introducing artifacts, such as shown in figure 13. Apart from the noisy artifacts described above, there are instances of blocking-artifacts from the upsampling operation in super-resolution, and ringing-artifacts from the Weiner filter in deblurring.

F. More Visual Results

F.1. More visual results on restoration tasks

More images on the restoration tasks, highlighting that we preserve both pixel-level fidelity and rich perceptual quality. Zoom in and highlight differences.

F.2. Plug-and-Play(PnP) on pre-trained Flow models

We recreate the results of PnP-Flow [13] to verify if a Plug-and-play type algorithm can achieve visually appealing images after image restoration. It was observed that it performs well only when the underlying model is the provided model from the authors, which is specifically trained for the class of images being tested. For example, with the pre-trained flow model (which has been trained on a cat dataset) it gives plausible results on cat images. However, when flux-dev [2] (which is trained on a much broader class of data) is used as the underlying flow model, the Plug-and-play method fails to converge to a plausible reconstruction. A grid search was run to select the hyper-parameters of the PnP algorithm and the values: $(\alpha, \gamma, \eta_{dn}) = (0.3, 0.8, 0.3)$ were selected. This is demonstrated in Figure 14.

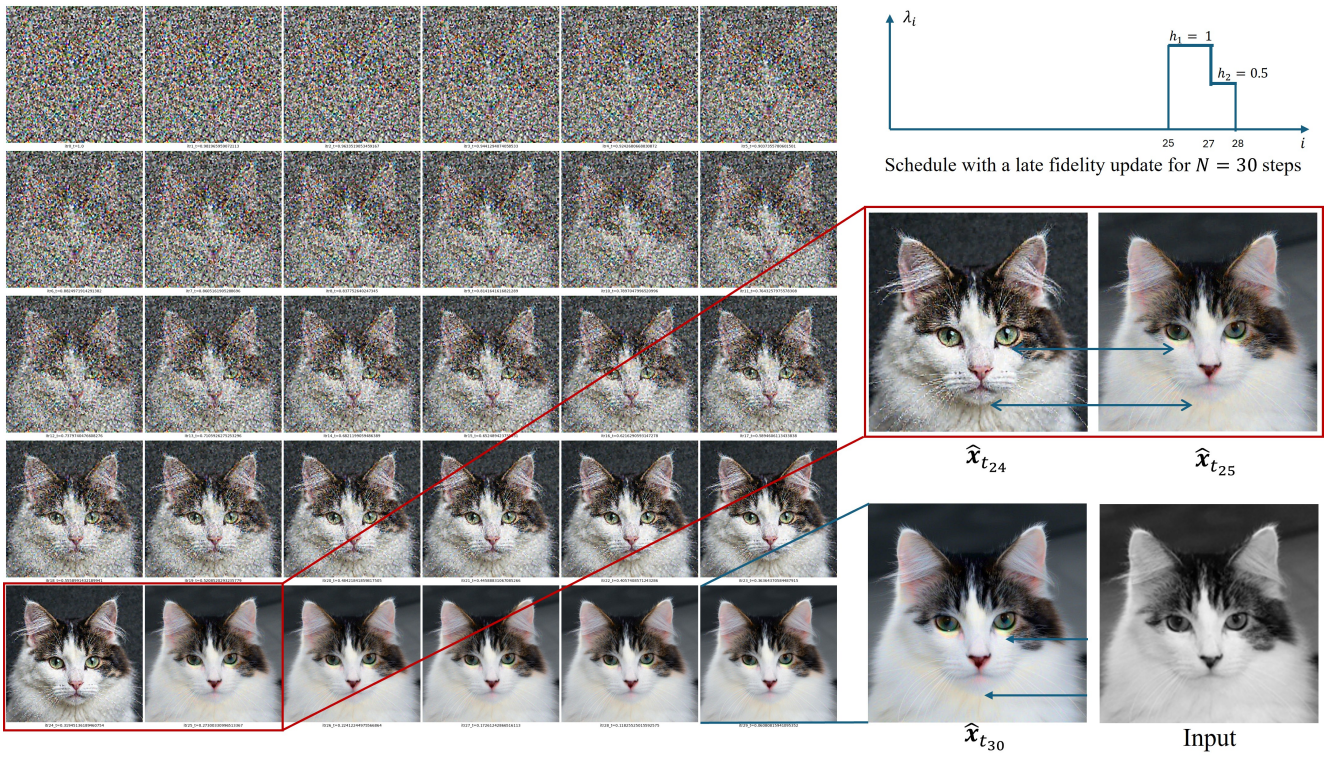

Figure 10. Visualising the reconstruction path with the fidelity update being late. Before the fidelity update has been appled ($\widehat{\mathbf{x}}_{t_{24}}$), the flux model has already hallucinated fine textures and colors. Immediately after activation ($\widehat{\mathbf{x}}_{t_{25}}$), the fidelity update steers the path toward the measurement. However, some residual artifacts persist in the final reconstruction.

G. More Quantitative Results

G.1. Inversion-free models

During the review and rebuttal process, we were encouraged to compare FlowSteer against FlowChef [14] and FLAIR [5]. Both methods target inversion-free reconstruction and, like FlowSteer, are zero-shot approaches that leverage pre-trained diffusion or flow models to estimate the underlying velocity field. We reproduced FlowChef [14] using the publicly available Flux1-dev [2] implementation. Since the released version of FLAIR [5] uses Stable Diffusion 3 (SD3) [18] as its backbone, we additionally constructed an analogous version based on Flux1-dev [2] to enable a more consistent comparison across methods.

Table 5 reports these additional results alongside the baseline comparisons from Table 1 of the main paper. We observe that both methods perform relatively poorly on image colorization, while remaining competitive with PnP-Flow [13] on the other reconstruction tasks. In contrast, FlowSteer consistently achieves a stronger trade-off between pixel-level fidelity and perceptual quality. That said, we acknowledge that FLAIR [5] is likely to be more effective under severe degradations, such as 8× or 12× super-resolution and motion deblurring. This reflects a limitation of FlowSteer: under heavy degradation, the pseudo-inverse operator can introduce undesirable artifacts that degrade reconstruction quality.

Overall, our experiments suggest that FlowSteer handles a broad range of inverse problems more effectively, with particularly strong gains in image colorization.

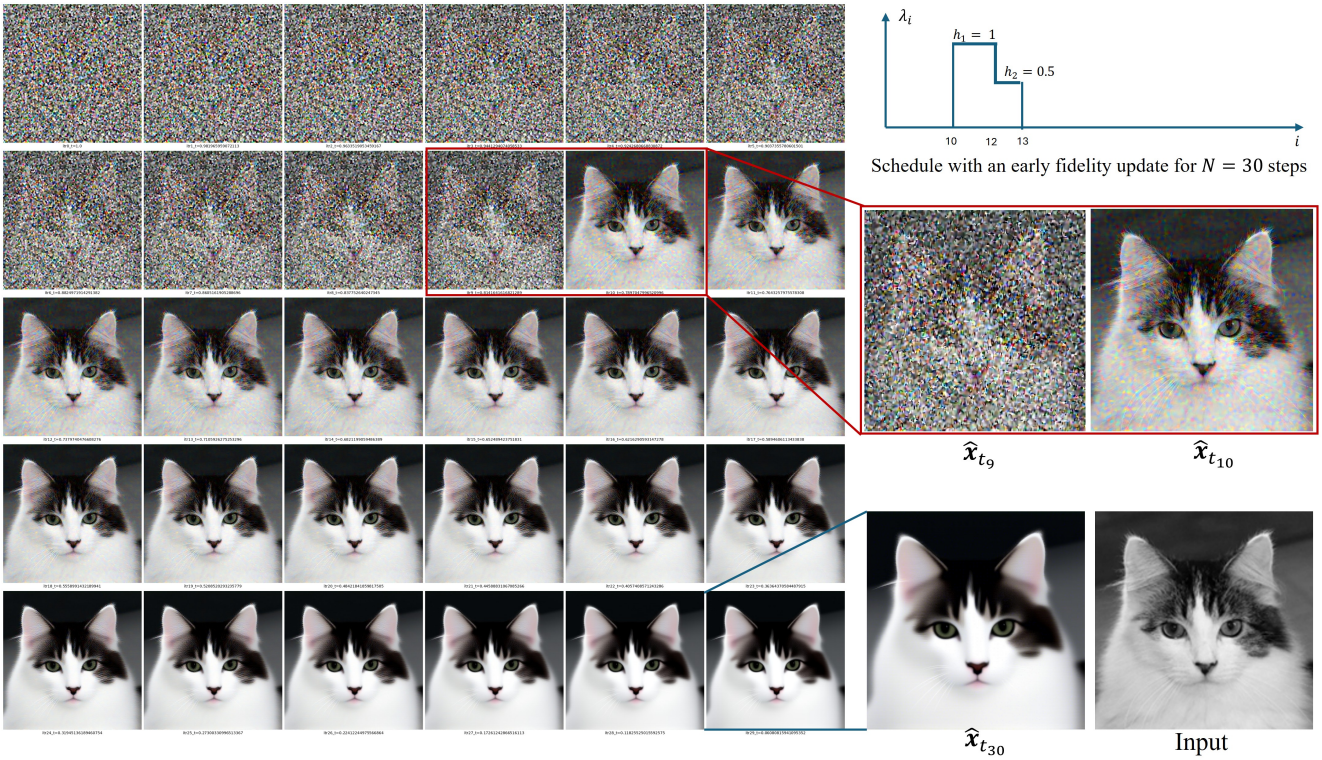

Figure 11. Visualising the reconstruction path with the fidelity update being early. Before the fidelity update has been appled ($\hat{\mathbf{x}}_{t_9}$), the flux model has not completely formed the color pallette and textures. The fidelity update conditions the path to have less color($\hat{\mathbf{x}}_{t_{10}}$), and the large number of steps without the fidelity update over smoothens the result. The same effect will be seen when a higher number of flux steps are padded at the end of the schedule, as described in Section D.3.

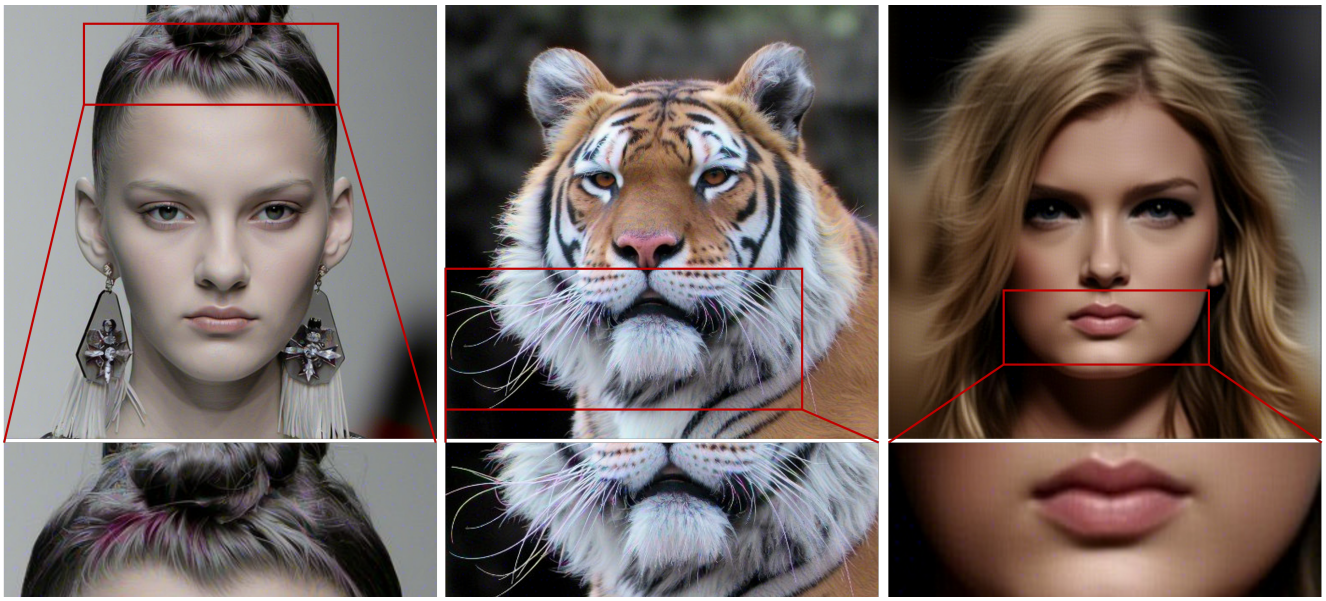

Figure 12. Noise artifacts are seen in some of the final reconstructed image with the two-step scheduler. The restoration tasks corresponding to these images are colorization (left and center) and deblurring (right). Fine tuning the scheduler on a per-image basis may reduce these artifacts as discussed in Section E. Zoom in to clearly see the noise artifacts in some pixels.

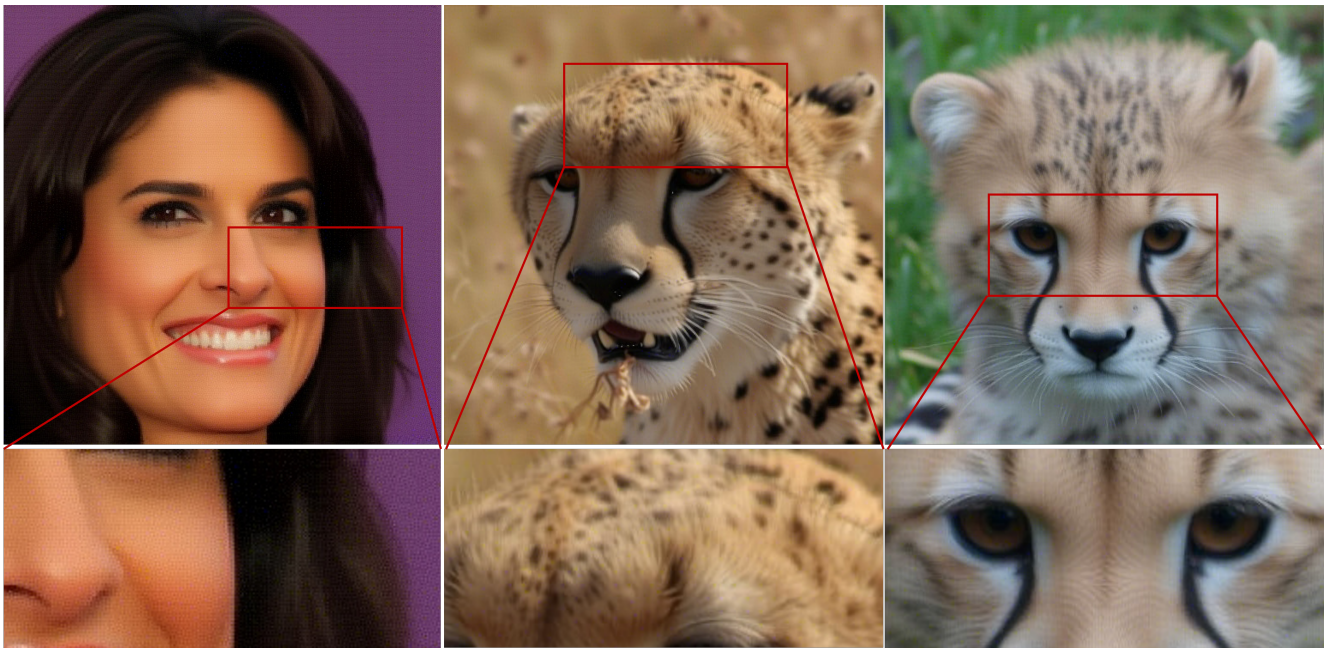

Figure 13. Artifacts resulting from Pseudo-inverse operators in FlowSteer as discussed in Section E. Blocking artifacts affect edges in super-resolution(left). The Weiner filter creates ringing effects in deblurring(center and right). Zoom in to clearly see the artifacts.

Type	#	Source prompt C_1	Target prompt C_2
Colorization			
Pets	1	A black and white image of a cat. The cat is white with black patches. The background is dark. The nose of the cat is pink. The eyes of the cat are green.	A colorful image of a cat. The cat is white with black patches. The background is dark. The nose of the cat is pink. The eyes of the cat are green.
Wild	2	A black and white image of a cheetah. The fur of the leopard is bright, with dark spots. The leopard has dark streaks running down from its dark eyes.	A colored image of a cheetah. The fur of the leopard is golden yellow, with dark spots. The leopard has dark streaks running down from its dark eyes.
Wild	3	A black and white image of a fox. The fox has brown fur with white streaks. The nose of the fox is black. The eyes of the fox are dark brown.	A colorful image of a fox. The fox has brown fur with white streaks. The nose of the fox is black. The eyes of the fox are dark brown.
Humans	4	A black and white image of a man.	A colorful image of a man. The man has black eyes.
Deblurring			
Pets	1	A blurred image of a cat. The cat has white fur with black spots.	A sharp image of a cat. The cat has white fur with black spots. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Wild	2	A blurred image of a lion. The lion has golden colored fur and dark brown eyes.	A sharp image of a lion. The lion has golden colored fur and dark brown eyes. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Wild	3	A blurred image of a leopard. The leopard has brown fur and black patches.	A sharp image of a leopard. The leopard has brown fur and black patches. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Humans	4	A blurred image of a man.	A sharp image of a man. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Super-resolution			
Pets	1	A low resolution image of a dog. The dog is brown with white patches.	A sharp, high resolution image of a dog. The dog is brown with white patches. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Wild	2	A low resolution image of a lion. The lion has golden colored fur and dark brown eyes.	A sharp, high resolution image of a lion. The lion has golden colored fur and dark brown eyes. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Humans	3	A low resolution image of a woman.	A sharp, high resolution image of a woman. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail.
Denoising			
Pets	1	A noisy image of a dog. The dog is brown with white patches.	A clean, noise free image of a dog. The dog is brown with white patches. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail. There are no RGB noise artifacts.
Wild	2	A noisy image of a cheetah. The fur of the cheetah is bright, with dark spots. The cheetah has dark streaks running down from its dark eyes.	A clean, noise free image of a cheetah. The fur of the cheetah is bright, with dark spots. The cheetah has dark streaks running down from its dark eyes. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail. There are no RGB noise artifacts.
Human	3	A noisy image of a man.	A clean, noise free image of a man. Highly detailed, taken using a Canon EOS R camera, hyper detailed photo-realistic maximum detail. There are no RGB noise artifacts.

Table 4. Sample prompt pairs per task. "Source" describes the input (e.g., grayscale, low-res, noisy, blurred); "Target" describes the intended restored image that is used to implicitly steer the flux model.

Figure 14. Plug and play approach on different types of flow models. Although it produces plausible results on a flow model trained on cats(left), it does poorly when implemented on a flow model trained on other general data. This is described in Section F.

Methods	Colorization				Super resolution				Deblurring				Denoising			
	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	CLIP↑
Reference	27.1891	0.8333	0.2334	0.5807	26.7337	0.7829	0.2381	0.4380	28.2371	0.7586	0.2821	0.3095	22.7706	0.4399	0.4427	0.3178
D-Flow [1]	18.5295	0.5554	0.4977	0.2724	23.4039	0.6236	0.4132	0.2984	22.2396	0.6316	0.4104	0.2798	19.0651	0.5232	0.4840	0.2451
OT-ODE [15]	N/A				29.1072	0.8296	0.1994	0.6033	31.0467	0.8612	0.1984	0.5908	28.8276	0.8123	0.2505	0.4416
Flow-Priors [22]	21.7889	0.5716	0.4639	0.4808	27.6858	0.7151	0.3036	0.4724	30.4326	0.8350	0.2264	0.5535	28.8047	0.7645	0.2770	0.4479
FlowChef [14]	18.5834	0.5537	0.4640	0.1857	31.3876	0.8430	0.1849	0.5290	32.8454	0.8929	0.2093	0.3065	29.6792	0.8803	0.3421	0.2711
PnP-flow [13]	27.1620	0.8640	0.2830	0.3482	31.2073	0.8753	0.1755	0.3938	32.7392	0.8840	0.1728	0.4597	30.5899	0.8733	0.2207	0.5103
RFEdit [19]	20.3255	0.6602	0.3648	0.8703	22.3243	0.7362	0.2822	0.7863	22.9813	0.7431	0.2750	0.7914	17.0219	0.4438	0.4545	0.8570
FLAIR [5]	26.7679	0.8396	0.2934	0.6328	31.0051	0.8833	0.1754	0.6072	30.9323	0.8940	0.1652	0.6990	30.3760	0.8873	0.2601	0.6089
Ours	27.4214	0.8696	0.2081	0.7734	32.8552	0.9022	0.1700	0.6714	32.8749	0.9052	0.1486	0.8177	32.2125	0.8924	0.1822	0.7679

Table 5. Extended quantitative comparison with flow-based restoration methods. *Reference* uses the degraded image y. FlowSteer has high perceptual quality, and high pixel-level fidelity. We highlight the best and second-best per metric.

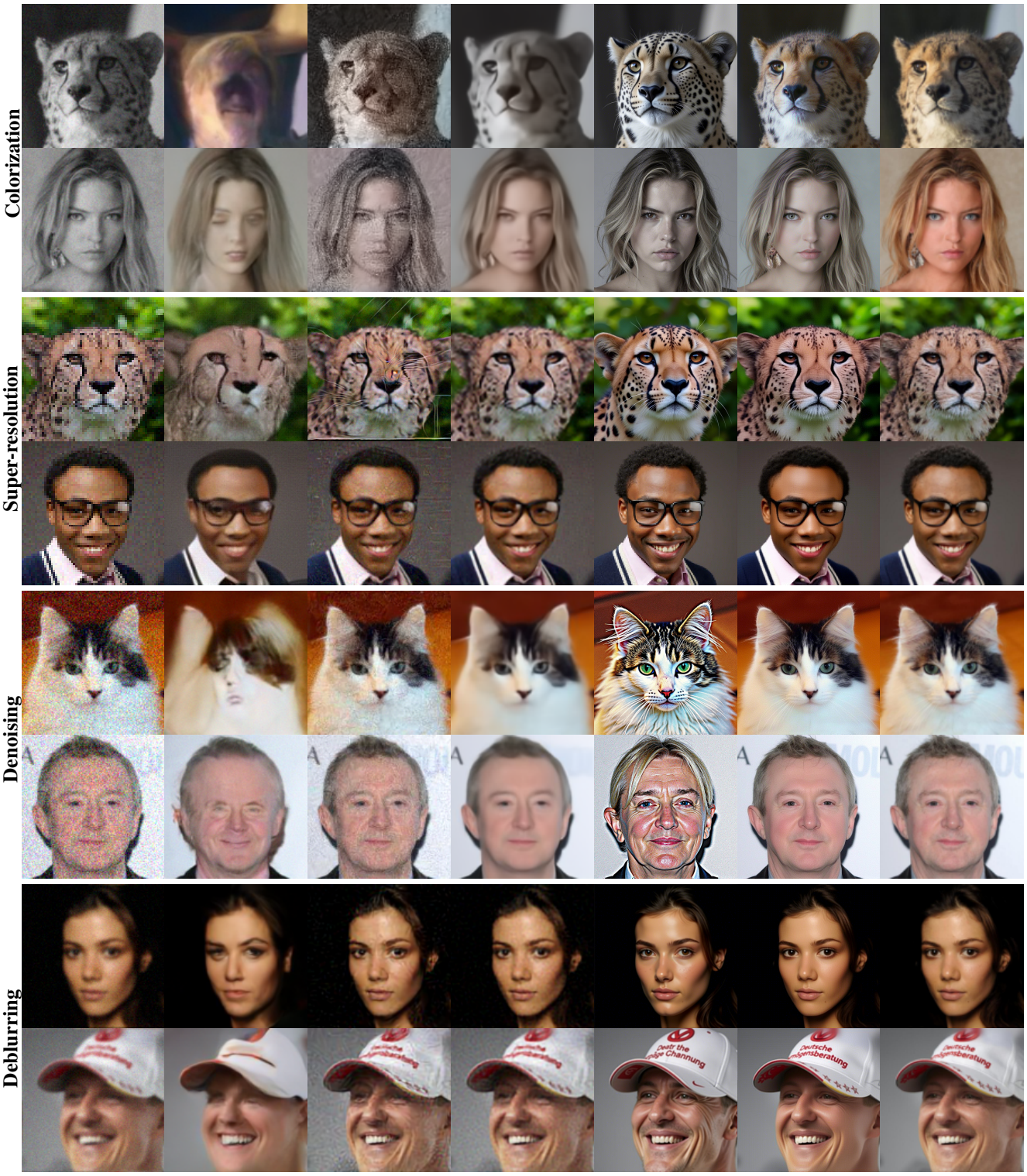

Figure 15. More qualitative comparisons of flow-based methods against FlowSteer. The restoration models in columns 2-4 have undesirable artifacts such as excessive blur. The image editing baseline in column 5 has poor fidelity. FlowSteer achieves better pixel-level fidelity, while generating viusally appealing details. Zoom in for better comparisons.

References

[1] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024. 9, 10

[2] Black Forest Labs. FLUX.1: Frontier visual intelligence. Technical report, Black Forest Labs, 2024. Technical announcement and model release. 3, 4, 5

[3] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22560–22570, 2023. 3

[4] Duygu Ceylan, Chun-Hao P Huang, and Niloy J Mitra. Pix2video: Video editing using image diffusion. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23206–23217, 2023. 3

[5] Julius Erbach, Dominik Narnhofer, Andreas Dombos, Bernt Schiele, Jan Eric Lenssen, and Konrad Schindler. Solving inverse problems with flair. 2025. 5, 9

[6] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024. 3

[7] Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arXiv:2307.10373*, 2023. 3

[8] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[9] Amir Hertz, Andrey Voynov, Shlomi Fruchter, and Daniel Cohen-Or. Style aligned image generation via shared attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4775–4785, 2024. 3

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 1

[11] Jimyeong Kim, Jungwon Park, Yeji Song, Nojun Kwak, and Wonjong Rhee. Reflex: Text-guided editing of real images in rectified flow via mid-step feature extraction and attention adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 3

[12] Shaoteng Liu, Yuechen Zhang, Wenbo Li, Zhe Lin, and Jiaya Jia. Video-p2p: Video editing with cross-attention control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8599–8608, 2024. 3

[13] Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. Pnp-flow: Plug-and-play image restoration with flow matching. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. 4, 5, 9, 10

[14] Maitreya Patel, Song Wen, Dimitris N. Metaxas, and Yezhou Yang. Steering rectified flow models in the vector field for controlled image generation. *arXiv preprint arXiv:2412.00100*, 2024. 5, 9

[15] Ashwini Pokle, Matthew J. Muckley, Ricky T. Q. Chen, and Brian Karrer. Training-free linear image inverses via flows. *Transactions on Machine Learning Research (TMLR)*, 2024. Often referred to as OT-ODE. 9

[16] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15932–15942, 2023. 3

[17] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020. 1

[18] Stability AI. Stable Diffusion 3: Multimodal Diffusion Transformer release. https://stability.ai/news/stable-diffusion-3-research-paper, 2024. Technical announcement and model weights release. 5

[19] Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024. 3, 9, 10

[20] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. 1

[21] Yuechen Zhang, Jinbo Xing, Eric Lo, and Jiaya Jia. Real-world image variation by aligning diffusion inversion chain. *Advances in Neural Information Processing Systems*, 36: 30641–30661, 2023. 3

[22] Yasi Zhang, Peiyu Yu, Yaxuan Zhu, Yingshan Chang, Feng Gao, Ying Nian Wu, and Oscar Leong. Flow priors for linear inverse problems via iterative corrupted trajectory matching. *Advances in Neural Information Processing Systems*, 37:57389–57417, 2024. 9, 10