Allocate Where It Matters: State-Adaptive Source Projection for Training-Free Image Editing

Anonymous authors
Paper under double-blind review

Abstract

Source-anchored image editors suppress unintended changes by projecting an editing trajectory toward a source reference. Existing methods mainly determine where preservation acts, but say little about how a limited correction budget should be distributed as the editing state evolves. We formulate source preservation as online fixed-budget correction allocation and introduce State-Adaptive Source Projection (ASP), a training-free mechanism that senses source-relative trajectory deviation, reallocates the available correction budget toward locations with greater current need, and feeds the corrected state back into the editing trajectory. Under matched editor loops, source references, supports, and realized budgets, ASP improves preservation by about 0.95 dB over Uniform Projection. Shuffling the same allocation values collapses to Uniform, reversing their state ordering degrades sharply, and freezing the first-step allocation also eliminates the gain, establishing the importance of spatial alignment, allocation direction, and continual state tracking. Integrated into the released SAM-Flow pipeline, ASP reduces Structure Distance by 18.8% and improves unedited-region PSNR by 0.70 dB on the 171 PIE-Bench++ cases with valid keep regions, while CLIP-based edit fidelity changes by at most 1.1%. On the FlowEdit Benchmark, background PSNR improves by 0.70 dB on $276/280$ cases and background LPIPS decreases on $279/280$ cases. ASP introduces no learned parameters, leaves peak memory unchanged, and adds only 0.61% editing-stage overhead.

1 Introduction

Text-guided image editing must satisfy a new instruction without needlessly regenerating the rest of the source image. Modern training-free methods steer pretrained diffusion or flow trajectories through inversion, internal-representation reuse, or direct source-conditioned transport (Mokady et al., 2023; Tumanyan et al., 2023; Kulikov et al., 2025). They avoid task-specific training, yet iterative steering can still alter identity, geometry, texture, or background (Kulikov et al., 2025; Avrahami et al., 2025; Wang et al., 2025). Source-anchored editors counter this drift by repeatedly injecting or projecting information from an aligned source trajectory (Zhu et al., 2025; Cui et al., 2026). A successful edit must therefore balance two competing requirements, changing the requested content while keeping unrelated content stable.

Most existing designs concentrate on where preservation should act. A user mask, attention map, cached feature set, or dynamic support separates editable from protected content. Once that support is selected, however, the correction inside it is commonly uniform or prescribed in advance. This leaves unresolved whether every preserved location should receive the same correction at every step. A support identifies where correction is allowed; it does not specify where the finite correction effort is currently most useful.

Trajectory deviation is spatially uneven and changes over time. Stable regions can remain close to the source without repeated intervention, while localized drift can emerge, move, or disappear as the edit proposal and earlier corrections interact. Uniform and static projection therefore operate open loop and cannot react when preservation needs shift. Under a fixed budget, they may spend correction on already stable locations while under-correcting active drift.

We formulate preservation as online correction allocation and introduce State-Adaptive Source Projection (ASP), a training-free inference-time allocation mechanism. At each step, ASP senses source-

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.

relative excess deviation on the current latent trajectory, converts it into a spatial demand field, and redistributes exactly the same total projection budget as a uniform baseline. The corrected state changes the next observation, closing the loop between state sensing, budget allocation, source projection, and subsequent state evolution. This interface is independent of how the support or edit proposal is obtained. Oracle masks enable controlled analysis, whereas an existing editor can supply its own dynamic support.

Under matched FLUX editor loops, source references, supports, random conditions, and realized per-step budgets, ASP consistently improves preservation over Uniform Projection. Shuffling the same allocation values across locations collapses to Uniform, reversing their state ordering causes a large degradation, and freezing the first-step allocation eliminates the gain. Location, direction, and continual updating therefore matter even when the total correction budget is fixed.

The allocation principle also transfers to a complete source-anchored editor. Using the released SAM-Flow pipeline while preserving its scout, dynamic support, temporal schedule, source trajectory, and per-step correction mass, ASP reduces Structure Distance from 7.291 to 5.920 and improves unedited-region PSNR from 31.687 to 32.383 on PIE-Bench++. The improvement is highly case-consistent, with P-PSNR improving on $162/171$ valid-mask cases. On the FlowEdit Benchmark, ASP improves background PSNR on $276/280$ cases and background LPIPS on $279/280$ cases. These gains require no learned parameters or additional peak memory and add only 0.61% editing-stage overhead.

Our contributions are threefold:

- We formulate source preservation in iterative image editing as an online fixed-budget correction-allocation problem, separating where correction may act from how a finite correction resource should be distributed as the trajectory evolves.
- We introduce ASP, a training-free closed-loop mechanism that senses source-relative trajectory deviation and redistributes source correction while exactly preserving the host editor's per-step correction mass.
- We provide matched-budget causal interventions, cross-backbone ordering evidence, and plug-in evaluation on both PIE-Bench++ and the FlowEdit Benchmark, showing that the gains depend on state-aligned spatial allocation and continual online adaptation.

2 Related Work

Training-free image editing. Modern training-free editing inherits its generative prior from diffusion models (Ho et al., 2020; Song et al., 2021; Rombach et al., 2022) and, increasingly, from flow matching and rectified flow (Lipman et al., 2023; Liu et al., 2023) implemented with scalable diffusion transformers (Peebles & Xie, 2023; Esser et al., 2024). Early approaches expose different control interfaces: SDEdit perturbs and denoises a user-guided image (Meng et al., 2022), Prompt-to-Prompt manipulates cross-attention (Hertz et al., 2023), and DiffEdit derives a semantic edit mask (Couairon et al., 2023). Real-image methods instead improve inversion or reuse internal representations, including Null-text Inversion (Mokady et al., 2023), Plug-and-Play features (Tumanyan et al., 2023), MasaCtrl (Cao et al., 2023), Pix2Pix-Zero (Parmar et al., 2023), and ReNoise (Garibi et al., 2024). Recent editors transport, bypass, or regularize source-conditioned trajectories (Kulikov et al., 2025; Avrahami et al., 2025; Wang et al., 2025; Rout et al., 2025; Kim et al., 2026; Yang & Ye, 2026; Han & Ni, 2026); VicoEdit instead introduces visual-context conditioning through posterior sampling (Song et al., 2026). These methods change the proposal, inversion, or conditioning mechanism. Given an edit proposal and aligned source trajectory, ASP controls how a fixed amount of source correction is allocated online.

Source preservation in diffusion and flow editing. Preservation mechanisms include attention or feature injection (Hertz et al., 2023; Tumanyan et al., 2023; Cao et al., 2023; Avrahami et al., 2025; Wang et al., 2025), posterior constraints (Tian et al., 2025), and explicit spatial anchoring. KV-Edit preserves background tokens through a DiT key–value cache within user-specified regions (Zhu et al., 2025), while DragFlow uses region supervision and hard background constraints for drag editing (Zhou et al., 2026). SAM-Flow localizes editable content from a scout pass and attention, then anchors the complementary region to a source trajectory using dynamic soft masks and temporal

accumulation (Cui et al., 2026). RegionE also distinguishes edited and unedited trajectories, but uses that distinction to reduce redundant computation (Chen et al., 2026). These methods primarily determine where preservation or computation acts. ASP keeps the host localization and temporal rules fixed and instead redistributes the realized per-step correction mass within that support.

Inference-time trajectory control. Several recent methods steer pretrained trajectories without parameter updates, including posterior sampling (Tian et al., 2025), stochastic optimal control for rectified dynamics (Rout et al., 2025), vector-field steering (Patel et al., 2025), and trajectory regularization (Kim et al., 2026). Proximal-Mean Inversion stabilizes rectified-flow inversion through gradient and velocity corrections (Wang et al., 2026), while reward-guided editing treats the complete generative path as a trajectory optimal-control problem (Chang et al., 2026). These methods intervene on trajectories, velocities, or rewards. ASP instead reallocates a fixed source-projection budget over space. We use Uniform, Shuffled, Inverse, Static, and Online controls to test where correction is placed, in which direction it is reallocated, and whether the allocation is updated over time.

3 Method

Figure 1 summarizes the transition from a generic source-anchored host editor to ASP. The host trajectories and projection interface are preserved, while ASP changes only how the realized correction budget is distributed within the host-provided support.

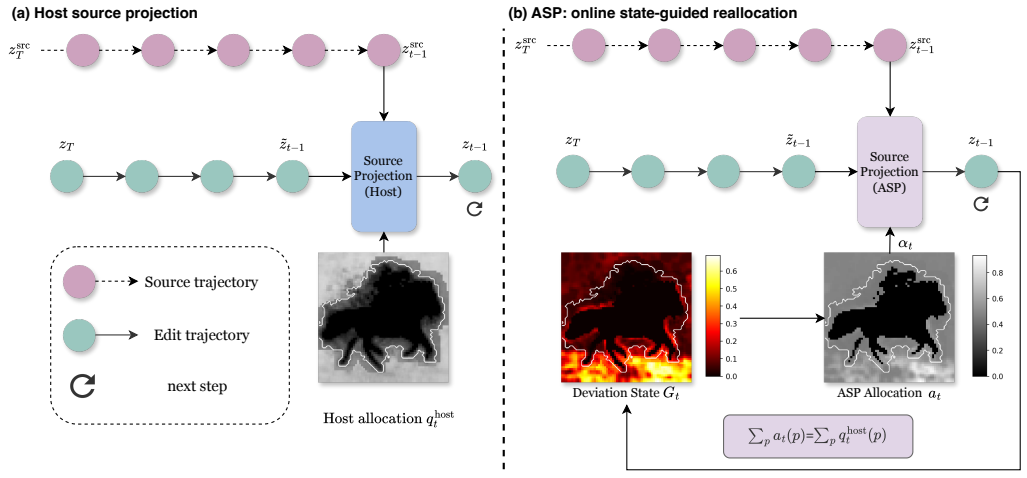

Figure 1: **Overview of State-Adaptive Source Projection (ASP).** (a) A source-anchored host editor evolves source and edit trajectories and performs source projection using the host allocation q_t^{host} to obtain the corrected state z_{t-1}. (b) ASP preserves the host trajectories and projection interface, but senses the current source-relative deviation G_t and redistributes the host correction budget into a state-adaptive allocation a_t. The total correction mass is exactly matched, $\sum_p a_t(p) = \sum_p q_t^{\text{host}}(p)$, and the corrected state is then fed into the next editing step.

3.1 Source-Anchored Editing as Correction Allocation

A source-anchored editor can be separated into three components: (i) localization, which supplies a preservation support; (ii) an edit proposal, which advances the target-conditioned trajectory; and (iii) correction allocation, which determines how much of the proposal is retained at each protected location. ASP changes only the third component. It does not depend on a particular localization or proposal module, so either component can be improved or replaced without changing the allocation rule.

Let $\mathbf{z}_t$ be the current latent editing state, $\mathbf{z}_t^{\text{src}}$ the aligned source trajectory, and $\mathcal{E}_t$ an arbitrary iterative source-anchored editor whose edit and source states share coordinates. After the editor proposes

$\widetilde{\mathbf{z}}_{t-1} = \mathcal{E}_t(\mathbf{z}_t)$, a source projection produces

$$\mathbf{z}_{t-1} = \mathbf{z}^{\text{src}}_{t-1} + (1 - \mathbf{a}_t) \odot \left(\widetilde{\mathbf{z}}_{t-1} - \mathbf{z}^{\text{src}}_{t-1}\right), \tag{1}$$

where $a_t(p) \in [0, 1]$ is the correction allocated to position p. A value of one restores the source state; zero keeps the edit proposal. Let $m_t(p) \in [0, 1]$ be a supplied preservation support and α_t the baseline mean projection strength. Uniform Projection uses $a_t(p) = \alpha_t m_t(p)$. We instead construct a state-adaptive allocation while matching its realized correction mass:

$$0 \le a_t(p) \le 1, \qquad \frac{\sum_p a_t(p)}{\sum_p m_t(p)} = \alpha_t, \qquad a_t(p) = 0 \text{ if } m_t(p) = 0. \tag{2}$$

Equation 2 makes the comparison budget-controlled: adaptivity may redistribute correction, but cannot increase its total amount. If $\sum_p m_t(p) = 0$, we set $a_t = 0$ and return the host proposal unchanged; below we assume positive support mass.

The budget must be matched at every step because projection changes the state from which all later edit proposals are computed. Matching only a trajectory-averaged budget would still allow a method to front-load or delay correction. We therefore impose Equation 2 independently at every step. The controlled comparison fixes the proposal mechanism, prompts, random conditions, source reference, support construction, and instantaneous correction mass. The edited trajectories may diverge after intervention because each method feeds back its own corrected state, but the allocation rule remains the only manipulated control variable.

3.2 Online Floor-Calibrated Deviation Sensing

A useful deviation signal must distinguish correctable editing drift from the source image's encode–decode floor. The raw edit–source difference contains both, and treating the latter as correctable drift would repeatedly spend budget on an unavoidable residual. Let $\widehat{\mathbf{x}}_{0,t}$ be the host editor's clean-latent estimate at step t, $\mathbf{x}^{\text{src}}$ the source latent, and $\mathbf{x}^{\text{floor}} = \mathcal{P}_{\text{VAE}}(\mathbf{x}^{\text{src}})$ one frozen VAE decode–encode round trip in the same normalized coordinates; $\mathbf{z}_t$ denotes the evolving noisy state. Writing $\langle\cdot\rangle_f$ for channel averaging and $\sigma(u) = (1 + \exp(-u))^{-1}$, we form

$$\begin{aligned} e_t(p) &= \text{LPF}\Big(\big\langle|\widehat{\mathbf{x}}_{0,t} - \mathbf{x}^{\text{src}}|\big\rangle_f\Big)(p), \\ b(p) &= \text{LPF}\Big(\big\langle\big|\mathbf{x}^{\text{floor}} - \mathbf{x}^{\text{src}}\big|\big\rangle_f\Big)(p), \\ d^{\text{exc}}_t(p) &= \text{ReLU}(e_t(p) - \kappa b(p)), \\ G_t(p) &= m_t(p)\,\sigma\left(\frac{d^{\text{exc}}_t(p) - \mu}{T_g}\right). \end{aligned} \tag{3}$$

Each operation removes a distinct confound: the clean estimate makes timesteps comparable, taking the magnitude before spatial filtering prevents opposite-signed residuals from cancelling, the patch-grid low-pass filter favors coherent drift over patch noise, and subtracting κb discounts the deviation already induced by source reconstruction. The smooth gate produces the current correction demand G_t without a discontinuous threshold. Because the signal is recomputed after projection, it describes the trajectory actually being controlled rather than a detached scout estimate. We use $\kappa = 1$ and $(\mu, T_g) = (0.05, 0.02)$ for FLUX. All operations remain in latent space; no intermediate decoding or additional backbone evaluation is required. Further implementation details, including cross-backbone scale calibration, are in Appendix A.1.

3.3 Budget-Constrained State-Adaptive Allocation

We convert sensed demand into correction while preserving the realized mean budget. Let $\bar{G}_t = \sum_p m_t(p)G_t(p) / \sum_p m_t(p)$. We first center the demand field so that positive and negative reallocations balance before saturation:

$$\widetilde{a}_t(p) = \alpha_t \left[1 + \lambda \left(G_t(p) - \bar{G}_t\right)\right]. \tag{4}$$

Clipping can otherwise change the realized mean. We therefore add a scalar bias β_t after centering and define

$$a_t(p) = m_t(p)\,\text{clip}(\widetilde{a}_t(p) + \beta_t, 0, 1), \tag{5}$$

Algorithm 1 State-Adaptive Source Projection (ASP)

Require: Edit state $\mathbf{z}_t$, source trajectory $\{\mathbf{z}_t^{\text{src}}\}$, host support m_t, host budget α_t, editor $\mathcal{E}_t$
Ensure: Corrected state $\mathbf{z}_{t-1}$
$\widetilde{\mathbf{z}}_{t-1} \leftarrow \mathcal{E}_t(\mathbf{z}_t)$ ▷ host edit proposal
Compute deviation state G_t using Equation 3
$\bar{G}_t \leftarrow \sum_p m_t(p) G_t(p) \,/\, \sum_p m_t(p)$
$\widetilde{a}_t(p) \leftarrow \alpha_t[1 + \lambda(G_t(p) - \bar{G}_t)]$
Find β_t by bisection s.t. $\sum_p m_t(p)\,\mathrm{clip}(\widetilde{a}_t(p) + \beta_t, 0, 1) = \alpha_t \sum_p m_t(p)$
$a_t(p) \leftarrow m_t(p)\,\mathrm{clip}(\widetilde{a}_t(p) + \beta_t, 0, 1)$
$\mathbf{z}_{t-1} \leftarrow \mathbf{z}_{t-1}^{\text{src}} + (1 - a_t) \odot (\widetilde{\mathbf{z}}_{t-1} - \mathbf{z}_{t-1}^{\text{src}})$
return $\mathbf{z}_{t-1}$

where a scalar bisection solve selects β_t so that Equation 2 holds after clipping. The clipped sum is continuous and monotone in β_t, so a feasible solution exists for every $\alpha_t \in [0, 1]$. This guarantees exact post-clipping budget matching rather than matching only the unclipped proposal. Thus ASP changes only where correction is spent. Setting $\lambda = 0$ numerically recovers Uniform Projection; $\lambda > 0$ prioritizes higher-demand states, and $\lambda < 0$ yields the Inverse intervention. The complete existence argument and solver details are in Appendix A.1.

The coefficient λ controls redistribution rather than total source strength. At $\lambda = 0$, all supported locations receive the baseline allocation. Increasing λ transfers budget from below-average to above-average demand; negative λ intentionally reverses that ordering. This nesting makes Uniform and Inverse exact algorithmic controls rather than separately tuned competitors.

3.4 Closed-Loop Source Projection

The allocation from Equation 5 is applied through Equation 1. Because the corrected state determines the next clean estimate and demand field, ASP repeatedly senses the current state, allocates the fixed budget, projects toward the source, and uses the corrected state at the next step. A Static intervention freezes the first-step allocation, whereas Online ASP recomputes G_t and a_t at every step. Shuffled permutes the same instantaneous allocation values across the full support; Inverse reverses their state ordering. These interventions isolate temporal feedback, spatial alignment, and direction without changing the prescribed budget.

Algorithm 1 introduces no latent-state optimization, decoder call, or additional denoiser/transformer evaluation.

3.5 Plug-in Instantiation and Complexity

For controlled mechanism isolation, FlowEdit supplies the candidate trajectory, the benchmark mask supplies the preservation support, and Uniform Projection supplies the per-step budget α. Every controlled variant shares prompts, random conditions, source trajectory, schedule, and realized correction mass. This is intentionally a generic Uniform Projection experiment, not a reimplementation of the complete SAM-Flow system.

To evaluate ASP inside a complete source-anchored editor, we use the released SAM-Flow pipeline. The editor first obtains a scout image and attention-derived dynamic soft support, then applies its core/ring/outer and temporal rules. We retain these components together with the released candidate and source-trajectory construction. If its projection is written as $\mathbf{z}_{t-1}^{\text{host}} = \mathbf{z}_{t-1}^{\text{src}} + \rho W_t^{\text{soft}} \odot (\widetilde{\mathbf{z}}_{t-1} - \mathbf{z}_{t-1}^{\text{src}})$, its effective correction field is

$$q_t^{\text{host}}(p) = \mathrm{clip}\big(1 - \rho W_t^{\text{soft}}(p), 0, 1\big)\,. \tag{6}$$

The plug-in defines the feasible support as $m_t(p) = \mathbf{1}\{q_t^{\text{host}}(p) > 0\}$ and sets the target budget to

$$\alpha_t^{\text{host}} = \frac{\sum_p q_t^{\text{host}}(p)}{\sum_p m_t(p)}. \tag{7}$$

Because $q_t^{\text{host}}(p) \in [0, 1]$ and is zero outside this support, $0 \le \sum_p q_t^{\text{host}}(p) \le \sum_p m_t(p)$ and therefore $\alpha_t^{\text{host}} \in [0, 1]$. The requested mass is consequently feasible under the box and support constraints at

every step. For this instantiation, we denote $q_t^{\text{SAM}} \equiv q_t^{\text{host}}$ and write the generic ASP allocation a_t as q_t^{ASP} when comparing the two correction fields. ASP then replaces only the within-support allocation, with bisection enforcing $\sum_p q_t^{\text{ASP}}(p) = \sum_p q_t^{\text{SAM}}(p)$ at that step. Consequently, SAM-Flow and the ASP instantiation use the same scout, attention maps, temporal schedule, transformer calls, and correction mass; ASP changes only its state-dependent placement. More generally, a source-anchored host can adopt the same allocation mechanism whenever it exposes an aligned source trajectory, a preservation support, and a projection field.

The additional computation is linear in the number of latent patches. Let P be the number of patches and K the fixed bisection count. Sensing, centering, and projection are $O(P)$; the scalar solve is $O(KP)$ with a small K and constant memory aside from patch-grid maps. ASP does not alter the denoiser, flow velocity, localization module, or edit direction and introduces no learned parameters. This analysis predicts negligible overhead when backbone evaluations dominate, consistent with the measured 0.61% released-pipeline editing-stage increase.

4 Experiments

4.1 Experimental Setup

We retain outputs for all 700 PIE-Bench++ pairs (Huang et al., 2024), which extend PIE-Bench (Ju et al., 2023): 540 modification (MOD), 80 deletion (GENdel), and 80 addition (GENadd) cases. Local preservation metrics require at least 1% keep area, an output-independent mask rule yielding 171 cases (127 MOD, 42 GENdel, and 2 GENadd); statistical claims use MOD and GENdel, while GENadd is descriptive. We report Structure Distance; unedited-region P-PSNR/LPIPS (Zhang et al., 2018)/MSE/SSIM; and whole-image/edited-region CLIP-W/E (Radford et al., 2021). Diagnostic editDmag measures masked mean absolute RGB change, and objDrop measures the CLIP-similarity decrease to the removed-object phrase. Mask accounting and metric details are in Appendix A.2; perceptual diagnostics are in Appendices A.10 and A.11.

Two complementary protocols separate mechanism isolation from complete-system validation. For controlled causal analysis, FLUX.1-dev is run in a shared FlowEdit ODE loop with identical prompts, source trajectories, oracle supports, random conditions, and schedules; Uniform Projection and ASP use exactly the same realized per-step correction budget, with default $\alpha = 0.1$ and $\lambda = 2$. For plug-in evaluation, we use the released SAM-Flow pipeline on PIE-Bench++ and the 280-case FlowEdit Benchmark. SAM-Flow and ASP share frozen token annotations, random conditions, scout and support construction, schedules, source trajectories, and per-step correction mass; ASP changes only its spatial allocation. Appendix A.3 documents annotation construction and freezing.

4.2 Fixed-Budget Controlled Evaluation

At the default $\alpha = 0.1$ budget, Table 1 shows that ASP improves matched-budget Uniform Projection by 0.97 dB on MOD and 0.85 dB on GENdel, winning $124/127$ and $40/42$ paired cases. Edit-strength diagnostics remain comparable; confidence intervals and paired tests are reported in Appendix A.6.

Table 1: Controlled full-set comparison on valid local masks. Uniform Projection and ASP share the editor, source-reference construction, oracle supports, and realized budgets. Bold marks the better preservation value within the matched $\alpha = 0.1$ comparison.

Method	α	MOD ($n = 127$)			GENdel ($n = 42$)		
		P-PSNR ↑	LPIPS ↓ $\times 10^2$	editDmag $\times 10^2$	P-PSNR ↑	LPIPS ↓ $\times 10^2$	objDrop ↑ $\times 10^2$
FlowEdit	0	19.90	15.24	9.78	20.04	19.99	2.75
Uniform Projection	0.1	28.98	3.76	8.00	29.61	3.99	1.16
ASP ($\lambda = 2$)	0.1	**29.95**	**3.11**	7.96	**30.47**	**3.33**	1.05
Uniform Projection	0.2	33.03	1.44	6.89	32.97	1.79	0.68

At $\alpha = 0.1$, the candidate editor, source trajectory, oracle support, random conditions, and realized correction mass are fixed, so the comparison isolates where correction is placed. The scaled edit-

strength measures remain close at this budget (MOD editDmag 8.00 versus 7.96; GENdel objDrop 1.16 versus 1.05), unlike the larger strength reduction produced by increasing Uniform Projection to $\alpha = 0.2$. That row is therefore shown only as a different preservation–editability operating point, not as a matched competitor.

4.3 Causal Analysis of State-Adaptive Allocation

Non-uniform allocation alone is insufficient. Shuffling preserves the same instantaneous values but destroys their correspondence to current demand, returning performance to Uniform. Reversing the ordering retains state dependence but causes a sharp degradation (Figure 2(a)). Both spatial alignment and allocation direction therefore matter. Freezing the first-step field likewise removes the gain even though that field was initially state-informed (Figure 2(b)), showing that the allocation must be recomputed as the trajectory evolves. Complementary numerical diagnostics are in Appendix A.9.

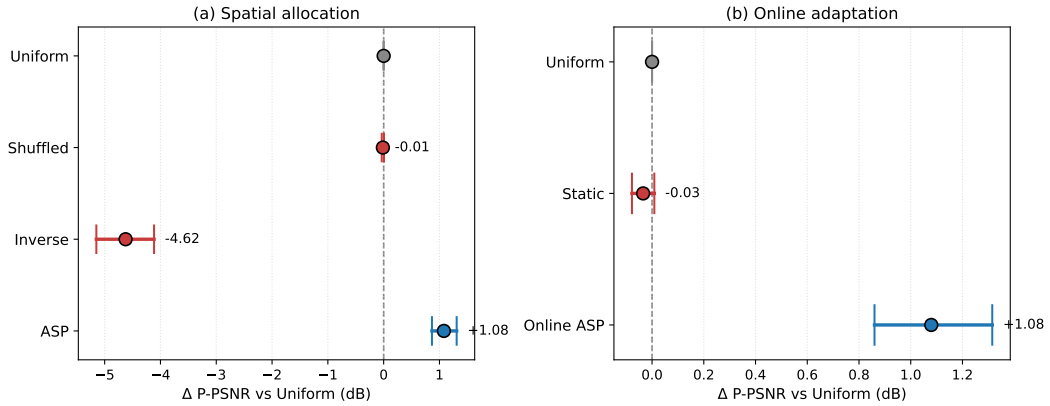

Figure 2: **Matched-budget causal controls** on the common 71-case subset (34 MOD, 35 GENdel, 2 GENadd). (a) Shuffled collapses to Uniform and Inverse degrades sharply. (b) Static collapses to Uniform, whereas Online ASP retains the gain. Error bars are paired 95% bootstrap confidence intervals; Table 1 uses the full MOD/GENdel pools.

4.4 SAM-Flow Plug-in Validation Across Benchmarks

To test whether fixed-budget allocation remains useful inside a complete editor, we replace only SAM-Flow's within-support allocation while retaining its scout, dynamic support, temporal rules, source trajectory, random conditions, and per-step correction mass. The same trend appears under both evaluation protocols (Table 2). PIE-Bench++ provides the unified cross-editor setting used later, while the FlowEdit Benchmark preserves the host-native setup. Similar PSNR gains on the two benchmarks, with improvements on nearly all paired cases, indicate that the effect is not tied to one protocol or a few outliers. Case-wise plots are in Appendix A.5.

Table 2: Host-controlled validation across benchmarks. Each row reports ASP relative to the released SAM-Flow pipeline under shared source trajectories, supports, schedules, and per-step correction mass. Percentage changes are reported for Structure Distance and CLIP fidelity; other entries are absolute differences. The final column reports the editDmag change once per benchmark.

Benchmark	Metric	Change	Improved cases	Δ editDmag
PIE-Bench++	P-PSNR ↑	$+0.697$ **dB**	$162/171$	-0.0038
PIE-Bench++	Structure Distance ↓	-18.8%	–	
PIE-Bench++	CLIP fidelity	$\leq 1.1\%$ change	–	
FlowEdit Benchmark	BG-PSNR ↑	$+0.702$ **dB**	$276/280$	$+0.0029$
FlowEdit Benchmark	BG-LPIPS ↓	-0.007	$279/280$	

The same budget yields different outcomes because ASP continually reshapes the host correction field as the sensed deviation evolves (Figure 3). The shift Δq_t remains inside the unchanged support and conserves correction mass, but moves source anchoring toward emerging local drift; the corresponding outputs show cleaner structure without a stronger global pull to the source. This mechanism is consistent with the matched-budget controls above. Further cases are in Appendix A.4, and the same causal ordering transfers to SD3 (Appendix A.13).

The two benchmarks also expose opposite changes in realized edit magnitude. On PIE-Bench++, editDmag decreases by 0.0038, whereas on the host-native FlowEdit Benchmark it increases by 0.0029 (95% CI $[0.0018, 0.0039]$) while target-text CLIP changes by only -0.0013. Preservation therefore improves under both weaker and stronger realized RGB modification, arguing against global edit attenuation as the explanation.

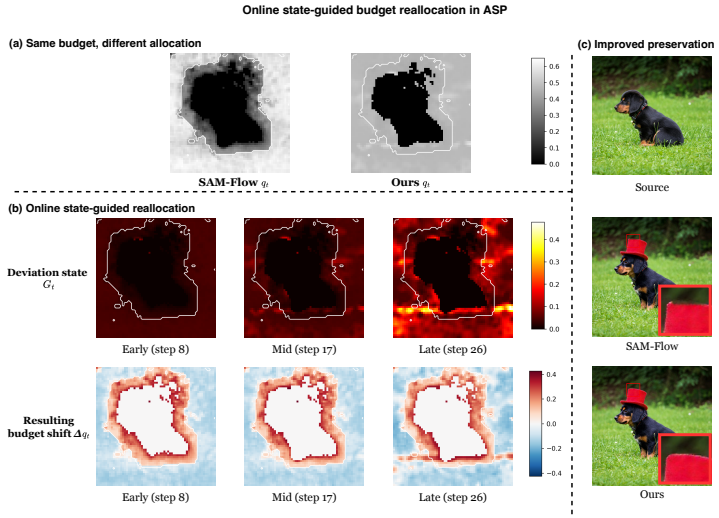

Figure 3: **Online state-guided budget reallocation.** (a) SAM-Flow and ASP distribute the same correction budget differently. (b) G_t drives an evolving shift $\Delta q_t = q_t^{\text{ASP}} - q_t^{\text{SAM}}$ within the shared support (white). (c) The resulting edit better preserves local structure.

4.5 Comparison with Existing Editors

Table 3 compares FlowEdit (Kulikov et al., 2025), KV-Edit (Zhu et al., 2025), RF-Solver-Edit (Wang et al., 2025), StableFlow (Avrahami et al., 2025), DirectEdit (Yang & Ye, 2026), and SAM-Flow (Cui et al., 2026). These editors occupy different preservation–editability points, so the matched SAM-Flow→ASP pair provides the cleanest attribution by sharing its proposal, support, schedule, trajectory, and correction mass. ASP improves every reported preservation metric while CLIP-W/E change by only $1.1\%/0.9\%$; cross-editor operating-point diagnostics are in Appendix A.10.

Table 3: Unified PIE-Bench++ comparison. Structure Distance and CLIP similarity use all 700 cases; pixel metrics use the 171 valid keep regions. LPIPS, MSE, and SSIM are rescaled for readability.

Method	Preservation					CLIP Similarity	
	Struct. Dist. ↓	P-PSNR ↑	LPIPS ↓ $\times 10^2$	MSE ↓ $\times 10^3$	SSIM ↑ $\times 10^2$	CLIP-W ↑	CLIP-E ↑
FlowEdit	33.263	22.728	15.10	8.35	92.60	25.994	25.096
KV-Edit	18.269	37.662	0.80	0.35	99.60	24.920	24.119
RF-Solver-Edit	4.459	34.207	3.10	0.69	99.20	21.848	20.967
StableFlow	22.785	23.480	10.50	7.94	93.20	24.053	23.226
DirectEdit	29.274	25.091	14.50	4.83	95.10	25.542	24.655
SAM-Flow	7.291	31.687	3.00	1.61	98.50	23.641	22.664
ASP (Ours)	5.920	32.383	2.60	1.30	98.70	23.386	22.450

Figure 4 shows that the gain reflects local preservation rather than failure to edit: across five cases, ASP retains source geometry and boundary structure while realizing the target semantics. Perceptual-quality and efficiency diagnostics are in Appendices A.11 and A.12.

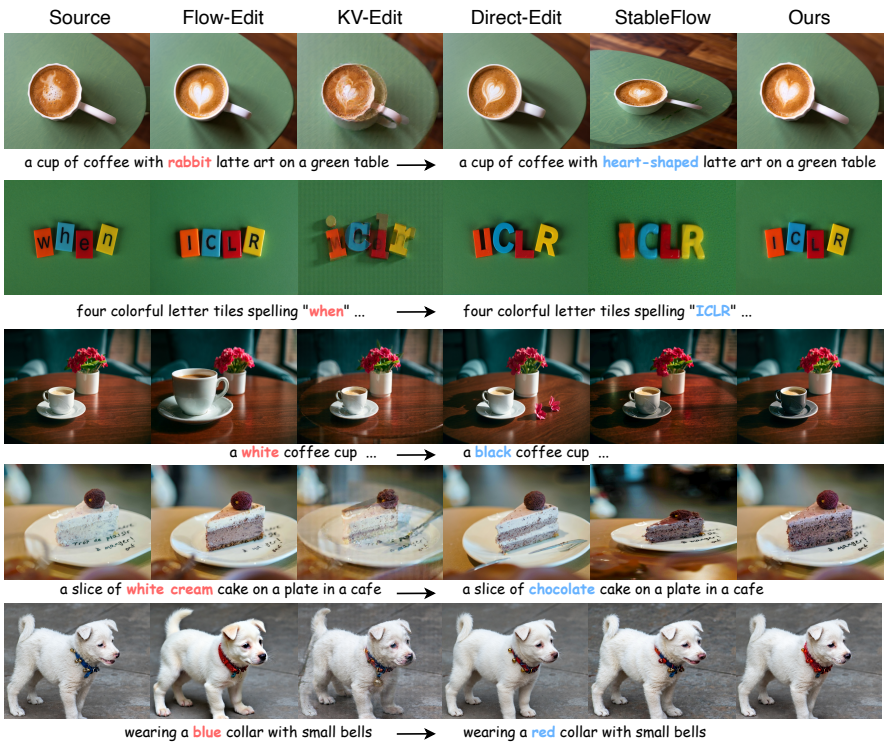

Figure 4: **Complete five-case qualitative comparison.** ASP preserves source geometry, background content, and local structure while realizing the requested target semantics.

5 Conclusion and Discussion

We introduced ASP, a training-free mechanism that reallocates a fixed source-projection budget from current trajectory deviation. Matched-budget controls show that aligned online allocation matters, while shuffled, reversed, or frozen variants remove or reverse the gain. In the released SAM-Flow pipeline, ASP improves preservation on two benchmarks with no learned parameters, unchanged peak memory, and 0.61% overhead; opposite changes in RGB edit magnitude rule out globally weaker editing. The results establish correction assignment as a control axis distinct from localization and total correction strength.

ASP assumes a usable proposal, aligned source trajectory, support, and host budget; recovering these inputs or selecting total budget remains outside its scope. Validation covers two rectified-flow backbones and valid-mask addition cases are scarce. Future work can jointly select and allocate correction, couple it with uncertainty-aware localization, and extend scale-normalized sensing to score-based diffusion trajectories.

Beyond its methodological implications, more reliable source-preserving editing may support creative design, localized retouching, and content adaptation by reducing unintended changes and subsequent manual correction. However, the same capability may also facilitate subtle image manipulation or deceptive media. Responsible deployment should therefore be accompanied by appropriate content-provenance, disclosure, and safety mechanisms.

Reproducibility Statement

An anonymized reference implementation of ASP is provided in the supplementary material. Experimental settings and protocol details are documented in Sections 3–4 and the Appendix.

AI Use Statement

The authors conducted and verified all experiments. Generative AI was used only to assist with literature retrieval and language polishing.

References

Omri Avrahami, Or Patashnik, Ohad Fried, Egor Nemchinov, Kfir Aberman, Dani Lischinski, and Daniel Cohen-Or. Stable Flow: Vital layers for training-free image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7877–7888, 2025.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22560–22570, 2023.

Jinho Chang, Jaemin Kim, and Jong Chul Ye. Training-free reward-guided image editing via trajectory optimal control. In *International Conference on Learning Representations*, 2026.

Pengtao Chen, Xianfang Zeng, Maosen Zhao, Mingzhu Shen, Peng Ye, Bangyin Xiang, Zhibo Wang, Wei Cheng, Gang Yu, and Tao Chen. RegionE: Adaptive region-aware generation for efficient image editing. In *International Conference on Learning Representations*, 2026.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. DiffEdit: Diffusion-based semantic image editing with mask guidance. In *International Conference on Learning Representations*, 2023.

Haowang Cui, Rui Chen, Tao Luo, Tao Guo, Zheng Qin, and Jiaze Wang. SAM-Flow: Source-anchored masked flow for training-free image editing. *arXiv preprint arXiv:2606.06228*, 2026.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 12606–12633, 2024.

Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. ReNoise: Real image inversion through iterative noising. In *European Conference on Computer Vision*, 2024.

Menglin Han and Zhangkai Ni. FlowBypass: Rectified flow trajectory bypass for training-free image editing. *arXiv preprint arXiv:2602.01805*, 2026.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-Prompt image editing with cross attention control. In *International Conference on Learning Representations*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Mingzhen Huang, Jialing Cai, Shan Jia, Vishnu Suresh Lokhande, and Siwei Lyu. ParallelEdits: Efficient multi-aspect text-driven image editing with attention grouping. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

Jeongsol Kim, Yeobin Hong, Jonghyun Park, and Jong Chul Ye. FlowAlign: Trajectory-regularized, inversion-free flow-based image editing. In *International Conference on Learning Representations*, 2026.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. FlowEdit: Inversion-free text-based editing using pre-trained flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19721–19730, October 2025.

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.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. NULL-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.

Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. *ACM Transactions on Graphics*, 42(4), 2023.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. 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.

Rui Song, Guo-Hua Wang, Qing-Guo Chen, Weihua Luo, Tongda Xu, Zhening Liu, Yan Wang, Zehong Lin, and Jun Zhang. Training-free image editing with visual context integration and concept alignment. *arXiv preprint arXiv:2604.04487*, 2026.

Feng Tian, Yixuan Li, Yichao Yan, Shanyan Guan, Yanhao Ge, and Xiaokang Yang. PostEdit: Posterior sampling for efficient zero-shot image editing. In *International Conference on Learning Representations*, 2025.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1921–1930, 2023.

Chenru Wang, Beier Zhu, and Chi Zhang. Free lunch for stabilizing rectified flow inversion. In *International Conference on Learning Representations*, 2026.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 64044–64058, 2025.

Desong Yang and Mang Ye. DirectEdit: Step-level accurate inversion for flow-based image editing. In *International Conference on Machine Learning*, 2026.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Zihan Zhou, Shilin Lu, Shuli Leng, Shaocong Zhang, Zhuming Lian, Xinlei Yu, and Adams Wai-Kin Kong. DragFlow: Unleashing DiT priors with region-based supervision for drag editing. In *International Conference on Learning Representations*, 2026.

Tianrui Zhu, Shiyi Zhang, Jiawei Shao, and Yansong Tang. KV-Edit: Training-free image editing for precise background preservation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16607–16617, October 2025.

A Additional Experimental Details

A.1 Method Rationale and Budget Solver

Deviation-signal rationale. Each term in Equation 3 removes a distinct confound. The clean estimate places observations from different timesteps in a comparable coordinate system; magnitude aggregation before low-pass filtering avoids cancellation across signed latent residuals; the low-pass operator favors coherent drift over patch-scale noise; subtracting κb discounts the source reconstruction floor; and the smooth gate avoids discontinuous changes when a location crosses the demand threshold. The sensed field is recomputed after projection, so it describes the trajectory actually being controlled rather than a detached scout or static image prior. In the released FLUX plug-in, feature magnitudes are averaged on the packed patch grid and filtered with $\sigma = 1$ patch; the corresponding controlled implementation operates on its native latent spatial grid with the resolution-matched $\sigma = 2$ filter.

For hard supports, $\bar{G}_t$ is the ordinary in-support mean. With a soft support, the first factor m_t attenuates sensing near uncertain boundaries and the second defines the support-weighted centering measure, deliberately giving those locations lower confidence in both operations.

All FLUX hyperparameters are fixed before full-benchmark evaluation. The SD3 calibration uses only latent deviation statistics from 12 calibration samples and does not access image-quality, preservation, or edit-fidelity scores.

For paired FLUX runs, the editing random seed is reset to 42 for every method and case; the released host uses reference-noise seed 999 for its scout construction. Shuffled allocations use the deterministic per-step seed $1234 + t$. The released annotation files are loaded in strict mode and identified by SHA-256 hashes recorded with each run.

Existence and implementation of the budget solve. Define $F_t(\beta) = \sum_p a_t(p;\beta) / \sum_p m_t(p)$. Because clipping is continuous and non-decreasing, F_t is continuous and monotone, with limits 0 and 1 as β tends to $-\infty$ and $+\infty$. Hence a solution exists for every feasible $\alpha_t \in [0, 1]$; when the allocation is not entirely saturated, it is unique up to flat clipped intervals. The extrema of $-\widetilde{a}_t$ and $1 - \widetilde{a}_t$ provide a finite bracket. The reference implementation constructs a guaranteed bracket from the current extrema, for example $[-\max_p \widetilde{a}_t(p) - 1,\ 1 - \min_p \widetilde{a}_t(p) + 1]$. The reported released-pipeline and controlled experiments use 40 and 50 iterations, respectively, on globally fixed brackets $[-1, 1]$ and $[-2, 2]$, whose endpoint masses were verified to bracket every reported target. This deterministic scalar solve introduces no learned per-image variable and guarantees the post-clipping realized budget to floating-point precision.

A.2 Mask Validity and Sample Accounting

PIE-Bench++ provides 700 pairs, and every method retains all 700 outputs for full-benchmark evaluation. Of the released masks, 529 cover at least 99% of the image and therefore leave less than 1% keep area for defining local preservation metrics. The remaining 171 masks are all eligible cases under this output-independent rule and consist of 127 MOD, 42 GENdel, and 2 GENadd cases. All P-PSNR comparisons use identical IDs within each pair of methods; zero-error values are capped at 100 dB. A border artifact in the released mask representation can create a one- to two-pixel frame; edit-region bounding-box evaluation removes the outer three pixels before cropping. Full-benchmark structure, edit-fidelity, and naturalness metrics use all available cases and do not require a valid keep mask.

Table 4: PIE-Bench++ mask accounting by edit category. "Local" counts cases whose released mask leaves at least 1% keep area; no method output is used in this eligibility decision.

Category	Total	Local	Category	Total	Local
Random	140	13	Change object	80	2
Add object	80	2	Delete object	80	42
Change background	80	59	Change style	80	0
Attribute content	40	20	Attribute pose	40	17
Attribute color	40	2	Attribute material	40	14
			All categories	700	171

The local-preservation conclusion is insensitive to the eligibility cutoff (Table 5). Thresholds from 0.5% to 5% change the eligible set by at most two cases, while the paired gain remains between 0.697 and 0.701 dB. The 0.5% and 1% rules select the same cases because no released mask falls between these two keep-area cutoffs.

Table 5: Robustness to the keep-region eligibility threshold on PIE-Bench++. Confidence intervals use paired bootstrap resampling of ASP-minus-SAM-Flow P-PSNR differences.

Minimum keep area	Cases	ΔP-PSNR ↑ (95% CI)	Improved
0.5%	171	$+0.697$ $[+0.597, +0.804]$	162/171
1%	171	$+0.697$ $[+0.598, +0.803]$	162/171
2%	170	$+0.700$ $[+0.601, +0.807]$	161/170
5%	169	$+0.701$ $[+0.599, +0.807]$	160/169

A.3 SAM-Flow Token Annotation Protocol

For reproducible localization, we construct and release fixed benchmark-specific token annotations while leaving all benchmark prompts unchanged. For the FlowEdit Benchmark, a deterministic rule proposes source-only and target-only content words together with a shared edited-subject noun; all 280 entries are reviewed and then frozen. For PIE-Bench++, the same semantic rule is applied deterministically to all 700 cases, with a source-anchor fallback when addition or material-transformation prompts would otherwise produce an empty source-token set.

Every paired SAM-Flow/ASP run loads the same frozen annotation file in strict mode, with no method-dependent fallback or regeneration. The runner records the file's SHA-256 hash and copies the annotations into the run directory, preserving the exact token inputs. Annotation construction is therefore fixed independently of the allocation method being evaluated.

A.4 Additional Local Preservation Comparisons

Figure A1 complements the trajectory visualization in Figure 3 with four additional editing cases. Both methods realize the requested edit; the crops isolate local differences in source-region continuity and texture.

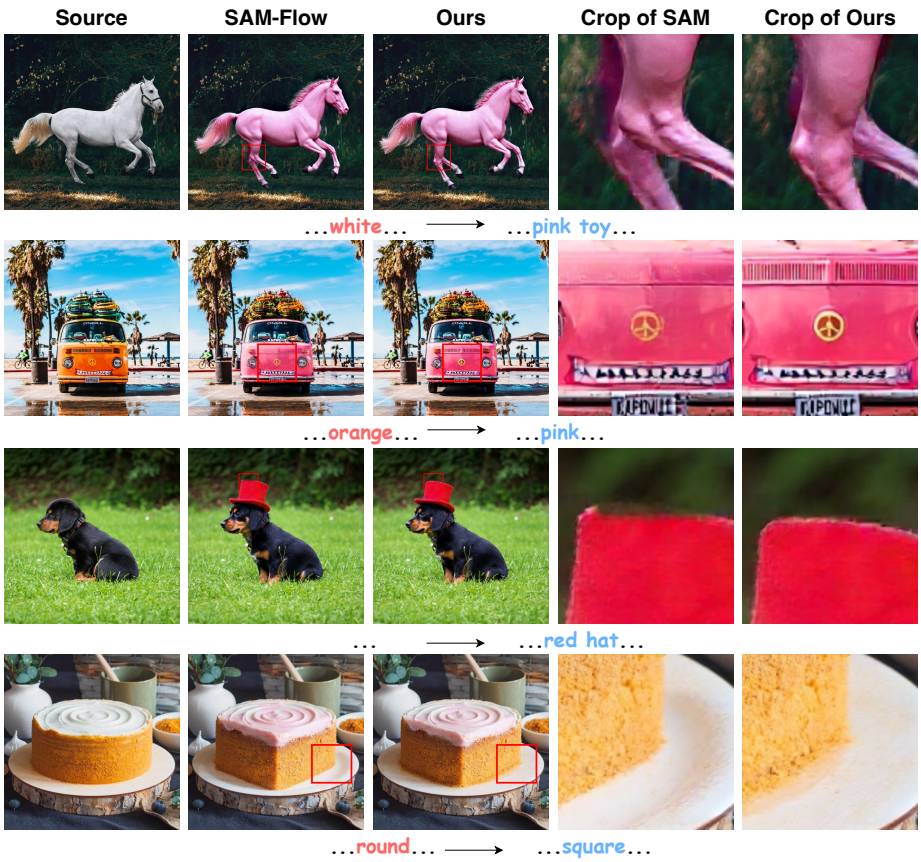

Figure A1: **Additional local preservation comparisons with SAM-Flow.** Both SAM-Flow and ASP realize the requested semantic edit. The enlarged regions highlight representative local differences in edge continuity, texture stability, and unintended source-region changes. ASP generally produces cleaner local transitions while preserving the intended edit.

A.5 Case-Wise Consistency

Figure A2 reports paired case-wise results for the released SAM-Flow plug-in evaluation.

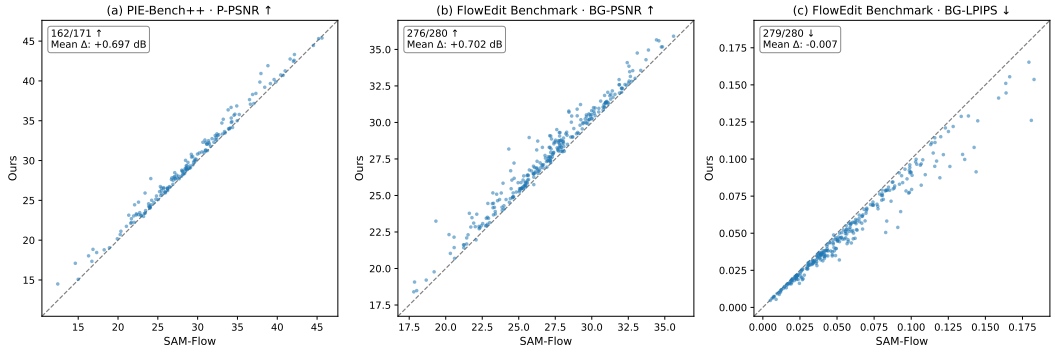

Figure A2: **Case-wise consistency of ASP preservation gains.** Each point compares SAM-Flow and ASP on the same case; the dashed diagonal denotes equal performance. ASP improves PIE-Bench++ P-PSNR on $162/171$ cases and FlowEdit-Benchmark background PSNR/LPIPS on $276/280$ and $279/280$ cases, respectively.

A.6 Controlled Statistical Details

At $\alpha = 0.1$, ASP improves P-PSNR over matched-budget Uniform Projection by 0.969 dB on MOD (95% CI $[0.81, 1.14]$, wins/ties/losses $124/0/3$) and 0.854 dB on GENdel (95% CI $[0.62, 1.12]$, $40/0/2$); both paired sign tests have $p < 10^{-4}$. Equivalence uses a paired-bootstrap TOST criterion whose 90% confidence interval must lie inside a pre-specified margin equal to 10% of the Uniform mean (± 0.0098 for MOD editDmag and ± 0.0027 for GENdel objDrop). MOD edit magnitude passes this test. GENdel objDrop changes by -0.0011 and is statistically non-significant, but its interval reaches the equivalence margin, so we claim comparability rather than strict equivalence. Confidence intervals use paired bootstrap resampling over common sample IDs.

A.7 Budget and Parameter Diagnostics

All controlled runs use $T = 28$ integration steps, $n_{\max} = 24$ as the top-sigma editing-window bound, and $n_{\text{avg}} = 1$ velocity sample per step. Table 6 shows the controlled λ sweep on cases with complete outputs for every swept value. Preservation improves monotonically while editCLIP and editDmag remain essentially constant. Across all evaluated variants, the realized mean allocation equals its target budget (0.1000 or 0.2000). Setting $\lambda = 0$ numerically recovers Uniform Projection: over the common 71 cases, the mean absolute P-PSNR difference is 0.0018 dB and the maximum is 0.0340 dB.

Table 6: Fixed-budget λ sweep on the common controlled subset with complete outputs for all swept values (34 MOD / 35 GENdel).

Method	α	MOD P-PSNR	MOD SSIM	MOD LPIPS	MOD editCLIP	MOD editDmag	GENdel P-PSNR	GENdel LPIPS	GENdel objDrop
Uniform	.1	31.23	.9474	.0344	.2540	.0949	29.31	.0410	.0129
ASP $\lambda = 1$	.1	32.08	.9506	.0301	.2538	.0945	29.84	.0376	.0131
ASP $\lambda = 2$	.1	32.50	.9523	.0279	.2539	.0943	30.15	.0347	.0121
ASP $\lambda = 3$	.1	32.78	.9534	.0263	.2537	.0940	30.33	.0336	.0115
Uniform	.2	35.62	.9652	.0130	.2513	.0832	32.71	.0188	.0080
ASP $\lambda = 3$	.2	36.12	.9666	.0113	.2515	.0823	32.87	.0181	.0067

A.8 Sensor Component Diagnostics

Table 7 separates the roles of floor calibration and spatial low-pass sensing under the controlled protocol. The components regulate the preservation–editability operating point rather than independently maximizing preservation. Removing floor calibration produces stronger realized edits but weaker preservation than full ASP. Removing the low-pass filter improves preservation further while reducing MOD edit magnitude and GENdel removal strength. Full ASP therefore retains more of the editing strength while still improving preservation substantially over Uniform Projection.

Table 7: Controlled component ablation using the source-relative deviation sensor in Equation 3 and exactly matched per-step correction mass. "w/o floor" sets $\kappa = 0$; "w/o LPF" replaces the low-pass operators in both the current-deviation and reconstruction-floor terms with the identity.

	MOD ($n = 127$)			GENdel ($n = 42$)		
Variant	P-PSNR ↑	LPIPS ↓ $\times 10^2$	editDmag $\times 10^2$	P-PSNR ↑	LPIPS ↓ $\times 10^2$	objDrop ↑ $\times 10^2$
Uniform Projection	28.980	3.755	7.996	29.612	3.988	1.164
ASP w/o floor calibration	29.639	3.221	8.235	30.358	3.372	1.417
ASP w/o LPF	30.769	2.571	7.870	31.401	2.720	0.985
ASP (full)	29.949	3.111	7.956	30.466	3.330	1.054

A.9 Controlled Causal Diagnostics

Table 8 lists the mean effects visualized in Figure 2. Each comparison preserves the realized per-step correction budget; paired bootstrap confidence intervals are shown by the error bars in the figure.

Table 8: Mean causal effects relative to Uniform Projection on the common 71-case controlled subset (34 MOD, 35 GENdel, and 2 GENadd). Spatial controls test allocation location and direction; temporal controls test continual state adaptation.

Intervention	Variant	ΔP-PSNR (dB)
Spatial allocation	Uniform	0.00
	Shuffled	−0.01
	Inverse	−4.62
	ASP	+1.08
Online adaptation	Uniform	0.00
	Static	−0.03
	Online ASP	+1.08

A.10 Preservation–Edit-Strength Diagnostics Across Editors

Table 9 retains the diagnostic operating-point comparison used in our mechanism analysis. Unlike the community-standard metrics in Table 3, editDmag quantifies pixel-level modification and objDrop isolates deletion strength. The table shows why high preservation numbers must be interpreted jointly with realized edit strength: for example, KV-Edit and RF-Solver-Edit operate at substantially lower MOD edit magnitudes, and RF-Solver-Edit has negative mean objDrop on deletion.

All compared methods have complete output sets. The reduced GENdel counts use the intersection for which a removed-object token and the required objDrop measurement are available for every method in the joint preservation–edit-strength diagnostic, yielding 37 valid-mask cases and 75 full-task cases.

Table 9: Preservation–edit-strength diagnostics across editors. Fidelity uses the strict common valid-mask sets (MOD $n = 127$; GENdel $n = 37$), whereas edit strength uses the common full-task pools (MOD $n = 540$; GENdel $n = 75$); these values are therefore not directly comparable to metrics computed on the 171-case set in Table 3.

Method	MOD P-PSNR ↑	GENdel P-PSNR ↑	MOD editDmag ↑	MOD editCLIP ↑	GENdel objDrop ↑
FlowEdit	19.90	20.20	0.088	**0.260**	**0.0184**
Uniform Projection ($\alpha = .1$)	28.98	29.66	0.084	0.259	0.0106
ASP	29.95	30.46	0.084	0.258	0.0102
StableFlow	20.93	20.07	0.083	0.239	0.0127
KV-Edit	**35.04**	**34.00**	0.053	0.248	0.0103
RF-Solver-Edit	31.69	30.30	0.022	0.215	−0.0047

A.11 Three-Layer Perceptual Evaluation

Table 10 summarizes paired ASP-minus-Uniform differences. DINO/P-DINO are source-consistency distances (lower is better), whereas CLIPIQA+ and MUSIQ are no-reference perceptual-quality scores (higher is better). ASP consistently improves source consistency while maintaining perceptual quality.

On the full MOD set, ASP and matched Uniform Projection have indistinguishable whole-image CLIPIQA+ ($\Delta = 0.0000$, $p = 1.0$) and MUSIQ ($\Delta = 0.0002$, $p = 0.29$), while ASP reduces DINO distance by 0.0012 ($p < 10^{-4}$). GENdel shows the same pattern: CLIPIQA+ and MUSIQ remain matched ($p = 0.88$ and 0.64), while DINO distance decreases by 0.0023 ($p = 0.0003$). Local edit-region quality is also maintained.

Table 10: Three-layer perceptual evaluation of ASP relative to matched-budget Uniform Projection. Fill-in metrics use 79 GENdel cases because one edit-region bounding-box crop is invalid after border removal. ΔDINO is not reported inside the intended deletion region, where source consistency is not a meaningful target.

Region	Group (n)	ΔDINO ↓	ΔCLIPIQA+ ↑	ΔMUSIQ ↑
Whole image	MOD (540)	-0.0012	$+0.0000$	$+0.0002$
Whole image	GENdel (80)	-0.0023	$+0.0012$	$+0.06$
Edit region	MOD (540)	-0.0012	$+0.0002$	$+0.004$
Fill-in region	GENdel (79)	–	-0.0004	$+0.06$

For deletion fill-in, absolute CLIPIQA+/MUSIQ values are $0.7186/66.85$ for ASP, $0.7190/66.79$ for Uniform Projection, $0.6351/63.10$ for KV-Edit, $0.6771/65.25$ for RF-Solver-Edit, and $0.6608/62.86$ for StableFlow. Keep-region P-DINO changes from 0.0070 to 0.0052 on MOD and from 0.0100 to 0.0063 on GENdel.

For the SAM-Flow plug-in, editing-strength diagnostics also rule out uniform attenuation: ASP slightly decreases editDmag on PIE-Bench++ ($\Delta = -0.0038$) but slightly increases it on the FlowEdit Benchmark ($\Delta \approx +0.0029$), while source preservation improves by approximately 0.70 dB in both settings.

A.12 Efficiency Details

Table 11 separates the controlled and released-pipeline protocols. Under the released SAM-Flow pipeline, ASP adds 1.07 s to a 175 s editing stage, does not change the 34.31 GB peak memory, and introduces no learned parameters.

Table 11: Runtime and memory overhead. Released-pipeline end-to-end time includes the 63.9 s scout stage.

Protocol	Baseline	+ASP	Overhead
Controlled	7.994 s	8.089 s	$+1.3\%$
Released, edit stage	175.05 ± 1.92 s	176.12 ± 1.40 s	$+0.61\%$
Released, end-to-end	238.9 s	239.9 s	$+0.45\%$
Peak memory	34.31 GB	34.31 GB	0

A.13 SD3 Cross-Backbone Results

Using an unsupervised 12-sample latent-statistics calibration, the 171-case SD3 evaluation reproduces ASP $>$ Uniform $\approx$ Shuffled $\gg$ Inverse ($+0.113$ vs. -0.662 dB). The variants use the same cases, but their forward noise is not strictly shared; this result therefore supports transfer of the ordering rather than a matched-randomness causal estimate.

Table 12: Calibrated SD3 evaluation on the 171 valid-mask cases. The reported 95% half-widths are obtained by bootstrapping case-wise differences relative to Uniform, but remain descriptive because forward noise is not strictly shared. Calibration uses latent statistics only and does not access evaluation scores.

Variant	P-PSNR	Δ vs. Uniform	CI half-width
Uniform	31.646	–	–
ASP	**31.759**	$+0.113$	0.044
Shuffled	31.629	-0.017	0.041
Inverse	30.984	-0.662	0.134