FLOW-FILTER: PLUG-AND-PLAY IMAGE SMOOTHING WITH FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Structure-aware image smoothing seeks to remove textures while preserving structures, but balancing texture suppression and structural fidelity remains challenging. Handcrafted priors offer explicit local geometric constraints yet risk over-smoothing, while pretrained generative priors capture rich natural-image statistics but may retain or reconstruct unwanted textures. To address this challenge, we propose **Flow-Filter**, a plug-and-play image smoothing framework that uses a pretrained Flow Matching model as a generative structural prior for structure–texture decomposition and requires no task-specific training or fine-tuning. We further introduce a joint gradient structural prior that couples first- and second-order differential responses, allowing global natural-image modeling from the generative prior to complement explicit local geometric constraints. To adapt FM to selective texture removal, we couple its denoising process with progressive ADMM-based decomposition and introduce per-step clean–noise endpoint modulation to reduce structural drift and persistent texture residuals. Extensive experiments demonstrate that Flow-Filter suppresses complex textures more effectively while preserving salient structures and fine geometric details, achieving a favorable balance between smoothing strength and structural fidelity. Our results show that pretrained Flow Matching models can go beyond their conventional role in generation and serve as effective plug-and-play priors for selective image filtering. Code is available at `https://anonymous.4open.science/r/FM-Filter-CB8D/`.



Figure 1: Representative smoothing results of Flow-Filter.

1 INTRODUCTION

Image smoothing is a selective removal problem, i.e., it aims to suppress fine-scale variations while preserving salient structures. The challenge is that high-frequency components may correspond

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

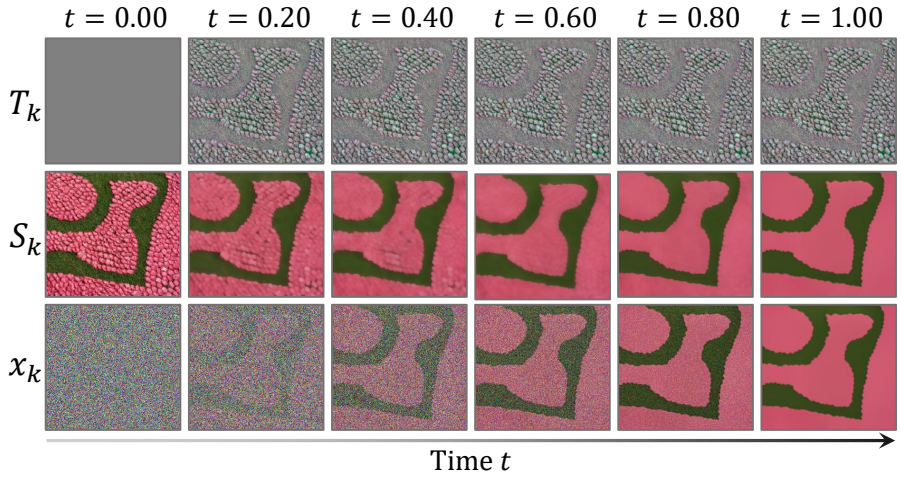


Figure 2: Evolution of x_k, S_k, and T_k across flow levels.

either to textures that should be removed or to meaningful structural details that must be preserved. The central difficulty is therefore not simply how much to smooth, but what should be removed. Existing methods mainly rely on handcrafted priors (Liu et al., 2022; Yang et al., 2026; Qi et al., 2025) or learned image mappings (Feng et al., 2022; Li et al., 2022; Zheng et al., 2026). Handcrafted methods provide explicit control over the smoothing process, but their decisions are often limited by local image statistics. Learning-based methods exploit richer image information, yet typically require task-specific training and may not generalize reliably across different texture patterns.

Pretrained generative models learn rich natural-image statistics from large-scale data and can provide powerful image priors without smoothing-specific retraining (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023). Such priors have shown strong performance in inverse restoration problems (Kim et al., 2025). Image smoothing, however, places a different demand on the generative prior. Restoration starts from a degraded observation and seeks to recover plausible natural content, whereas smoothing starts from an already natural image and deliberately removes part of its plausible content. A generative prior may therefore preserve, or even reconstruct, textures that should instead be suppressed. This suggests that the generative prior should focus on structural modeling rather than direct smoothing. It can provide globally plausible structural proposals, while an explicit prior determines which local variations should be removed.

Based on this observation, we propose **Flow-Filter**, a plug-and-play image smoothing framework that exploits pretrained Flow Matching priors for structure–texture decomposition. We formulate **a smoothing model that combines generative structural modeling with explicit mixed-order gradient regularization**, allowing global structural information and local texture suppression to play complementary roles. We establish a **Flow–ADMM denoising-level correspondence** to coordinate the generative prior with the evolving optimization scale, and introduce **per-step clean–noise endpoint modulation** to keep the flow state coupled to the current decomposition, where clean-side anchoring limits structural drift and noise-side refresh suppresses persistent high-frequency patterns across flow levels. We further observe that repeated inference on a fixed patch grid can accumulate phase-locked prediction errors, and develop **two-dimensional cyclic patch-phase shifting** to alleviate these artifacts without retraining or additional network evaluations. Experiments on texture filtering, JPEG artifact removal, and detail enhancement demonstrate that Flow-Filter effectively suppresses textures while preserving fine structures. Representative smoothing results are shown in Fig. 1.

2 Related Work

2.1 Global Filtering via Numerical Optimization

Global filtering methods formulate image smoothing as optimization problems with handcrafted priors, differing mainly in their regularization. Representative approaches include weighted ℓ_2 gradient regularization in WLS Farbman et al. (2008), sparse ℓ_0 and ℓ_1 penalties Xu et al. (2011); Guo et al. (2017); Yang et al. (2023), and more flexible nonconvex regularizers such as Charbonnier, truncated

Huber, and proximal p-norm penalties Liu et al. (2020b; 2022); Yang et al. (2026). These methods offer explicit and interpretable control over smoothing, but rely largely on predefined local gradient statistics. Since structures and textures can exhibit similar local responses, a handcrafted prior alone may struggle to suppress diverse textures while preserving complex structures.

2.2 Learning-Based Filtering and Generative Priors

Early methods learn edge-aware filtering or approximate traditional filters through supervised training Xu et al. (2015); Chen et al. (2017); Wu et al. (2018), while subsequent work explores unsupervised learning Fan et al. (2018), synthetic or RGBD supervision Feng et al. (2022); Li et al. (2022); Zhong et al. (2024), and parameterized networks with adjustable smoothing strength Fan et al. (2021); Yang et al. (2024). GTF (Zheng et al., 2026) uses supervised and reinforcement fine-tuning for texture filtering. PnP-Flow (Martin et al., 2025) combines data-fidelity updates, flow-path reprojection, and FM denoising for inverse reconstruction. Unlike GTF, Flow-Filter keeps the pre-trained model frozen; unlike PnP-Flow, it performs explicit structure–texture decomposition with a joint gradient prior and per-step endpoint modulation.

3 Background

3.1 Flow Matching and JiT Endpoint Prediction

Flow Matching (FM) learns a time-dependent transport from a noise distribution to the data distribution along a prescribed probability path (Lipman et al., 2023). Let $x_0 \sim \mathcal{N}(0, I)$ and $x_1 \sim p_{\text{data}}$ denote the noise and clean-data endpoints, respectively. Under our noise-to-clean convention, $t \in [0, 1]$ and we use the linear path

$$x_t = (1 - t)x_0 + tx_1, \tag{1}$$

whose target velocity is $x_1 - x_0$. A learned vector field $v_\theta(x_t, t)$ approximates this transport.

JiT (Just image Transformers) is a pixel-space generative model that directly predicts the clean endpoint (Li & He, 2026):

$$\widehat{x}_1 = D_\theta(x_t, t). \tag{2}$$

Under the linear path in Eq. (1), this prediction induces the equivalent FM velocity

$$v_\theta(x_t, t) = \frac{\widehat{x}_1 - x_t}{1 - t} = \frac{D_\theta(x_t, t) - x_t}{1 - t}, \qquad t < 1. \tag{3}$$

Thus, clean-endpoint and velocity predictions are interchangeable for a given (x_t, t). In Flow-Filter, the pretrained JiT model serves as the pixel-space generative prior, and we directly use its clean-endpoint prediction in the subsequent formulation.

4 Methodology

Given an input image f, Flow-Filter decomposes it into a structure layer S and a texture layer T by coupling a pretrained Flow Matching prior with explicit gradient regularization under an ADMM framework. JiT provides structure proposals that are progressively refined throughout the decomposition, while endpoint modulation and cyclic patch-phase shifting stabilize the generative updates. As shown in Fig. 2, T_k, S_k, and x_k progressively evolve toward texture separation, structure smoothing, and denoising, respectively.

4.1 Image Smoothing Model with Generative and Gradient Priors

Given $f = S + T$, we combine an implicit FM generative prior, an explicit mixed-order gradient prior, and an RGB-coupled texture prior:

$$\min_{S,T} \lambda\Phi_\theta(S) + R_{\mu,\gamma}(\nabla S) + \|T\|_{2,1}, \qquad \text{s.t. } f = S + T. \tag{4}$$

Generative structural prior. $\Phi_\theta(S)$ denotes the implicit prior induced by the pretrained FM model, providing global natural-image statistics and structural constraints. We realize it through time-conditioned clean-endpoint prediction as a plug-and-play structure operator. The nominal weight $\lambda > 0$ is absorbed into scale calibration (Appendix A.1).

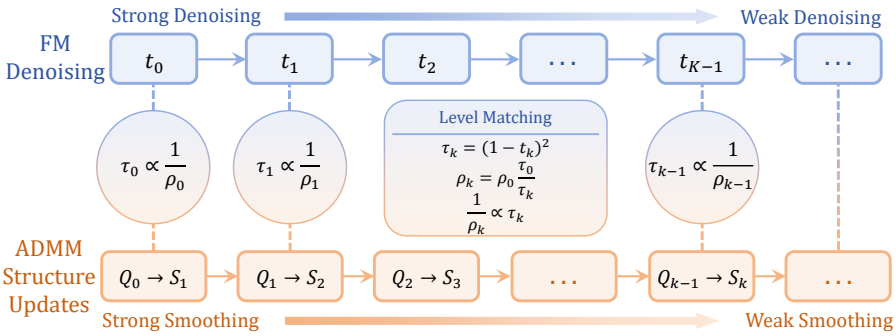


Figure 3: Flow–ADMM denoising-level alignment.

Gradient structural prior. To provide explicit local geometric constraints, we define a mixed-order gradient prior that jointly models first- and second-order differential responses:

$$R_{\mu,\gamma}(\nabla S) = \sum_{d,p} \phi_{\mu,\gamma}\left(\|J_d S(p)\|_2\right), \qquad J_d S(p) = \begin{bmatrix} \nabla_d S(p) \\ \xi \nabla_d^2 S(p) \end{bmatrix}. \tag{5}$$

Here, d indexes the horizontal, vertical, and two diagonal directions, $\xi > 0$ balances the first- and second-order responses, and $\phi_{\mu,\gamma}$ denotes the minimax concave penalty (MCP). The iteration-dependent regularization strength and structure refinement are described in Sec. 4.2.2.

Texture regularizer. We use the RGB-coupled group-sparsity prior

$$\|T\|_{2,1} = \sum_p \|T(p)\|_2, \tag{6}$$

which enforces a shared texture support across RGB channels and collects fine-scale variations and residual oscillations in T. The constraint $f = S + T$ maintains decomposition consistency.

4.2 ADMM-Based Model Optimization

We use an ADMM backbone to enforce the decomposition constraint in Eq. (4). Introducing the unscaled dual variable b and the penalty parameter ρ_k, the corresponding augmented Lagrangian for the fixed-prior formulation is

$$\mathcal{L}_{\rho_k}(S,T,b) = \lambda\Phi_\theta(S) + R_{\mu,\gamma}(\nabla S) + \|T\|_{2,1} + \langle b, S+T-f\rangle + \frac{\rho_k}{2}\|S+T-f\|_2^2. \tag{7}$$

We initialize $S_0 = f$, $T_0 = 0$, and $b_0 = 0$. For K structure updates, let

$$0 \le t_0 < t_1 < \cdots < t_{K-1} < t_K = 1, \qquad \sigma_k = 1 - t_k. \tag{8}$$

To match the FM denoising level t_k with the ADMM regularization scale ρ_k, we set

$$\tau_k = (\sigma_k)^2 = (1-t_k)^2, \qquad \rho_k = \rho_0 \frac{\tau_0}{\tau_k} = \rho_0 \left(\frac{1-t_0}{1-t_k}\right)^2, \quad k = 0, \ldots, K-1. \tag{9}$$

Here, $t_K = 1$ is only the terminal clean endpoint and is not used for an additional structure update. The derivation of this Flow–ADMM denoising-level correspondence is provided in Appendix A.1. Fig. 3 illustrates the correspondence between the FM perturbation scale τ_k and the effective ADMM regularization scale $1/\rho_k$.

4.2.1 Texture Update

With S_k and b_k fixed, the texture subproblem is

$$T_{k+1} = \arg\min_T \|T\|_{2,1} + \frac{\rho_k}{2}\left\|T - \left(f - S_k - \frac{b_k}{\rho_k}\right)\right\|_2^2, \tag{10}$$

with the RGB-coupled vector soft-thresholding solution

$$T_{k+1} = \mathrm{VST}\left(f - S_k - \frac{b_k}{\rho_k}, \frac{1}{\rho_k}\right), \tag{11}$$

where

$$\mathrm{VST}(z,a)=\left(1-\frac{a}{\|z\|_2}\right)_+ z, \quad z\neq 0, \qquad \mathrm{VST}(0,a)=0 \tag{12}$$

is applied to the RGB vector at each pixel.

4.2.2 Structure Update

After updating T_{k+1}, we define the pseudo-observation

$$Q_k = f - T_{k+1} - \frac{b_k}{\rho_k}, \tag{13}$$

and formally write the normalized structure subproblem as

$$S_{k+1} = \arg\min_S \frac{\lambda}{\rho_k}\Phi_\theta(S) + \frac{1}{\rho_k}R_{\mu,\gamma}(\nabla S) + \frac{1}{2}\|S-Q_k\|_2^2. \tag{14}$$

Equation (14) provides the fixed-prior ADMM reference subproblem. Since the joint proximal mapping of the implicit FM prior and the explicit gradient prior is unavailable, we realize the structure update through two sequential plug-and-play surrogates: generative-prior prediction followed by a level-adaptive gradient-prior refinement.

Generative-prior structure update. We denote the FM generative-prior operator at iteration k by $\mathcal{G}_{\theta,k}$:

$$\widetilde{S}_{k+1} = \mathcal{G}_{\theta,k}(Q_k). \tag{15}$$

Its flow-state construction is described in Sec. 4.3, while the patch-phase shifting used during JiT inference is detailed in Appendix C.

Gradient-prior refinement. For the explicit structural refinement, we employ a level-adaptive MCP continuation rather than directly scaling a fixed MCP penalty. At outer iteration k, we set

$$\mu_k = \frac{\mu}{\rho_k}, \tag{16}$$

while keeping γ fixed. Importantly, $R_{\mu_k,\gamma}$ is not identified with $\rho_k^{-1}R_{\mu,\gamma}$; for MCP these two operations are generally not equivalent. Instead, Eq. (16) defines the level-dependent regularizer used by our plug-and-play structure refinement. The MCP is defined as

$$\phi_{\mu_k,\gamma}(z) = \begin{cases} \mu_k z - \dfrac{z^2}{2\gamma}, & 0\le z\le \gamma\mu_k, \\ \dfrac{\gamma\mu_k^2}{2}, & z>\gamma\mu_k, \end{cases} \qquad \mu_k>0,\ \gamma>1. \tag{17}$$

Applied to the joint magnitude $\|J_d S(p)\|_2$, the resulting refinement is

$$S_{k+1} = \arg\min_S \frac{1}{2}\|S-\widetilde{S}_{k+1}\|_2^2 + \sum_{d,p}\phi_{\mu_k,\gamma}\left(\|J_d S(p)\|_2\right). \tag{18}$$

Because ρ_k increases along the Flow–ADMM schedule, μ_k decreases accordingly, yielding progressively weaker structural shrinkage at later flow levels.

We optimize a VAHQ-relaxed realization of Eq. (18) using VAHQ–FFT; the numerical details are provided in Appendix B.

4.2.3 Consistency Update

Finally, the dual variable is updated as

$$b_{k+1} = b_k + \rho_k\big(S_{k+1} + T_{k+1} - f\big). \tag{19}$$

This yields a plug-and-play ADMM procedure, since the learned FM operator is applied sequentially with the explicit gradient-prior refinement rather than as an exact joint proximal update.

4.3 Per-Step Clean–Noise Endpoint Modulation

Following the endpoint decomposition used in FlowDPS (Kim et al., 2025), under our noise-to-clean convention the endpoints associated with (x_k, t_k) are

$$\widehat{x}_{1|k} = x_k + (1-t_k)v_\theta(x_k,t_k), \qquad \widehat{x}_{0|k} = x_k - t_k v_\theta(x_k,t_k). \tag{20}$$

Algorithm 1 Flow-Filter

Require: Input image f, pretrained JiT model D_θ, flow levels $\{t_k\}_{k=0}^{K}$, parameters $\rho_0, \mu, \xi, \gamma, \alpha_0, \eta$
Ensure: Structure layer S_K and texture layer T_K
Initialize $S_0 \leftarrow f$, $T_0 \leftarrow 0$, $b_0 \leftarrow 0$, $\widetilde{x}_{0|0} \sim \mathcal{N}(0, I)$
for $k = 0, \ldots, K-1$ **do**
  Calculate ρ_k via Eq. (9)
  Update T_{k+1} via Eq. (11)
  Calculate Q_k via Eq. (13)
  Calculate $\widetilde{x}_{1|k}$ via Eqs. (22)–(23)
  Calculate x_k via Eq. (26)
  Predict $\widetilde{S}_{k+1}$ using $D_\theta(x_k, t_k)$
  Refresh $\widetilde{x}_{0|k+1}$ via Eqs. (21)–(25)
  Update S_{k+1} via Eq. (18)
  Update b_{k+1} via Eq. (19)
end for
return S_K, T_K

Since JiT directly predicts the clean endpoint,

$$\widehat{x}_{1|k} = D_\theta(x_k, t_k), \qquad \widehat{x}_{0|k} = \frac{x_k - t_k \widehat{x}_{1|k}}{1 - t_k}. \tag{21}$$

We use hats for predicted or recovered endpoints and tildes for controlled endpoints used to construct the next flow state. The modulation involves clean-endpoint anchoring (CEA) and noise-endpoint refresh (NER). Fig. 4 summarizes the per-step clean–noise endpoint modulation used to reconstruct the current flow state.

4.3.1 Clean-Endpoint Anchoring

To limit structural drift induced by the generative prior, we anchor the current pseudo-observation Q_k to the input f:

$$\widetilde{x}_{1|k} = (1 - \alpha_k) Q_k + \alpha_k f, \tag{22}$$

with a linearly decaying strength

$$\alpha_k = \alpha_0 \left(1 - \frac{k}{K-1}\right). \tag{23}$$

The stronger early anchoring preserves input-consistent structures, while its decay lets later updates rely increasingly on the current ADMM decomposition without reintroducing the original texture.

4.3.2 Noise-Endpoint Refresh

We initialize the noise endpoint as

$$\widetilde{x}_{0|0} \sim \mathcal{N}(0, I). \tag{24}$$

For $k \geq 1$, the previously recovered noise endpoint is partially refreshed with independent Gaussian noise:

$$\widetilde{x}_{0|k} = \sqrt{1 - \eta^2}\, \widehat{x}_{0|k-1} + \eta \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, I), \qquad \eta \in [0, 1]. \tag{25}$$

With ϵ_k independent of $\widehat{x}_{0|k-1}$, this update preserves identity covariance when $\mathrm{Cov}(\widehat{x}_{0|k-1}) = I$. For $\eta = 1$, it reduces to $\widetilde{x}_{0|k} = \epsilon_k$, i.e., full independent Gaussian refresh without noise-endpoint history. The controlled endpoints reconstruct the current FM state as

$$x_k = t_k \widetilde{x}_{1|k} + (1 - t_k) \widetilde{x}_{0|k}. \tag{26}$$

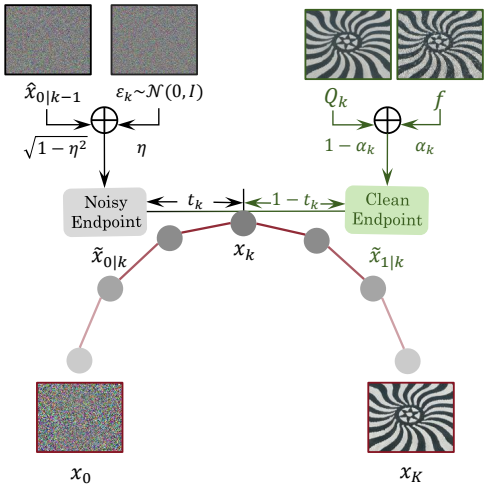
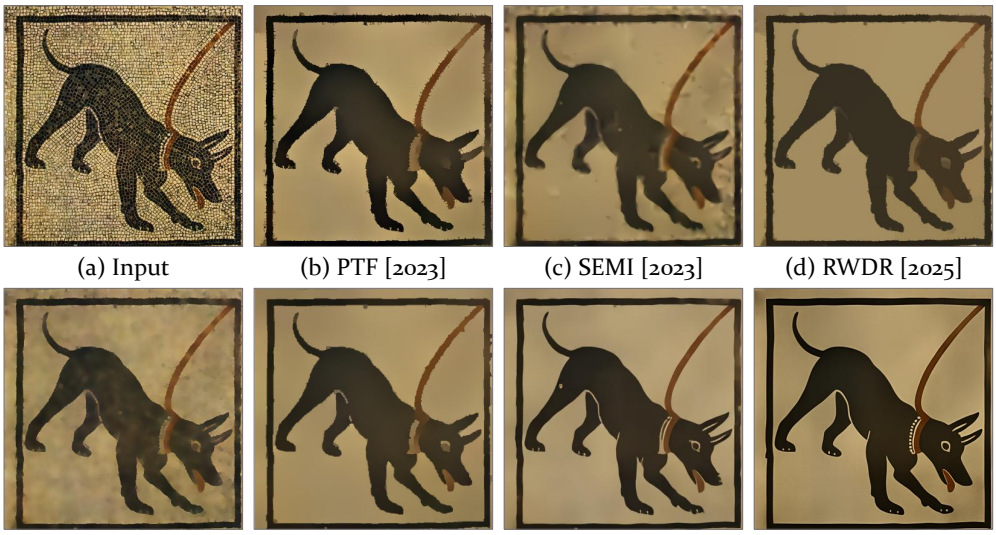


Figure 4: Per-step clean–noise endpoint modulation.

JiT predicts $\widehat{x}_{1|k}$ from (x_k, t_k), and Eq. (21) recovers $\widehat{x}_{0|k}$ for the next refresh. Reconstructing the flow state at every iteration keeps the generative prior coupled to the current decomposition, while noise refresh reduces the persistent propagation of high-frequency texture patterns.

4.4 Two-Dimensional Cyclic Patch-Phase Shifting

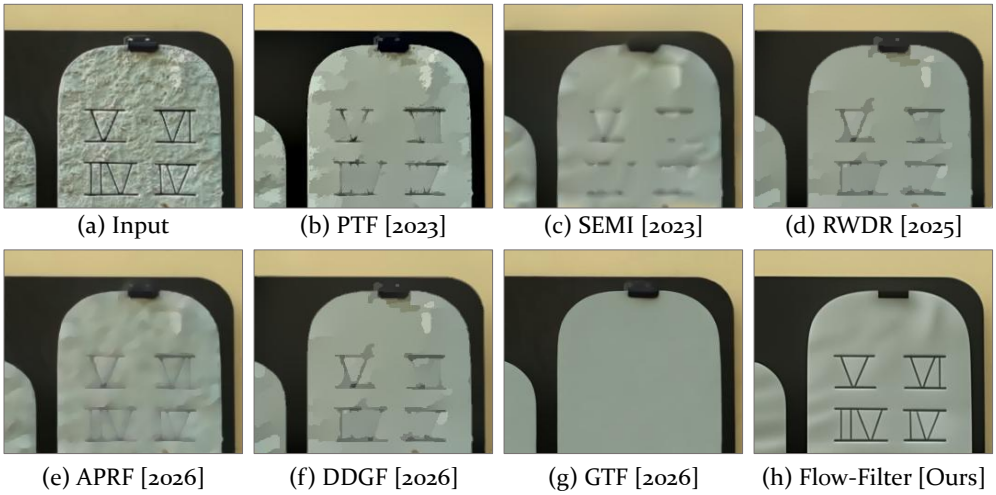
Repeated JiT inference on a fixed patch grid can reinforce patch-boundary prediction errors at the same spatial locations, producing phase-locked seam or block artifacts. We mitigate this effect with patch-phase shifting (PPS), i.e., we cyclically vary the patch phase across flow levels while mapping each prediction back to the original image coordinates. The strategy requires neither retraining nor additional network evaluations; the shift schedule and inference operator are detailed in Appendix C.



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 5: Qualitative comparison on the Dog example. Compared methods include PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 6: Qualitative comparison on the Stele example. Compared methods include PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

5 Experiments

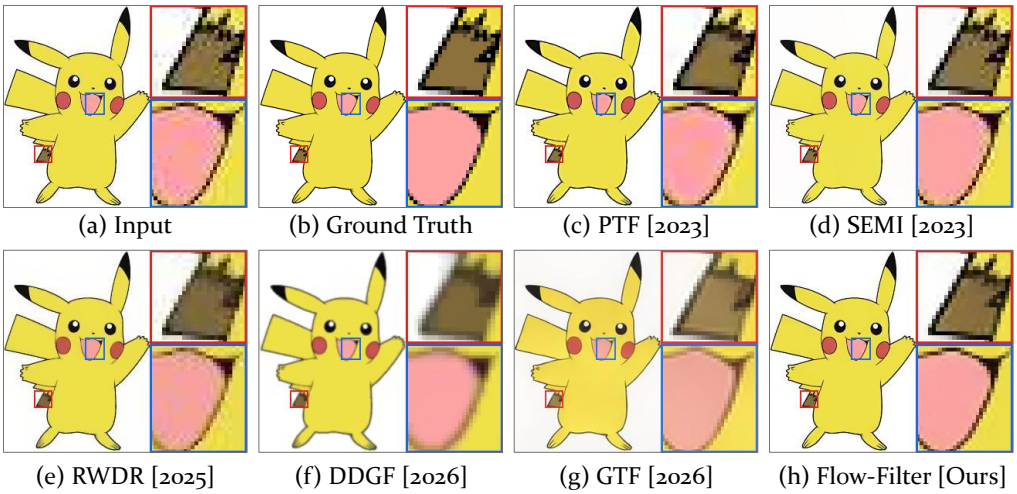
5.1 Texture Filtering

Texture filtering focuses on removing undesired textures while preserving important structures. We compare two groups of images with different challenges. The first group contains dense, high-contrast textures. As shown in Figure 5, most methods lose fine details while removing textures, whereas Flow-Filter effectively suppresses the textures and preserves the small circular structures on the dog's collar.

In the second group, textures and fine structures occur at similar scales and are therefore more difficult to distinguish. As shown in Figure 6, most methods struggle to remove the rough textures on the stone surface while preserving the thin structures of the Roman numerals, whereas Flow-Filter achieves a better balance between the two. Additional results are provided in Appendix G.

Table 1: Quantitative comparison of JPEG compression artifact removal under different quality factors. Higher PSNR and lower MDSI indicate better performance. The best and second-best results are highlighted.

Method	PSNR (↑)							MDSI (↓)						
	10%	20%	30%	40%	50%	60%	70%	10%	20%	30%	40%	50%	60%	70%
JPEG Input	27.50	29.64	31.11	31.97	32.81	33.46	34.67	0.3020	0.2724	0.2562	0.2450	0.2363	0.2281	0.2125
GSF (Liu et al. (2022))	28.43	30.47	31.71	32.29	32.75	33.07	33.59	0.2834	0.2585	0.2501	0.2459	0.2415	0.2394	0.2342
E2H (Feng et al. (2022))	27.27	28.59	29.96	30.08	30.39	30.47	31.08	0.2930	0.2701	0.2596	0.2536	0.2482	0.2458	0.2418
SDF (Ham et al. (2018))	27.72	29.51	30.60	31.18	31.72	32.08	32.61	0.3123	0.2942	0.2874	0.2830	0.2793	0.2768	0.2732
SKF (Sun et al. (2020))	27.38	29.42	30.77	31.55	32.29	32.84	33.90	0.2986	0.2672	0.2506	0.2395	0.2308	0.2232	0.2096
ELS (Liu et al. (2020a))	27.63	29.76	31.20	32.03	32.85	33.49	34.64	0.2979	0.2683	0.2525	0.2421	0.2328	0.2249	0.2109
ILS (Liu et al. (2020b))	26.67	28.40	29.58	30.22	30.88	31.38	32.16	0.2985	0.2721	0.2571	0.2479	0.2388	0.2315	0.2194
SWF (Yin et al. (2019))	26.84	28.45	29.44	29.96	30.42	30.75	31.32	0.3065	0.2801	0.2668	0.2593	0.2536	0.2482	0.2395
SCF (Yang et al. (2022a))	26.83	27.85	28.33	28.52	28.67	28.79	28.89	0.3303	0.3253	0.3251	0.3253	0.3252	0.3253	0.3255
L_0 (Yang et al. (2022b))	28.38	30.47	31.98	32.85	33.75	34.40	35.62	0.2800	0.2645	0.2322	0.2238	0.2160	0.2106	0.2045
L_1 (Yang et al. (2023))	27.60	29.35	30.56	31.23	31.89	32.39	33.14	0.2933	0.2691	0.2559	0.2484	0.2413	0.2356	0.2272
CAN (Chen et al. (2017))	27.53	29.19	30.14	30.70	31.20	31.58	32.06	0.2910	0.2719	0.2623	0.2574	0.2527	0.2494	0.2448
DGF (Wu et al. (2018))	27.27	28.94	29.87	30.51	31.06	31.50	32.04	0.2986	0.2756	0.2633	0.2563	0.2495	0.2440	0.2354
RES (Zhu et al. (2019))	27.89	29.92	30.95	31.78	32.51	33.10	33.72	0.2849	0.2581	0.2442	0.2367	0.2293	0.2237	0.2157
SEMI (Huang et al. (2023))	27.89	29.99	31.45	32.28	33.10	33.73	34.97	0.2986	0.2728	0.2593	0.2515	0.2450	0.2391	0.2289
DDGF (Huang et al. (2026))	28.90	31.32	33.02	33.90	34.70	35.36	36.60	0.2822	0.2431	0.2277	0.2168	0.2070	0.1977	0.1872
APRF (Yang et al. (2026))	29.07	31.32	33.06	33.99	34.85	35.54	36.79	0.2761	0.2418	0.2250	0.2153	0.2059	0.1996	0.1842
RWDR (Qi et al. (2025))	28.72	30.97	32.61	33.52	34.32	34.98	36.09	0.2731	0.2428	0.2276	0.2194	0.2119	0.2058	0.1976
PTF (Zhang et al. (2023))	29.16	31.13	32.42	33.05	33.55	33.96	34.78	0.2678	0.2463	0.2274	0.2224	0.2139	0.2100	0.2049
Flow-Filter	29.42	32.26	33.92	34.46	35.11	35.54	36.54	0.2626	0.2360	0.2189	0.2119	0.2021	0.1942	0.1853



(a) Input (b) Ground Truth (c) PTF [2023] (d) SEMI [2023]

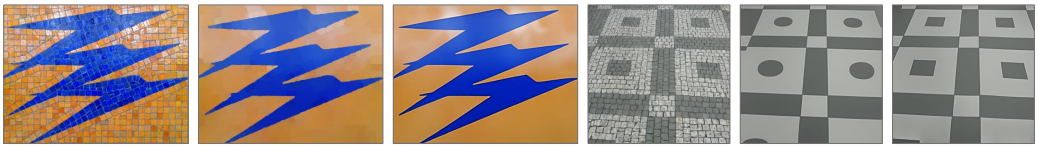
(e) RWDR [2025] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 7: Qualitative comparison of JPEG compression artifact removal. Compared methods include PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

5.2 JPEG Compression Artifact Removal

JPEG artifact removal evaluates the ability of a smoothing method to suppress compression distortions while preserving genuine structures. We generate JPEG-degraded images from the 50-image clip-art benchmark using quality factors $Q \in \{10, 20, 30, 40, 50, 60, 70\}$, with the uncompressed images as ground truth, and evaluate PSNR and MDSI. As shown in Table 1, Flow-Filter achieves the best or tied-best PSNR at six of seven quality factors and the lowest MDSI at six of seven quality factors.

As shown in Figure 7, most competing methods either leave noticeable compression artifacts or damage genuine structures during artifact removal. In contrast, Flow-Filter more thoroughly removes the artifacts while preserving fine details, such as the markings on Pikachu's tail and the contour around the mouth.



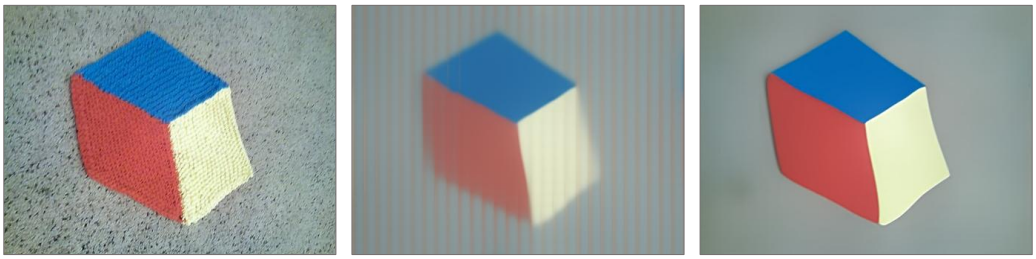
(a) Input (b) w/o $\Phi_\theta(S)$ (c) Ours (d) Input (e) w/o $R(\nabla S)$ (f) Ours

Figure 8: Ablation study on the generative and gradient priors.



(a) Input (b) w/o CEA (c) Ours (d) Input (e) w/o NER (f) Ours

Figure 9: Ablation study on clean-endpoint anchoring (CEA) and noise-endpoint refresh (NER).



(a) Input (b) w/o PPS (c) Ours

Figure 10: Ablation study on 2D cyclic patch-phase shifting (PPS).

5.3 Ablation Study

Generative and Gradient Priors. As shown in Figure 8, removing the generative prior still suppresses part of the texture but leads to unnatural structural variations and local discontinuities, while removing the gradient prior makes the output more prone to deviating from the input geometry. These results show that the two priors are complementary: the generative prior provides global natural-structure constraints, whereas the gradient prior preserves local geometry.

Clean-Endpoint Anchoring and Noise-Endpoint Refresh. As shown in Figure 9, removing clean-endpoint anchoring causes noticeable structural drift or deformation, while removing noise-endpoint refresh allows high-frequency textures to persist across successive flow levels. The two mechanisms respectively stabilize structure and suppress residual textures.

2D Cyclic Patch-Phase Shifting. As shown in Figure 10, a fixed patch grid causes prediction errors to repeatedly accumulate at the same patch boundaries, producing seam or block artifacts. The proposed 2D cyclic patch-phase shifting alleviates this problem by varying patch alignment across iterations.

6 Conclusion

We presented **Flow-Filter**, a plug-and-play framework that integrates a pretrained Flow Matching prior with explicit structural regularization for image smoothing. Rather than using the generative model as a direct smoother, our framework employs it for structural modeling and coordinates its predictions with the evolving structure–texture decomposition. Experiments on texture filtering and JPEG artifact removal demonstrate effective texture suppression while preserving salient structures. These results suggest that pretrained generative priors can serve as effective structural models for image smoothing beyond direct generative inference.

AI USE STATEMENT

Generative AI tools were used only for language polishing. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the final manuscript.

REFERENCES

Black Forest Labs. FLUX.1: Transformer-powered flow models at scale. Black Forest Labs model release, 2024.

Qifeng Chen, Jia Xu, and Vladlen Koltun. Fast image processing with fully-convolutional networks. In *IEEE International Conference on Computer Vision*, pp. 2497–2506, 2017.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021.

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, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, volume 235, pp. 12606–12633, 2024.

Qingnan Fan, Jiaolong Yang, David P. Wipf, Baoquan Chen, and Xin Tong. Image smoothing via unsupervised learning. *ACM Trans. Graph.*, 37(6):259:1–259:14, 2018.

Qingnan Fan, Dongdong Chen, Lu Yuan, Gang Hua, Nenghai Yu, and Baoquan Chen. A general decoupled learning framework for parameterized image operators. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1):33–47, 2021.

Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.*, 27(3):67:1–67:10, 2008.

Yifan Feng, Shun Deng, Xiaohui Yan, Xue Yang, Mingqiang Wei, and Ligang Liu. Easy2Hard: Learning to solve the intractables from a synthetic dataset for structure-preserving image smoothing. *IEEE Trans. Neural Networks Learn. Syst.*, 33(12):7223–7236, 2022.

Xiaojie Guo, Yu Li, and Haibin Ling. LIME: Low-light image enhancement via illumination map estimation. *IEEE Trans. Image Process.*, 26(2):982–993, 2017.

Bumsub Ham, Minsu Cho, and Jean Ponce. Robust guided image filtering using nonconvex potentials. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(1):192–207, 2018.

Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013. doi: 10.1109/TPAMI.2012.213.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Junqing Huang, Haihui Wang, Xuechao Wang, and Michael Ruzhansky. Semi-sparsity for smoothing filters. *IEEE Trans. Image Process.*, 32:1627–1639, 2023. doi: 10.1109/TIP.2023.3247181.

Yifan Huang, Lanling Zeng, and Yang Yang. Dual dynamic guidance image filtering. *Pattern Recognit.*, 175:113062, 2026. doi: 10.1016/j.patcog.2026.113062.

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

Mingjia Li, Yuanbin Fu, Xinhui Li, and Xiaojie Guo. Deep flexible structure preserving image smoothing. In *ACM International Conference on Multimedia*, pp. 1875–1883, 2022.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Wei Liu, Ping Zhang, Xin Chen, Chunhua Shen, Xiaolin Huang, and Jie Yang. Embedding bilateral filter in least squares for efficient edge-preserving image smoothing. *IEEE Trans. Circuits Syst. Video Technol.*, 30(1):23–35, 2020a.

Wei Liu, Pingping Zhang, Xiaolin Huang, Jie Yang, Chunhua Shen, and Ian Reid. Real-time image smoothing via iterative least squares. *ACM Trans. Graph.*, 39(3):28:1–28:24, 2020b.

Wei Liu, Ping Zhang, Yinjie Lei, Xiaolin Huang, Jie Yang, and Michael K. Ng. A generalized framework for edge-preserving and structure-preserving image smoothing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):6631–6648, 2022.

Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. PnP-Flow: Plug-and-play image restoration with flow matching. In *International Conference on Learning Representations*, 2025.

Huiqing Qi, Xiaoliu Luo, Tingting Li, and Fang Li. Edge-aware image smoothing with relative wavelet domain representation. In *International Conference on Learning Representations*, 2025.

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

Zhiqiang Sun, Bo Han, Jinjun Li, Jian Zhang, and Xinbo Gao. Weighted guided image filtering with steering kernel. *IEEE Trans. Image Process.*, 29:500–508, 2020.

Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1838–1847, 2018.

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via L_0 gradient minimization. *ACM Trans. Graph.*, 30(6):174:1–174:12, 2011.

Li Xu, Jimmy S. J. Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. Deep edge-aware filters. In *International Conference on Machine Learning*, volume 37, pp. 1669–1678, 2015.

Yang Yang, Hongjun Hui, Lanling Zeng, Yang Zhao, Yizhou Zhan, and Tao Yan. Edge-preserving image filtering based on soft clustering. *IEEE Trans. Circuits Syst. Video Technol.*, 32(7):4150–4162, 2022a.

Yang Yang, Ling Tang, Lanling Zeng, Xinyu Wang, and Yongzhao Zhan. L_0 image smoothing via iterating truncated L_1 gradient regularization. *J. Electron. Imaging.*, 31(5):053016, 2022b. doi: 10.1117/1.JEI.31.5.053016.

Yang Yang, Hui Zheng, Lanling Zeng, Xiaohui Shen, and Yizhou Zhan. L_1-regularized reconstruction model for edge-preserving filtering. *IEEE Trans. Multim.*, 25:4148–4162, 2023.

Yang Yang, Ling Tang, Tao Yan, Lanling Zeng, Xiangjun Shen, and Yongzhao Zhan. Parameterized L_0 image smoothing with unsupervised learning. *IEEE Trans. Emerg. Top. Comput. Intell.*, 8(2): 1938–1951, 2024.

Yang Yang, Shunli Ji, Lanling Zeng, and Keyang Cheng. Adaptive proximal regularization for image smoothing. *Pattern Recognit.*, 171:112176, 2026. doi: 10.1016/j.patcog.2025.112176.

Hui Yin, Yuanhao Gong, and Guoping Qiu. Side window filtering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8758–8766, 2019.

Qing Zhang, Hao Jiang, Yongwei Nie, and Wei-Shi Zheng. Pyramid texture filtering. *ACM Trans. Graph.*, 42(4):110:1–110:11, 2023. doi: 10.1145/3592120.

Rongjia Zheng, Shangwei Huang, Lei Zhu, Wei-Shi Zheng, and Qing Zhang. Generative texture filtering. In *ACM SIGGRAPH Conference Papers*, pp. 82:1–82:11, 2026.

Zhiwei Zhong, Xianming Liu, Junjun Jiang, Debin Zhao, and Xiangyang Ji. Deep attentional guided image filtering. *IEEE Trans. Neural Networks Learn. Syst.*, 35(9):12236–12250, 2024.

Feida Zhu, Zhetong Liang, Xue Jia, Lei Zhang, and Yizhou Yu. A benchmark for edge-preserving image smoothing. *IEEE Trans. Image Process.*, 28(7):3556–3570, 2019.

A Theoretical Properties of Flow–ADMM Coupling

This appendix establishes two properties of the coupling between the Flow Matching (FM) prior and the outer optimization. First, we formalize the denoising-level correspondence used to determine the ADMM penalty schedule. Second, we show that the endpoint modulation in Sec. 4.3 is algebraically consistent with the linear FM path, while clean-side anchoring contracts the current clean endpoint toward the input and noise-side refresh preserves the nominal unit covariance under the stated assumptions. These results concern scale correspondence and endpoint consistency only; they do not require the implicit generative prior Φ_θ to admit an explicitly evaluable energy, nor do they interpret a single JiT clean-endpoint prediction as an exact proximal mapping of Φ_θ.

A.1 Flow–ADMM Denoising-Level Alignment

Proposition 1 (Unique Flow–ADMM denoising-level alignment). Let $\tau_k > 0$ denote the nominal perturbation variance at the k-th FM level, and let $\rho_k > 0$ denote the ADMM penalty used by the corresponding structure update. For a fixed $\lambda > 0$, suppose that the normalized generative-prior weight remains proportional to the nominal FM perturbation variance, i.e.,

$$\frac{\lambda/\rho_k}{\tau_k} = \frac{\lambda/\rho_0}{\tau_0}, \qquad k = 0, \dots, K-1. \tag{27}$$

Then, for a fixed initial pair (τ_0, ρ_0), the unique positive penalty schedule satisfying Eq. (27) is

$$\rho_k = \rho_0 \frac{\tau_0}{\tau_k}, \qquad k = 0, \dots, K-1. \tag{28}$$

For the linear noise-to-clean FM path used in Flow-Filter,

$$x_k = t_k x_1 + (1 - t_k) x_0, \qquad \sigma_k = 1 - t_k, \tag{29}$$

the nominal perturbation variance is

$$\tau_k = \sigma_k^2 = (1 - t_k)^2, \tag{30}$$

and therefore

$$\rho_k = \rho_0 \left(\frac{1 - t_0}{1 - t_k} \right)^2, \qquad k = 0, \dots, K-1. \tag{31}$$

Moreover, for $k = 0, \dots, K-2$, $t_{k+1} > t_k$ implies $\tau_{k+1} < \tau_k$ and $\rho_{k+1} > \rho_k$; hence the effective scale $1/\rho_k$ decreases monotonically with the nominal FM perturbation level.

Proof. Consider an auxiliary Gaussian denoising model used only to identify the relative prior scale. Given a pseudo-observation Q and a nominal noise variance $\tau > 0$, let $c > 0$ denote a level-independent constant associated with the overall scale of the generative prior:

$$S_\tau \in \arg\min_S \left\{ c\, \Phi_\theta(S) + \frac{1}{2\tau} \|S - Q\|_2^2 \right\}. \tag{32}$$

Multiplying the objective by the positive scalar τ leaves its minimizers unchanged, giving

$$S_\tau \in \arg\min_S \left\{ c\tau\, \Phi_\theta(S) + \frac{1}{2} \|S - Q\|_2^2 \right\}. \tag{33}$$

Thus, relative to the unit-weight quadratic term, the nominal generative-prior scale is $c\tau$.

In the normalized Flow–ADMM structure subproblem, the coefficient of the generative-prior term relative to the quadratic data term is λ/ρ_k. Matching only these relative scales gives

$$\frac{\lambda}{\rho_k} = c\tau_k. \tag{34}$$

At $k = 0$,

$$\frac{\lambda}{\rho_0} = c\tau_0, \tag{35}$$

so that $c = \lambda/(\rho_0\tau_0)$. Substituting this expression into Eq. (34) yields

$$\frac{\lambda}{\rho_k} = \frac{\lambda}{\rho_0\tau_0}\tau_k, \tag{36}$$

and, since $\lambda > 0$,

$$\rho_k = \rho_0\frac{\tau_0}{\tau_k}. \tag{37}$$

For fixed positive (τ_0, ρ_0) and any $\tau_k > 0$, the right-hand side determines exactly one positive ρ_k; hence the schedule is unique under the scale-matching condition in Eq. (27).

Under the linear FM parameterization in Eq. (29), the unit-covariance noise endpoint x_0 enters the intermediate state with amplitude $\sigma_k = 1 - t_k$. Its nominal perturbation variance is therefore $\tau_k = (1 - t_k)^2$. Substitution into Eq. (28) gives Eq. (31). Finally, for $k = 0, \ldots, K - 2$, we have $t_k < t_{k+1} < 1$, so

$$0 < 1 - t_{k+1} < 1 - t_k,$$

which implies $\tau_{k+1} < \tau_k$. Eq. (28) then gives $\rho_{k+1} > \rho_k$, or equivalently $1/\rho_{k+1} < 1/\rho_k$.

Remark. Proposition 1 establishes a denoising-level correspondence, not a variational equivalence between JiT inference and the ADMM structure subproblem. In particular, the argument neither assumes that Φ_θ has an explicitly evaluable energy nor requires the JiT clean-endpoint predictor to be an exact proximal operator. The terminal level $t_K = 1$ is used only as the final clean endpoint and is not inserted into Eq. (31) for an additional structure update. The initial pair (ρ_0, τ_0) determines the calibration constant $c = \lambda/(\rho_0\tau_0)$. Thus, λ cancels from the relative penalty schedule and is not an independently tuned parameter of the PnP implementation.

A.2 Consistency and Stability of Endpoint Modulation

We next record three properties of the endpoint modulation used in Flow-Filter.

Lemma 1 (Path consistency and stability of endpoint modulation). Let $t_k \in [0, 1)$ and let $\widehat{x}_{1|k}$ denote the clean endpoint predicted from the current FM state x_k. Define the recovered noise endpoint by

$$\widehat{x}_{0|k} = \frac{x_k - t_k\widehat{x}_{1|k}}{1 - t_k}. \tag{38}$$

Then the following properties hold.

(i) FM path consistency. The recovered endpoints exactly reconstruct the current FM state:

$$x_k = t_k\widehat{x}_{1|k} + (1 - t_k)\widehat{x}_{0|k}. \tag{39}$$

With

$$v_k = \frac{\widehat{x}_{1|k} - x_k}{1 - t_k}, \tag{40}$$

the explicit Euler update to any $t' \in [0, 1]$ satisfies

$$x_k + (t' - t_k)v_k = t'\widehat{x}_{1|k} + (1 - t')\widehat{x}_{0|k}. \tag{41}$$

(ii) Clean-side contraction toward the input. For

$$\widetilde{x}_{1|k} = (1 - \alpha_k)Q_k + \alpha_k f, \qquad \alpha_k \in [0, 1], \tag{42}$$

we have

$$\left\|\widetilde{x}_{1|k} - f\right\|_2 = (1 - \alpha_k)\left\|Q_k - f\right\|_2 \le \left\|Q_k - f\right\|_2. \tag{43}$$

(iii) Covariance-preserving noise refresh. Let

$$\widetilde{x}_{0|k} = \sqrt{1 - \eta^2}\,\widehat{x}_{0|k-1} + \eta\varepsilon_k, \qquad \varepsilon_k \sim \mathcal{N}(0, I), \quad \eta \in [0, 1]. \tag{44}$$

Assume that ε_k is independent of $\widehat{x}_{0|k-1}$, $\mathbb{E}[\widehat{x}_{0|k-1}] = 0$, and $\mathrm{Cov}(\widehat{x}_{0|k-1}) = \Sigma$. Then

$$\mathrm{Cov}\left(\widetilde{x}_{0|k}\right) = (1-\eta^2)\Sigma + \eta^2 I, \tag{45}$$

and

$$\mathrm{Cov}\left(\widetilde{x}_{0|k}, \widehat{x}_{0|k-1}\right) = \sqrt{1-\eta^2}\,\Sigma. \tag{46}$$

In particular, if $\Sigma = I$, then $\mathrm{Cov}(\widetilde{x}_{0|k}) = I$.

Proof. Eq. (39) follows directly by rearranging Eq. (38). Substituting it into Eq. (40) gives $v_k = \widehat{x}_{1|k} - \widehat{x}_{0|k}$, and therefore

$$\begin{aligned} x_k + (t' - t_k)v_k &= t_k\widehat{x}_{1|k} + (1-t_k)\widehat{x}_{0|k} + (t'-t_k)\left(\widehat{x}_{1|k} - \widehat{x}_{0|k}\right) \\ &= t'\widehat{x}_{1|k} + (1-t')\widehat{x}_{0|k}. \end{aligned} \tag{47}$$

This proves the path-consistency statement.

For clean-side anchoring,

$$\widetilde{x}_{1|k} - f = (1-\alpha_k)(Q_k - f), \tag{48}$$

and Eq. (43) follows by taking the Euclidean norm.

For noise-side refresh, let $a = \sqrt{1-\eta^2}$. By independence and the zero-mean assumption,

$$\begin{aligned} \mathrm{Cov}\left(\widetilde{x}_{0|k}\right) &= \mathrm{Cov}\left(a\widehat{x}_{0|k-1} + \eta\varepsilon_k\right) \\ &= a^2\Sigma + \eta^2 I = (1-\eta^2)\Sigma + \eta^2 I, \end{aligned} \tag{49}$$

and

$$\begin{aligned} \mathrm{Cov}\left(\widetilde{x}_{0|k}, \widehat{x}_{0|k-1}\right) &= a\ \mathrm{Cov}\left(\widehat{x}_{0|k-1}\right) \\ &= \sqrt{1-\eta^2}\,\Sigma. \end{aligned} \tag{50}$$

Setting $\Sigma = I$ gives covariance preservation. □

Remark. Lemma 1 makes no claim that noise refresh alone guarantees texture removal, nor that Eq. (43) contracts an arbitrary semantic structure error. It characterizes only the exact algebraic and second-order statistical properties of the modulation itself.

B VAHQ–FFT Optimization of the Mixed-Order Gradient Prior

This appendix details the optimization of the explicit mixed-order gradient prior. At each outer Flow–ADMM iteration k, JiT first produces the generative-prior structure candidate $\widetilde{S}_{k+1}$, after which the mixed-order differential-jet MCP regularizer refines local geometry. We use a vectorial additive half-quadratic (VAHQ) splitting that admits a closed-form auxiliary-jet update and an FFT-based image update on the padded working domain. We then establish a sufficient-descent result for the resulting VAHQ objective and connect it to the stopping quantity used by the implementation.

B.1 Mixed-Order Differential-Jet MCP Regularization

Following Sec. 4.1, for direction d and pixel p, define

$$J_d S(p) = \begin{bmatrix} \nabla_d S(p) \\ \xi\,\nabla_d^2 S(p) \end{bmatrix}, \tag{51}$$

where $\xi > 0$ balances the first- and second-order differential responses. For RGB images, $\|J_d S(p)\|_2$ is evaluated jointly over the color channels and the two differential orders.

The gradient-prior refinement introduced in the main paper is

$$S_{k+1} = \arg\min_S \frac{1}{2}\left\|S - \widetilde{S}_{k+1}\right\|_2^2 + \sum_{d,p} \phi_{\mu_k,\gamma}\left(\|J_d S(p)\|_2\right), \tag{52}$$

Here $\mu_k = \mu/\rho_k$ is the level-adaptive MCP parameter defined in Eq. (16). For a fixed outer iteration k, μ_k and γ remain constant throughout the inner VAHQ iterations. This continuation does not imply the identity $\rho_k^{-1} R_{\mu,\gamma} = R_{\mu_k,\gamma}$.

B.2 VAHQ Relaxation and Auxiliary-Jet Update

For each direction d, introduce an auxiliary variable Z_d with the same dimension as $J_d S$. For a fixed penalty $\beta_{\mathrm{v}} > 0$, consider

$$\begin{aligned}\mathcal{E}_{k,\beta_{\mathrm{v}}}(S, \{Z_d\}) = \frac{1}{2}\left\|S - \widetilde{S}_{k+1}\right\|_2^2 + \sum_{d,p} \phi_{\mu_k,\gamma}\left(\|Z_d(p)\|_2\right) \\ + \frac{\beta_{\mathrm{v}}}{2}\sum_d \|Z_d - J_d S\|_2^2 .\end{aligned} \tag{53}$$

For finite β_{v}, Eq. (53) is a VAHQ-relaxed realization of Eq. (52); it is not an exact reparameterization of the original MCP objective.

With $S^{(j)}$ fixed, the auxiliary variables decouple over directions and spatial locations. Let

$$Y = J_d S^{(j)}(p), \qquad r = \|Y\|_2. \tag{54}$$

The corresponding vector subproblem is

$$Z_d^{(j+1)}(p) = \arg\min_Z \left\{\phi_{\mu_k,\gamma}\left(\|Z\|_2\right) + \frac{\beta_{\mathrm{v}}}{2}\|Z - Y\|_2^2\right\}. \tag{55}$$

When $\beta_{\mathrm{v}}\gamma > 1$, this subproblem is strongly convex and its unique minimizer is

$$Z_d^{(j+1)}(p) = \begin{cases} 0, & r \le \mu_k/\beta_{\mathrm{v}}, \\ \dfrac{r - \mu_k/\beta_{\mathrm{v}}}{1 - 1/(\beta_{\mathrm{v}}\gamma)}\dfrac{Y}{r}, & \mu_k/\beta_{\mathrm{v}} < r < \gamma\mu_k, \\ Y, & r \ge \gamma\mu_k. \end{cases} \tag{56}$$

Thus, weak mixed-order jets are eliminated, intermediate jets undergo firm shrinkage, and sufficiently strong jets lie in the unbiased region of MCP.

B.3 FFT-Based Image Update

With $\{Z_d^{(j+1)}\}$ fixed, the S-subproblem is

$$S^{(j+1)} = \arg\min_S \left\{\frac{1}{2}\left\|S - \widetilde{S}_{k+1}\right\|_2^2 + \frac{\beta_{\mathrm{v}}}{2}\sum_d \left\|Z_d^{(j+1)} - J_d S\right\|_2^2\right\}. \tag{57}$$

Writing

$$Z_d = \left[Z_d^{(1)}, Z_d^{(2)}\right],$$

the normal equation becomes

$$\left[I + \beta_{\mathrm{v}}\sum_d \left(\nabla_d^*\nabla_d + \xi^2(\nabla_d^2)^*\nabla_d^2\right)\right] S = \widetilde{S}_{k+1} + \beta_{\mathrm{v}}\sum_d \left[\nabla_d^* Z_d^{(1)} + \xi(\nabla_d^2)^* Z_d^{(2)}\right]. \tag{58}$$

On the padded working domain, periodic directional differences diagonalize the operators under the two-dimensional discrete Fourier transform. Let $H_d(\omega)$ denote the frequency response of the first-order directional difference. Then

$$\Lambda(\omega) = \sum_d \left(|H_d(\omega)|^2 + \xi^2|H_d(\omega)|^4\right). \tag{59}$$

If R denotes the right-hand side of Eq. (58), the image update is

$$\widehat{S}^{(j+1)}(\omega) = \frac{\widehat{R}(\omega)}{1 + \beta_{\mathrm{v}}\Lambda(\omega)}. \tag{60}$$

Since $\Lambda(\omega) \ge 0$, the denominator is strictly positive at every frequency.

B.4 Convergence Analysis

Let Z denote the collection of all directional auxiliary jets.

Theorem 1 (Sufficient descent and subsequential stationarity of VAHQ–FFT). Fix an outer iteration k and suppose that $\gamma > 1$ and $\beta_{\mathrm{v}}\gamma > 1$. Let $\{(S^{(j)}, Z^{(j)})\}_{j\geq 0}$ be generated by exact minimization of the Z-block in Eq. (55), followed by exact minimization of the S-block in Eq. (57). Then

$$\begin{aligned}\mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j)}, Z^{(j)}\right) - \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j+1)}, Z^{(j+1)}\right)\\ \geq \frac{1}{2}\left\|S^{(j+1)} - S^{(j)}\right\|_2^2 + \frac{\beta_{\mathrm{v}} - 1/\gamma}{2}\left\|Z^{(j+1)} - Z^{(j)}\right\|_2^2.\end{aligned} \tag{61}$$

Consequently, $\mathcal{E}_{k,\beta_{\mathrm{v}}}(S^{(j)}, Z^{(j)})$ is monotonically nonincreasing and convergent, and

$$\left\|S^{(j+1)} - S^{(j)}\right\|_2 \to 0, \qquad \left\|Z^{(j+1)} - Z^{(j)}\right\|_2 \to 0. \tag{62}$$

The iterate sequence is bounded, and every accumulation point $(S^\star, Z^\star)$ is a critical point of $\mathcal{E}_{k,\beta_{\mathrm{v}}}$.

Proof. Define

$$g_k(z) = \phi_{\mu_k,\gamma}\left(\|z\|_2\right). \tag{63}$$

The MCP radial penalty g_k is $1/\gamma$-weakly convex because

$$g_k(z) + \frac{1}{2\gamma}\|z\|_2^2 = \begin{cases}\mu_k\|z\|_2, & \|z\|_2 \leq \gamma\mu_k,\\ \dfrac{\gamma\mu_k^2}{2} + \dfrac{1}{2\gamma}\|z\|_2^2, & \|z\|_2 > \gamma\mu_k.\end{cases} \tag{64}$$

The two convex branches match continuously with the same radial derivative at $\|z\|_2 = \gamma\mu_k$. Hence, for fixed Y,

$$z \mapsto g_k(z) + \frac{\beta_{\mathrm{v}}}{2}\|z - Y\|_2^2$$

is $(\beta_{\mathrm{v}} - 1/\gamma)$-strongly convex whenever $\beta_{\mathrm{v}}\gamma > 1$. Exact minimization of the Z-block therefore gives

$$\begin{aligned}\mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j)}, Z^{(j)}\right) - \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j)}, Z^{(j+1)}\right)\\ \geq \frac{\beta_{\mathrm{v}} - 1/\gamma}{2}\left\|Z^{(j+1)} - Z^{(j)}\right\|_2^2.\end{aligned} \tag{65}$$

For fixed $Z^{(j+1)}$, the Hessian of the S-subproblem is

$$I + \beta_{\mathrm{v}}\sum_d J_d^* J_d \succeq I. \tag{66}$$

Thus the S-subproblem is 1-strongly convex and has the unique minimizer given by Eq. (60). Hence

$$\begin{aligned}\mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j)}, Z^{(j+1)}\right) - \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(j+1)}, Z^{(j+1)}\right)\\ \geq \frac{1}{2}\left\|S^{(j+1)} - S^{(j)}\right\|_2^2.\end{aligned} \tag{67}$$

Adding Eqs. (65) and (67) proves Eq. (61).

All terms in Eq. (53) are nonnegative, so the objective is bounded below. Summing Eq. (61) over j gives

$$\sum_{j=0}^{\infty}\left\|S^{(j+1)} - S^{(j)}\right\|_2^2 < \infty, \qquad \sum_{j=0}^{\infty}\left\|Z^{(j+1)} - Z^{(j)}\right\|_2^2 < \infty, \tag{68}$$

which implies Eq. (62).

Moreover,

$$\frac{1}{2}\left\|S^{(j)} - \widetilde{S}_{k+1}\right\|_2^2 \leq \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^{(0)}, Z^{(0)}\right), \tag{69}$$

so $\{S^{(j)}\}$ is bounded. Since J_d is linear on the finite-dimensional image space, $\{J_d S^{(j)}\}$ is bounded. The quadratic splitting term then implies boundedness of $\{Z^{(j)}\}$, and the iterate sequence therefore has at least one accumulation point.

Let $(S^{(j_\ell)}, Z^{(j_\ell)}) \to (S^\star, Z^\star)$. By Eq. (62), the shifted subsequence has the same limit. Define the aggregate penalty $G_k(Z) = \sum_{d,p} g_k(Z_d(p))$. Exact minimization of the two blocks gives

$$0 \in \partial G_k\left(Z^{(j+1)}\right) + \beta_{\mathrm{v}}\left(Z^{(j+1)} - JS^{(j)}\right), \tag{70}$$

and

$$0 = S^{(j+1)} - \widetilde{S}_{k+1} + \beta_{\mathrm{v}} J^*\left(JS^{(j+1)} - Z^{(j+1)}\right), \tag{71}$$

where J stacks all directional mixed-order operators. Passing to the limit along the convergent subsequence yields

$$0 \in \partial_Z \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^\star, Z^\star\right), \qquad 0 = \nabla_S \mathcal{E}_{k,\beta_{\mathrm{v}}}\left(S^\star, Z^\star\right). \tag{72}$$

Hence every accumulation point is critical for $\mathcal{E}_{k,\beta_{\mathrm{v}}}$. □

Moreau-envelope interpretation. Define

$$\overline{g}_{k,\beta_{\mathrm{v}}}(y) = \min_z \left\{ g_k(z) + \frac{\beta_{\mathrm{v}}}{2}\|z - y\|_2^2 \right\}. \tag{73}$$

Under $\beta_{\mathrm{v}}\gamma > 1$, the minimizer is unique. Eliminating the auxiliary jets therefore gives

$$\mathcal{F}_{k,\beta_{\mathrm{v}}}(S) = \frac{1}{2}\left\|S - \widetilde{S}_{k+1}\right\|_2^2 + \sum_{d,p} \overline{g}_{k,\beta_{\mathrm{v}}}\left(J_d S(p)\right). \tag{74}$$

The criticality conclusion above is equivalently a stationarity statement for Eq. (74) after eliminating the unique auxiliary minimizers. This interpretation concerns only the explicit gradient-prior refinement and imposes no proximal interpretation on JiT.

B.5 Boundary Handling and Stopping Criterion

Periodic differences are convenient for Fourier diagonalization but would introduce direct wrap-around if applied to the original image domain. Before the VAHQ–FFT iterations, we therefore extend $\widetilde{S}_{k+1}$ by reflection padding. Periodic operators are applied only on this enlarged working domain, and the result is cropped back to the original spatial extent after termination.

Let $S^{(j)}$ and $S^{(j+1)}$ denote two consecutive VAHQ image iterates. The implementation uses

$$\Delta^{(j)} = \frac{\left\|S^{(j+1)} - S^{(j)}\right\|_2}{\max\left(\|S^{(j)}\|_2, \varepsilon_{\mathrm{num}}\right)} \tag{75}$$

as the stopping criterion, where $\varepsilon_{\mathrm{num}} > 0$ is used only for numerical stability. The iteration terminates once $\Delta^{(j)}$ falls below the prescribed tolerance or the maximum number of iterations is reached. Auxiliary-variable changes are retained as diagnostics but are not used for stopping, since MCP active-set switches can remain nonsmooth even when the image iterate is already close to stationarity.

Corollary 1. Under the assumptions of Theorem 1,

$$\Delta^{(j)} \to 0. \tag{76}$$

Proof. Theorem 1 gives $\|S^{(j+1)} - S^{(j)}\|_2 \to 0$, while the denominator in Eq. (75) is bounded below by $\varepsilon_{\mathrm{num}} > 0$. Therefore the ratio converges to zero. □

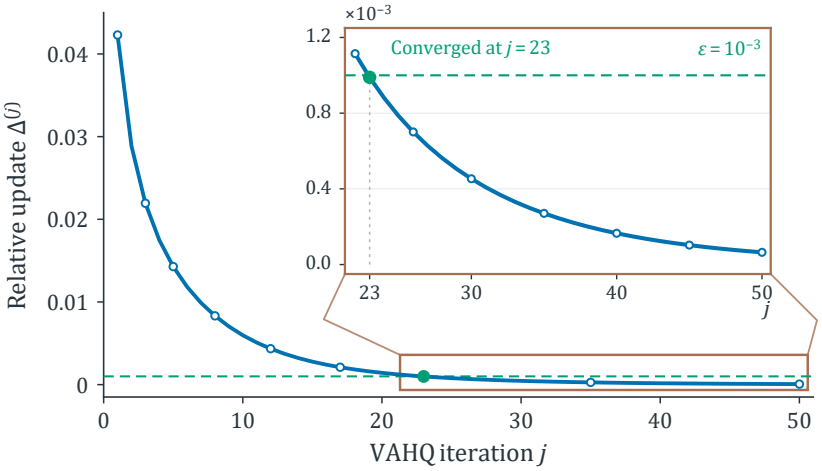


Figure 11: Convergence behavior of a representative VAHQ–FFT gradient-regularization subproblem. The relative update $\Delta^{(j)}$ decreases steadily and reaches the prescribed tolerance $\epsilon = 10^{-3}$ at iteration $j = 23$.

C Two-Dimensional Cyclic Patch-Phase Shifting

Repeated JiT inference on a fixed patch grid can reinforce patch-boundary prediction errors at the same spatial locations, producing phase-locked seam or block artifacts. To decorrelate these errors across flow levels, we introduce two-dimensional cyclic patch-phase shifting.

Let P denote the JiT patch size and define the basic shift unit as

$$q = P/4. \tag{77}$$

We use the eight-phase cycle

$$C = [(0,0),(1,1),(2,2),(3,3),(0,2),(1,3),(2,0),(3,1)], \tag{78}$$

with the shift at iteration k given by

$$\delta_k = qC_{k \bmod 8}. \tag{79}$$

Let $\mathcal{T}_\delta$ denote a two-dimensional cyclic shift. JiT inference is performed as

$$\hat{x}_{1|k} = \mathcal{T}_{-\delta_k}\left[D_\theta\left(\mathcal{T}_{\delta_k} x_k, t_k\right)\right]. \tag{80}$$

Thus, the patch alignment varies across flow levels while the prediction is restored to the original coordinates, preventing errors from repeatedly accumulating at fixed patch boundaries. The strategy requires only one JiT forward pass per flow level and introduces neither retraining nor model modification.

D Convergence Behavior of the Gradient-Regularization Subproblem

We examine the numerical behavior of VAHQ–FFT by recording the relative update within each gradient-regularization subproblem. For the j-th VAHQ image update, we use the same quantity as the stopping criterion in Appendix B:

$$\Delta^{(j)} = \frac{\left\|S^{(j+1)} - S^{(j)}\right\|_2}{\max\left(\|S^{(j)}\|_2, \varepsilon_{\text{num}}\right)}. \tag{81}$$

Because this quantity is also used by the actual solver, it directly measures how quickly the mixed-order differential-jet MCP refinement approaches its numerical fixed point.

As shown in Figure 11, the relative update decreases rapidly during the early iterations and then approaches the stopping tolerance smoothly. In this representative example, the criterion $\Delta^{(j)} \leq$

Table 2: Experiment-specific parameter settings of Flow-Filter for the main-text experiments. Values are rounded to two decimal places for display.

Experiment	Figure / Setting	K	ρ_0	α_0	$\sigma_{\max}$	η
Texture filtering	Figure 5	50.00	10.00	0.10	0.35	1.00
Texture filtering	Figure 6	50.00	10.00	0.10	0.25	1.00
JPEG artifact removal	$Q = 10$	70.00	1.91	0.15	0.37	0.98
JPEG artifact removal	$Q = 20$	40.00	0.84	0.55	0.42	0.95
JPEG artifact removal	$Q = 30$	10.00	0.57	1.00	0.47	1.00
JPEG artifact removal	$Q = 40$	10.00	0.49	1.00	0.47	1.00
JPEG artifact removal	$Q = 50$	10.00	0.69	1.00	0.27	1.00
JPEG artifact removal	$Q = 60$	10.00	0.95	1.00	0.19	1.00
JPEG artifact removal	$Q = 70$	10.00	0.95	1.00	0.14	1.00

10^{-3} is satisfied at $j = 23$, showing that the VAHQ–FFT iterations reach a stable fixed point within a moderate number of updates.

This analysis concerns the numerical convergence of the explicit gradient-regularization subproblem rather than a theoretical convergence guarantee for the complete plug-and-play Flow–ADMM procedure.

E Implementation Details

E.1 Inference and Solver Details

Flow-Filter is implemented in PyTorch and uses a pretrained JiT-H/16 model as its pixel-space generative prior. The model is kept in evaluation mode, with its weights fixed throughout inference. RGB inputs are represented in $[-1, 1]$. For network inference, the input is reflection-padded along the bottom and right boundaries to multiples of the patch size, and the prediction is subsequently cropped back to the input dimensions. Patch-phase shifting follows the schedule described in Appendix C, with one network evaluation per outer iteration.

Network inference uses BF16 autocasting on supported CUDA devices and FP16 otherwise. The reference implementation uses label index 1000 and random seed 1234. The VAHQ–FFT update equations are provided in Appendix B. The solver operates in FP32 and terminates when the relative change in the structure estimate is at most 10^{-3} or after 100 inner iterations. Experiment-specific parameter settings are reported in Section E.2.

The experiments are conducted on an `NVIDIA GeForce RTX 5080` GPU using Python `3.11.9`, PyTorch `2.11.0`, and CUDA `12.8`.

E.2 Parameter Settings

We fix the joint-gradient prior parameters to $\mu = 1.20$, $\xi = 0.30$, and $\gamma = 1.50$, and the VAHQ penalty to $\beta_v = 3.00$. Table 2 reports the experiment-specific settings of K, ρ_0, α_0, $\sigma_{\max}$, and η. Here, K denotes the number of outer updates, $\sigma_{\max} = \sigma_0 = 1 - t_0$, and the flow levels are uniformly spaced as $t_k = t_0 + k(1 - t_0)/K$, $k = 0, \ldots, K$. When $\eta = 1$, Eq. (25) reduces to a full Gaussian noise refresh; for $\eta < 1$, partial cross-level noise continuity is retained. The parameters ρ_0, α_0, and η denote the initial ADMM penalty, the initial clean-endpoint anchoring strength, and the noise-refresh coefficient, respectively. Texture-filtering configurations are indexed by figure, and JPEG configurations by quality factor Q.

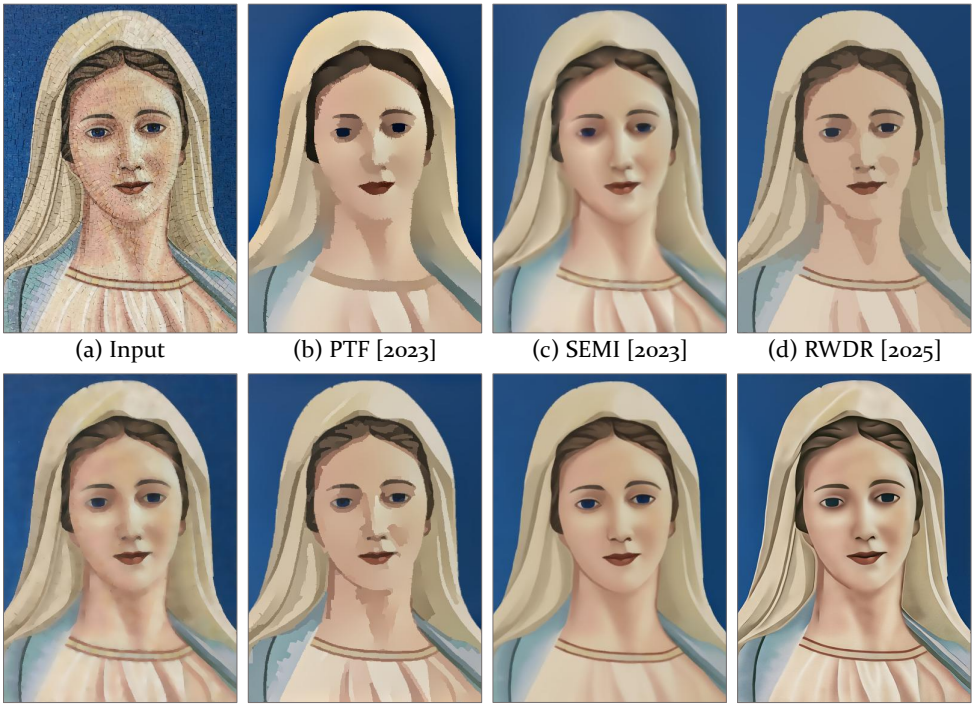
F Effect of Different Pre-trained Generative Models

We further study the choice of the pre-trained generative prior by replacing JiT while keeping the remaining components of Flow-Filter unchanged. We consider two latent-space generative models,



(a) Input (b) SD3 (c) FLUX.1 (d) ADM (e) Ours

Figure 12: Effect of different pre-trained generative models used as the plug-and-play prior in Flow-Filter. SD3 (Esser et al., 2024) and FLUX.1 (Black Forest Labs, 2024) operate in latent space, whereas ADM (Dhariwal & Nichol, 2021) and JiT (Li & He, 2026) operate directly in pixel space. Our default Flow-Filter uses JiT as the generative prior.



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 13: Additional qualitative texture-filtering results on a portrait with dense surface textures. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

Stable Diffusion 3 (SD3) (Esser et al., 2024) and FLUX.1 [dev] (Black Forest Labs, 2024), together with the pixel-space diffusion model ADM (Dhariwal & Nichol, 2021).

As shown in Figure 12, the latent-space priors SD3 and FLUX tend to alter the geometry of the input while suppressing texture. Such structural deviations are undesirable for image smoothing, where the filtered result is expected to remain closely aligned with the dominant structures of the original image. These results indicate that introducing a latent encoding–decoding stage can be less compatible with the pixel-domain decomposition used by Flow-Filter.

ADM performs diffusion directly in pixel space without a latent autoencoder (Dhariwal & Nichol, 2021), providing a more direct comparison with JiT in terms of representation domain. As shown in Figure 12, however, substantial high-frequency texture remains when ADM is used as the generative prior, whereas JiT produces a cleaner structure estimate with better consistency to the input. This comparison suggests that, within pixel-space generative priors, the Flow Matching prior used by JiT is more compatible with the iterative smoothing process than the diffusion prior considered here. Together with the results of SD3 and FLUX.1, these results support the choice of JiT among



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]



(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 14: Additional qualitative texture-filtering results on a mosaic still-life scene. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]



(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

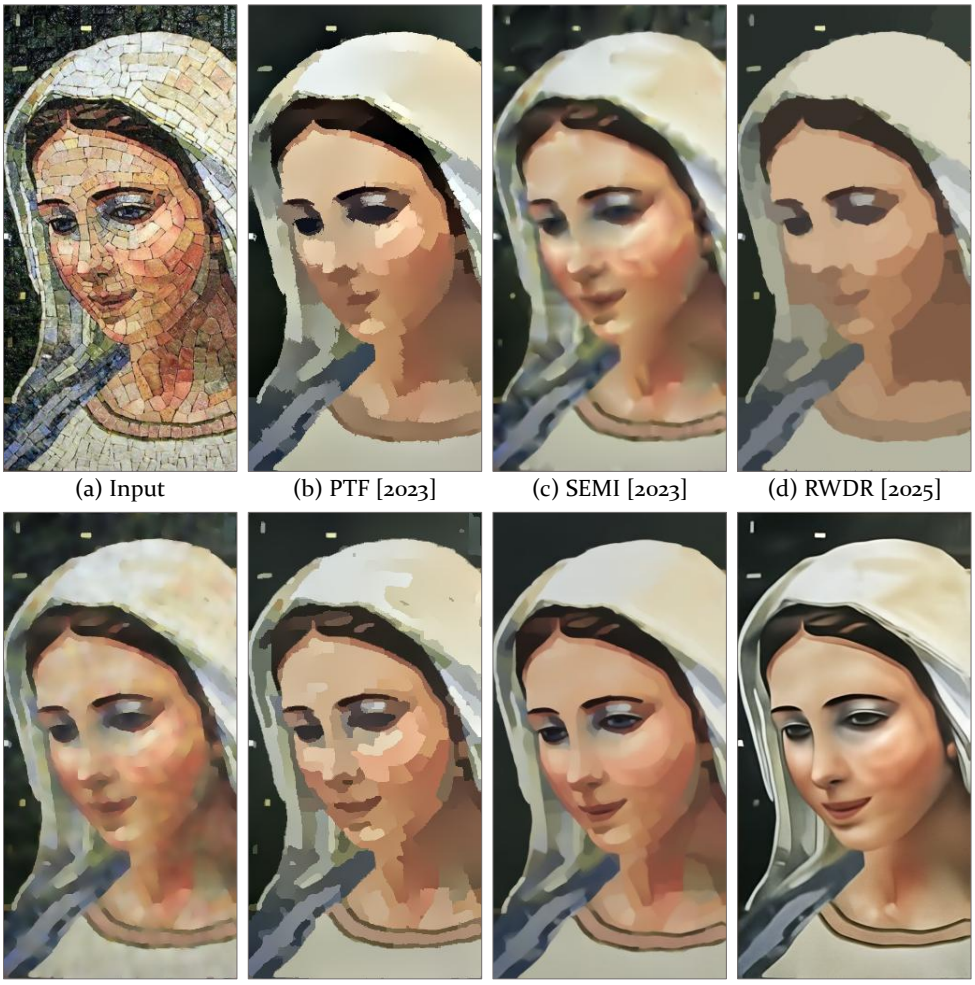
Figure 15: Additional qualitative texture-filtering results on structured text patterns. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

the pretrained models evaluated under the present implementation, but do not isolate the effects of representation space or training objective.

G Additional Comparisons

G.1 Additional Texture-Filtering Results

Section 5.1 presents the Dog and Stele examples as representative qualitative comparisons. Additional results are shown in Figures 13, 15, 14, and 16, covering mosaic portraits, lettering on a



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 16: Additional qualitative texture-filtering results on a portrait with strongly coupled texture and semantic content. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

textured surface, and a multi-object still-life scene. These examples highlight the challenge of suppressing dense surface textures while retaining fine structural contours and smooth tonal transitions.

In the portrait examples in Figures. 13 and 16, mosaic-tile boundaries extend across both facial features and smoothly shaded regions. APRF and SEMI soften facial details and fine folds, whereas DDGF, PTF, and RWDR produce coarse, piecewise-flat regions with abrupt tonal transitions. Flow-Filter suppresses the mosaic patterns while producing clearly delineated contours around the eyes, lips, and veil folds. It also maintains gradual shading across the face and clothing, yielding a cleaner structure layer without reducing these regions to large uniform color patches.

Figure 15 presents a different challenge: the embossed metal pattern overlaps the letter strokes. APRF and SEMI retain noticeable surface variations, while DDGF, PTF, and RWDR produce rough or blocky letter contours. GTF suppresses much of the background texture but softens the text boundaries. In contrast, Flow-Filter effectively suppresses the metal pattern while producing cleaner, more continuous stroke contours, keeping the lettering "TRAP DOOR" clearly legible.

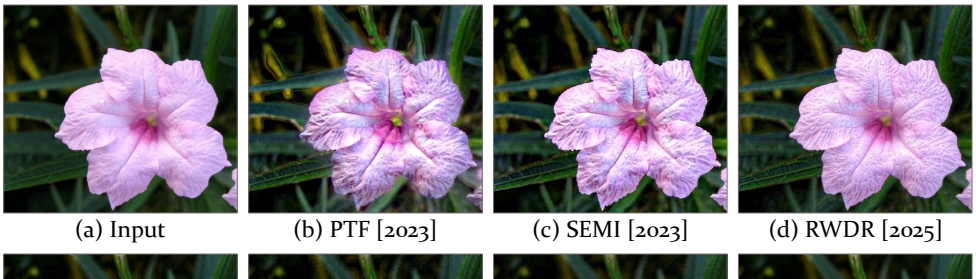
The still-life example in Figure 14 further examines texture filtering in the presence of adjacent objects and intricate boundaries. APRF and SEMI blur some fine contours, while DDGF, PTF, and RWDR simplify local structures into coarse color regions. Flow-Filter removes the dense mosaic



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 17: Detail enhancement comparison. Flow-Filter enhances fine petal textures while preserving smooth regions and clean boundaries. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 18: Additional detail enhancement comparison. Flow-Filter produces clearer petal details without noticeable boundary artifacts. The baselines are the same as in Figure 17.

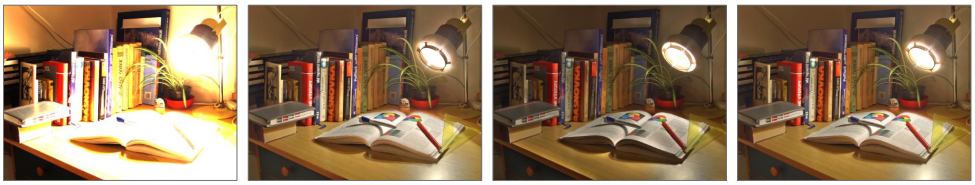
texture while keeping small features, such as the fish eye and crab legs, clearly distinguishable. The resulting image combines well-defined object boundaries with smooth shading within object regions, demonstrating a favorable balance between texture suppression and structural clarity.

G.2 Detail Enhancement

Given an input image I and its smoothed structure layer S, detail enhancement is performed as

$$I_{\text{enh}} = I + \beta(I - S),$$

with the same enhancement strength $\beta = 2.5$ for all methods.



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]



(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 19: Comparison of HDR tone-mapping results obtained using different image smoothing methods. Flow-Filter provides a favorable decomposition of large-scale illumination and local structures, yielding effective dynamic-range compression while preserving structural details and natural local contrast. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]



(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 20: Additional qualitative comparison of HDR tone-mapping results. The baselines are PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

In Figure 17, APRF and SEMI provide limited texture enhancement, DDGF, GTF, and RWDR exaggerate contrast near the flower center, and PTF introduces boundary artifacts. Flow-Filter enhances fine petal textures while preserving smooth regions and clean boundaries.

Figure 18 further shows clearer petal details with Flow-Filter, whereas competing methods either insufficiently enhance details or introduce boundary artifacts.

G.3 HDR Tone Mapping

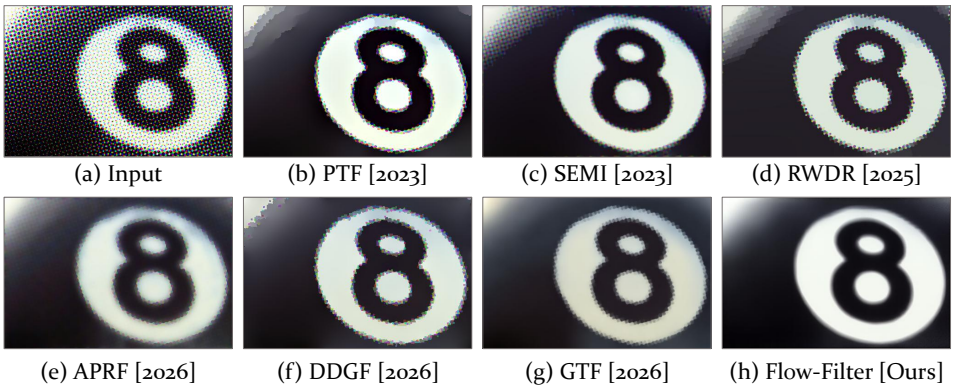
Edge-preserving image smoothing can be naturally applied to HDR tone mapping by decomposing the log-luminance image L into a base layer and a detail layer:

$$B = \mathcal{F}(L), \qquad D = L - B, \tag{82}$$

where $\mathcal{F}(\cdot)$ denotes the smoothing operator. The dynamic range is then compressed mainly in the base layer while retaining the detail component,

$$L_{\text{out}} = \kappa B + D, \qquad 0 < \kappa < 1, \tag{83}$$

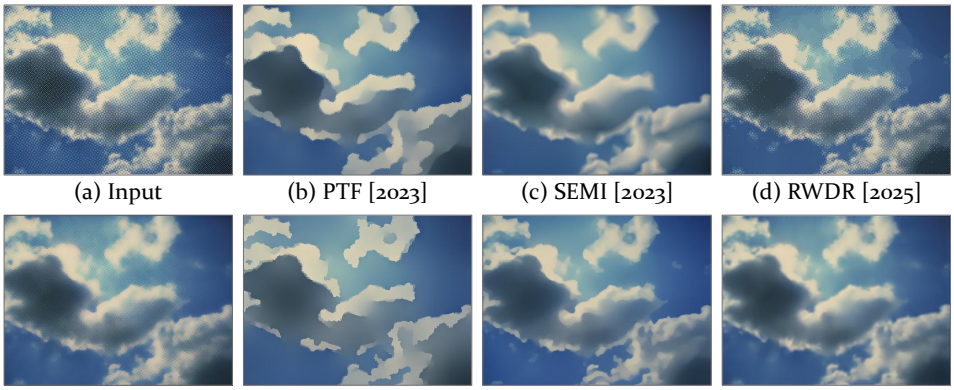
where κ controls the dynamic-range compression of the base layer.



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 21: Qualitative comparison of inverse halftoning on the digit "8" example. Our method more thoroughly suppresses halftone dot patterns while preserving smooth structural contours. Compared methods include PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).



(a) Input (b) PTF [2023] (c) SEMI [2023] (d) RWDR [2025]

(e) APRF [2026] (f) DDGF [2026] (g) GTF [2026] (h) Flow-Filter [Ours]

Figure 22: Additional qualitative comparison of inverse halftoning on a cloud scene. Our method effectively suppresses halftone patterns while preserving cloud contours and natural tonal transitions. Compared methods include PTF (Zhang et al., 2023), SEMI (Huang et al., 2023), RWDR (Qi et al., 2025), APRF (Yang et al., 2026), DDGF (Huang et al., 2026), and GTF (Zheng et al., 2026).

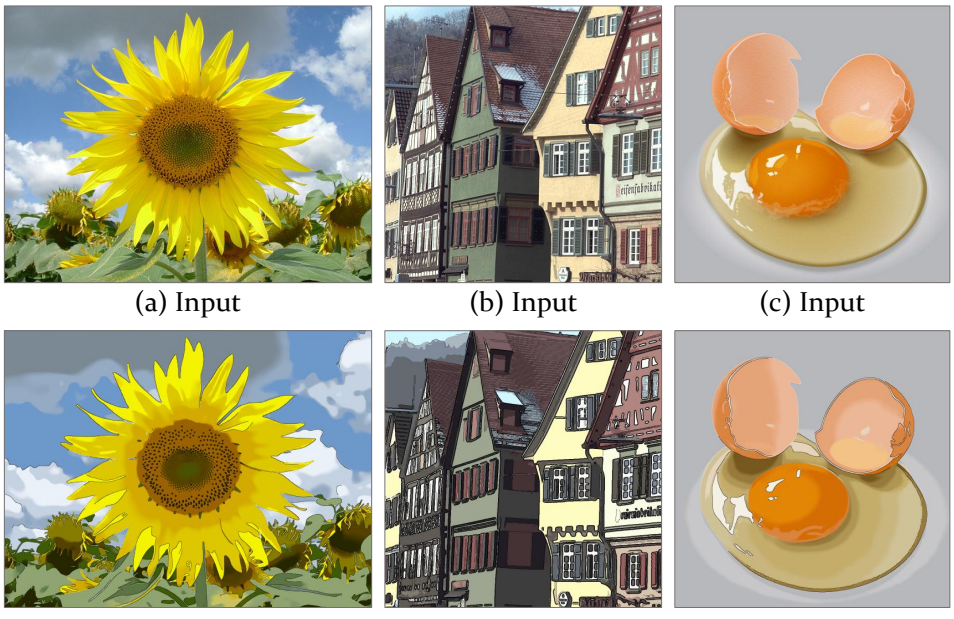
As shown in Figures 19 and 20, Flow-Filter achieves effective dynamic-range compression while producing more natural illumination and preserving clear local details. In the additional example, text on book pages and letter paper, as well as fine tabletop textures, remains clearly discernible. Flow-Filter also maintains well-defined object boundaries without noticeable edge artifacts.

G.4 Inverse Halftoning

Inverse halftoning aims to recover a continuous-tone image from a halftone input by suppressing dot patterns while preserving the underlying image structures. This task provides another application of structure-aware image smoothing, requiring effective halftone suppression without compromising structural fidelity.

As shown in Figure 21, competing methods leave visible halftone residues in smooth regions or exhibit irregularities along the digit boundaries. In contrast, Flow-Filter removes the dot patterns more thoroughly, producing the cleanest visual result while maintaining smooth contours of the digit "8" and the surrounding circular region.

Figure 22 presents a further comparison on a cloud scene. DDGF and PTF oversimplify cloud contours and introduce abrupt tonal transitions, whereas APRF and RWDR retain visible halftone patterns. Flow-Filter suppresses these patterns more effectively while preserving cloud shapes and smooth tonal transitions, yielding a more natural continuous-tone reconstruction.



(a) Input (b) Input (c) Input

(d) Ours (e) Ours (f) Ours

Figure 23: Image abstraction using Flow-Filter. Top: input images. Bottom: abstraction results. Fine-scale variations are reduced while prominent contours remain distinct, producing simplified, non-photorealistic depictions.

G.5 Image Abstraction

Image abstraction reduces visual complexity by suppressing fine-scale variations while retaining salient structures. Figure 23 presents three examples obtained using Flow-Filter. The results exhibit simplified color regions while keeping sunflower petals, architectural outlines, and the contours of the egg and plate clearly recognizable, yielding a stylized depiction of the original scenes.

G.6 Quantitative Evaluation on NKS

We evaluate Flow-Filter on the NKS benchmark, which consists of 200 synthetic images constructed by combining 20 structure images with 10 texture patterns. These patterns include wood, stone, leaf, grass, sand, ice, textile, grain, feather, and quartz, covering diverse appearances and spatial frequencies. Each textured input is paired with its corresponding structure image, enabling quantitative evaluation of texture suppression and structural fidelity.

We report mean PSNR and FSIM over all 200 images, with higher values indicating better performance. For our evaluation, FSIM uses the color variant (FSIMc). The parameter configuration is selected on NKS and applied unchanged to all 200 images.

As shown in Table 3, Flow-Filter achieves a PSNR of 35.3237 dB and an FSIM of 0.9680, obtaining the highest reported scores in both metrics among the compared methods. Its PSNR exceeds DDGF and APRF by approximately 0.05 and 0.07 dB, respectively. In terms of FSIM, Flow-Filter improves upon APRF (0.9664) and DDGF (0.9654) by 0.0016 and 0.0026, respectively, demonstrating competitive performance in both reconstruction accuracy and feature similarity.

H Runtime Analysis

We benchmark JiT-H/16 on three representative images using an NVIDIA GeForce RTX 5080, batch size one, and $K = 50$ outer updates (50 network evaluations). Network inference uses BF16, while VAHQ–FFT uses FP32. After two warm-up runs, we report the mean and sample standard deviation over five CUDA-synchronized runs. Timing includes all solver updates and final post-processing, but excludes model/input setup and image I/O.

Table 3: Quantitative results on NKS, averaged over 200 images. Higher PSNR and FSIM indicate better performance. FSIM denotes the color variant (FSIMc). The best and second-best results are highlighted.

Method	**PSNR** ↑	**FSIM** ↑	**Method**	**PSNR** ↑	**FSIM** ↑
L_0 (Xu et al., 2011)	28.3225	0.9122	SKF (Sun et al., 2020)	31.3293	0.8291
GIF (He et al., 2013)	31.4114	0.9033	E2H (Feng et al., 2022)	34.2430	0.9590
CAN (Chen et al., 2017)	32.0291	0.9062	GSF (Liu et al., 2022)	33.5760	0.9433
DGF (Wu et al., 2018)	30.9890	0.8609	L_1 (Yang et al., 2023)	33.6099	0.9414
SDF (Ham et al., 2018)	33.3696	0.9407	SSF (Huang et al., 2023)	31.0959	0.8837
RES (Zhu et al., 2019)	28.9339	0.9434	RWDR (Qi et al., 2025)	34.6735	0.9554
SWF (Yin et al., 2019)	32.0294	0.9075	APRF (Yang et al., 2026)	35.2504	0.9664
ELS (Liu et al., 2020a)	31.8634	0.8787	DDGF (Huang et al., 2026)	35.2725	0.9654
ILS (Liu et al., 2020b)	33.4706	0.9342	GTF (Zheng et al., 2026)	27.3714	0.9003
Flow-Filter (Ours)	PSNR: 35.3237		FSIM: 0.9680		

Table 4: Single-image solver runtime (mean ± sample standard deviation). VAHQ shares are measured relative to the total runtime of three separate profiling runs.

Image size ($W \times H$)	Total time (s)	VAHQ share
356×450	3.558 ± 0.042	15.8%
439×354	3.409 ± 0.043	19.3%
456×545	5.324 ± 0.024	11.2%

JiT prior updates account for 79.8–88.2% of the profiled runtime. All VAHQ–FFT subproblems meet the 10^{-3} relative-update tolerance within 22 iterations, averaging 5.00–6.42 iterations against a cap of 100. Thus, runtime is dominated by generative-prior inference, while explicit gradient refinement incurs a measurable cost without additional network evaluations.