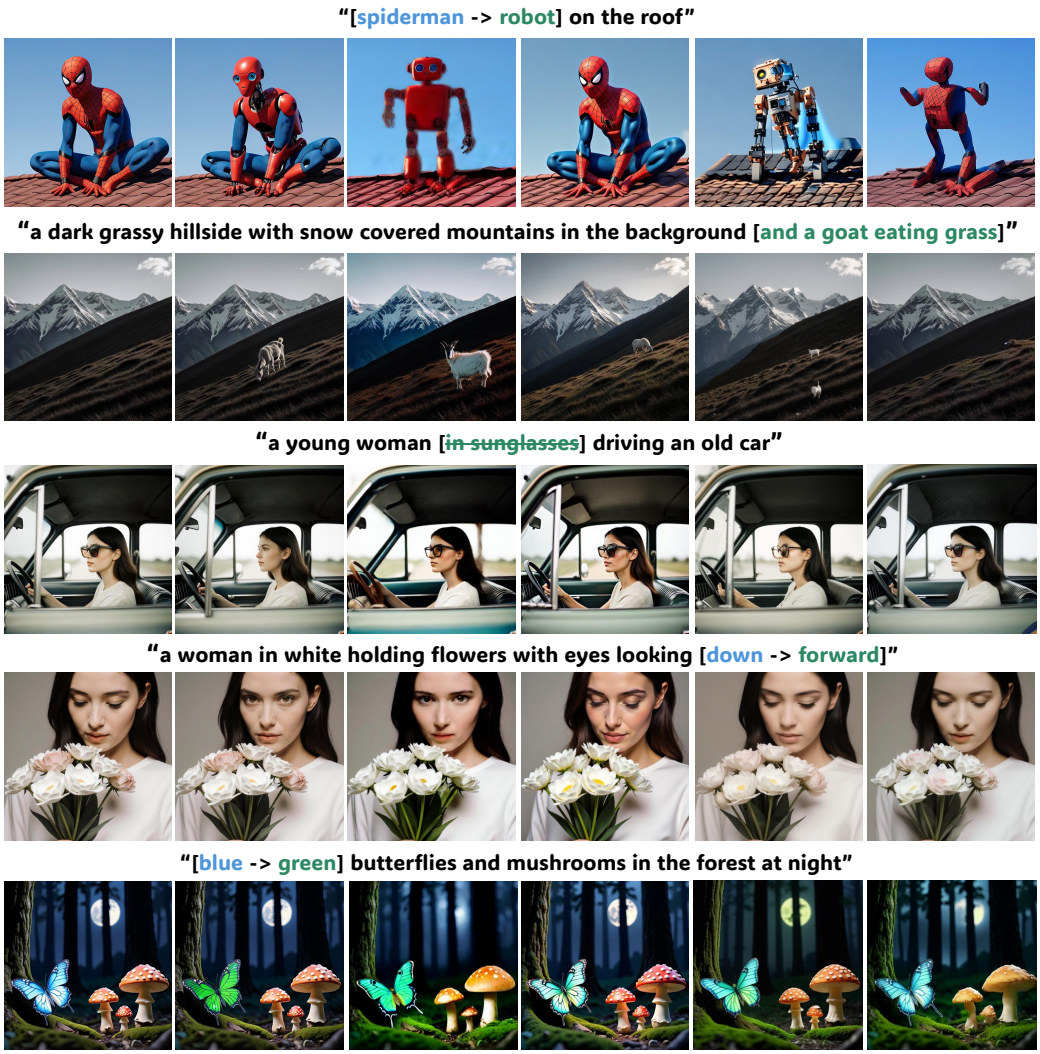
REFINEMENT IS INHERENTLY EDITABLE: TRAINING-FREE PROMPT-TO-PROMPT IMAGE EDITING WITH GENERATIVE REFINEMENT NETWORK

Anonymous authors
Paper under double-blind review



(a) Original (b) Ours (c) LEDits++ (d) FlowEdit (e) ChordEdit (f) MasaCtrl

Figure 1: Visual comparisons of text-guided image editing results from our method ***RefineEdit***, LEDits++ (Brack et al., 2024), FlowEdit (Kulikov et al., 2025), ChordEdit (Lu et al., 2026), and MasaCtrl (Cao et al., 2023). Blue and green indicate the original and editing prompts, respectively.

ABSTRACT

Text-guided image editing must introduce the requested changes while preserving unrelated source content. In training-free editing, diffusion editors often use spatial controls whose inaccuracies can leave edits incomplete or alter unrelated regions. Causal autoregressive editors face a further constraint: their fixed decoding order limits revision of earlier decisions. As the first to explore training-free

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.

image editing with Generative Refinement Network (GRN), we observe that its refinement process is inherently suitable for editing and offers a promising way to address these limitations. Motivated by this observation, we introduce ***RefineEdit***, a training-free prompt-to-prompt image editing framework built on the GRN. Our key idea is to couple edit localization with content generation through the global refinement of binary image codes, allowing editing evidence to be revised as the image evolves. More specifically, ***RefineEdit*** combines *bit routing* with two stabilization mechanisms: *adaptive spatial freezing* and *finite bit locking*. *Bit routing* starts from an intermediate source state and uses signed probability differences between the two branches to identify editable positions and bits. It directs selected bits toward editing refinement while anchoring the rest to the evolving source trajectory. *Adaptive spatial freezing* limits unnecessary expansion of the editing region, while *finite bit locking* maintains recent bit activations to support continued editing. The overall framework requires no additional training, external masks, or attention control. Across nine editing categories of PIE-Bench, ***RefineEdit*** achieves the best background-preservation scores in PSNR, LPIPS, MSE, and SSIM, together with the highest whole-image and edited-region CLIP scores among the evaluated methods. Code is available at https://anonymous.4open.science/r/RefineEdit.

1 INTRODUCTION

Text-to-image models can now generate high-quality images that closely follow text prompts (Rombach et al., 2022; Labs et al., 2025; Sun et al., 2024; Esser et al., 2024; Han et al., 2025). Yet visual creation is often iterative, and users may revise a generated image rather than start over. Text-guided image editing aims to make the changes specified by an editing prompt while preserving unrelated source content (Lu et al., 2026; Cao et al., 2023; Hertz et al., 2023; Kulikov et al., 2025; Brack et al., 2024). Balancing these goals is difficult: stronger edits may alter unrelated regions, while stronger preservation may weaken the intended change. The key challenge is to identify where the source conflicts with the editing prompt and make the required change while preserving other content.

Most training-free editors (Avrahami et al., 2025; Fu et al., 2025) address this challenge by adding spatial control to a pretrained generator. For example, diffusion methods use masks, attention maps, or injected features (Hertz et al., 2023; Avrahami et al., 2022; Cao et al., 2023; Brooks et al., 2023; Tumanyan et al., 2023), while autoregressive (AR) methods select between source and editing predictions during decoding (Wang et al., 2025). These controls determine which parts of the source may change, making their accuracy and evolution important to the editing quality: an overly narrow editing region may leave the requested change incomplete, whereas an overly broad region may alter unrelated content. Causal AR models make generation or editing decisions in early stages, which are difficult to revise (Esser et al., 2021; Sun et al., 2024; Kondratyuk et al., 2024; Tian et al., 2024; Wang et al., 2026). This raises a question: ***Can the evolving generation process itself provide the necessary evidence to revise the edit localization?***

The Generative Refinement Network (GRN) (Han et al., 2026) offers a viable setting for exploring this problem. As shown in Fig. 2, GRN repeatedly updates the full visual token map, moving from noise[1] to coarse structure and then fine details. Intermediate source states already contain useful spatial structure, allowing the editing branch to reuse this layout while revising object appearance. This resembles an artist developing the same rough feline sketch into a cat or a tiger. Beyond providing a reusable layout, GRN represents images with Hierarchical Binary Quantization (HBQ) codes whose spatial and channel coordinates align across branches. At the branching step, conditioning the shared state on the source and editing prompts yields different bit probabilities. Such differences can serve as the editing evidence at individual binary coordinates, suggesting a way to select editing updates while retaining source bits elsewhere. As refinement changes the edited state, subsequent predictions provide new evidence for bit selection. These selections guide the next content update, allowing edit localization and content generation to evolve together rather than remain separate processes. This leads to our central insight: ***Refinement is inherently editable.*** These properties make GRN a natural basis for editing through selective bit updates.

[1]Noise refers to a random binary code whose bits are independently sampled as 0 or 1 with equal probability.

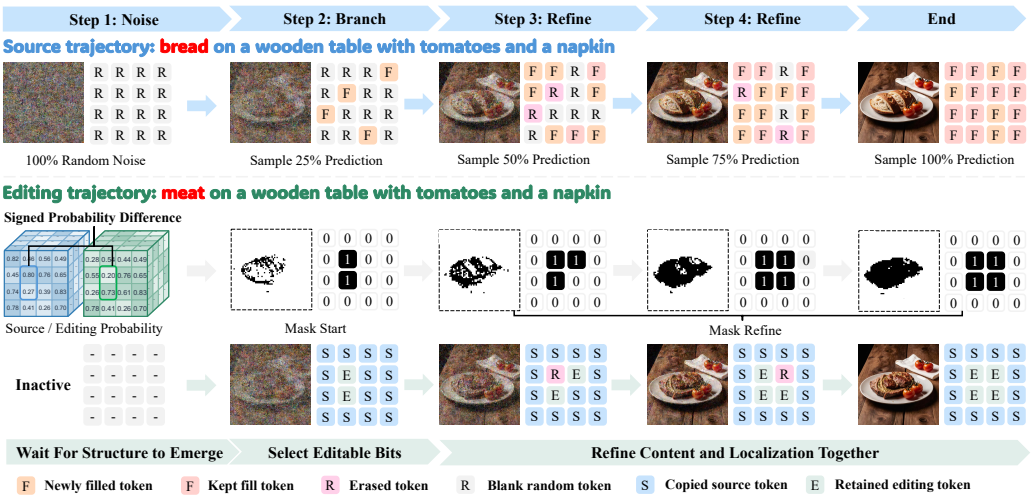


Figure 2: ***Top:*** GRN progressively develops image structure from a random binary code. ***Middle:*** Source and editing probabilities, illustrated by the blue and green cubes, are compared at the same sampled bit values. Black mask entries indicate editable positions. ***Bottom:*** Selected bits follow editing refinement, while unselected bits copy the evolving source state.

Motivated by the above observations, we propose ***RefineEdit***, a training-free framework for prompt-to-prompt image editing framework based on GRN. To the best of our knowledge, ***RefineEdit*** is the first method to adapt GRN to training-free image editing, extending its original applications in image generation and training-based video editing (Xie et al., 2026). Specifically, at the switch step of ***RefineEdit***, it initializes an editing branch from the current source state. Both the source and editing branches continue refinement with their corresponding prompts. From the switch step onward, we compare the probabilities assigned by the two branches to the same bits sampled from the source branch (source-sampled bits) at each refinement step. We use the resulting signed probability differences to construct a spatial mask that selects editable positions and a bitwise mask that determines which bits at those positions may change. *Source-anchored bit routing* accepts editing updates at selected coordinates and copies the evolving source state at all remaining coordinates. To stabilize these routing decisions across refinement steps, we therefore introduce *adaptive spatial freezing (AdaSF)*, which freezes spatial selection at the switch step when the initial editing response is sufficiently strong. Meanwhile, *finite bit locking (FBL)* retains editing permission for recently selected bits over a short window, so temporary probability fluctuations do not immediately interrupt their refinement. These two novel techniques in our method ***RefineEdit*** operate directly on binary codes, without external editing masks or attention control.

In our experiments, we compare ***RefineEdit*** with existing methods across nine editing categories of PIE-Bench (Ju et al., 2024). ***RefineEdit*** achieves the best background-preservation scores in PSNR, LPIPS, MSE, and SSIM, together with the highest whole-image and edited-region CLIP scores among the compared methods. To summarize, our contributions are as follows:

- Based on our observation that refinement is inherently editable, we introduce ***RefineEdit***, the first training-free prompt-to-prompt image editing framework built on GRN. RefineEdit couples edit localization with content generation, allowing editing evidence to evolve with the visual state.
- We develop source-anchored bit routing, which uses signed probability differences to select editable positions and bits while anchoring the remaining bits to the evolving source. *AdaSF* limits unnecessary expansion of the editing region, and *FBL* maintains editing permission for recently selected bits to support continued refinement.
- Experiments across nine editing categories of PIE-Bench demonstrate that ***RefineEdit*** effectively follows editing prompts while preserving unrelated source content. Ablation studies further show the complementary roles of AdaSF and FBL in stabilizing edit localization and supporting bitwise editing.

2 Related Work

2.1 Text-to-Image Generation

Text-to-image generation has evolved through several generative paradigms. Diffusion and flow models have established strong image fidelity and text alignment (Peebles & Xie, 2023; Sauer et al., 2024; Podell et al., 2024; Rombach et al., 2022; Esser et al., 2024; Ma et al., 2024). Recent autoregressive models show that, with improved visual tokenizers and model scaling, discrete next-token prediction can achieve competitive or even superior generation quality (Ramesh et al., 2021; Yu et al., 2022; Sun et al., 2024). However, their causal order prevents earlier tokens from being revised once generated. Visual autoregressive models reorganize generation as next-scale or bitwise prediction (Tian et al., 2024; Han et al., 2025), improving efficiency and discrete modeling but still committing to a fixed coarse-to-fine order. GRN (Han et al., 2026) overcomes this restriction through near-lossless Hierarchical Binary Quantization and global random refinement, allowing every binary coordinate to be repeatedly reconsidered. Its globally revisable binary representation provides a natural interface for image editing, enabling changes to be localized and controlled at the bit level.

2.2 Training-free image editing

Training-free image editing repurposes pretrained generators without parameter updates. Diffusion approaches commonly use mask blending, attention control, or intermediate-feature injection to introduce target semantics while retaining source structure (Hertz et al., 2023; Avrahami et al., 2022; Cao et al., 2023; Brooks et al., 2023; Tumanyan et al., 2023; Li et al., 2026; Brack et al., 2024; Lu et al., 2026; Wu et al., 2026). However, such continuous control often trades edit strength for background fidelity. Flow methods improve semantic transport and efficiency by constructing paths between source and target distributions (Kulikov et al., 2025; Rout et al., 2025), yet editable support remains implicit rather than enforced at native coordinates. Discrete autoregressive editing instead exploits token-distribution differences to localize changes (Wang et al., 2025; Hu et al., 2025), but its control follows a fixed causal order and does not directly match GRN's globally revisable generation (Xie et al., 2026). In this paper, our proposed ***RefineEdit*** uses aligned-trajectory contrast to route only prompt-sensitive bits to editing and anchor the rest to the evolving source.

3 Preliminaries

Binary image representation. A frozen Hierarchical Binary Quantization (HBQ) tokenizer $\mathcal{E}_{\mathrm{HBQ}}$ encodes an image I as $Y = \mathcal{E}_{\mathrm{HBQ}}(I) \in \{0,1\}^{N\times D}$, where N is the number of spatial positions and D is the number of bits per position. Each coordinate (n, d) identifies one bit at spatial position n.

Generative refinement. GRN initializes $Y_0 = Z$, where each bit of Z is sampled once as 0 or 1 with equal probability. The same random code is reused throughout refinement. Given the current binary input Y_t and text condition c, GRN predicts all binary coordinates in parallel:

$$P_t = \mathrm{softmax}(G_\theta(Y_t, c, t)), \qquad \widehat{Y}_t \sim \mathrm{Cat}(P_t). \tag{1}$$

Here, G_θ is the frozen GRN Transformer, which outputs two logits per binary coordinate. Softmax converts these logits into the distribution $P_t(n, d, \cdot)$ over $\{0, 1\}$. Cat samples each bit of $\widehat{Y}_t$ from its corresponding distribution.

The next state combines sampled predictions with the fixed random code:

$$Y_{t+1} = S_t \odot \widehat{Y}_t + (1 - S_t) \odot Z, \tag{2}$$

where $\odot$ denotes the element-wise multiplication operator. The selection mask S_t satisfies $S_{t,n,d} \sim \mathrm{Bernoulli}(\lambda_{t+1})$: it equals 1 with probability λ_{t+1} and 0 otherwise. Thus, λ_{t+1} controls the expected proportion of predictions retained in the next state. As this proportion increases, refinement moves from random bits toward a complete image. The selection mask S_t is resampled at every step, so a coordinate may return to its initial random value. All coordinates are predicted again at the next step rather than permanently fixed.

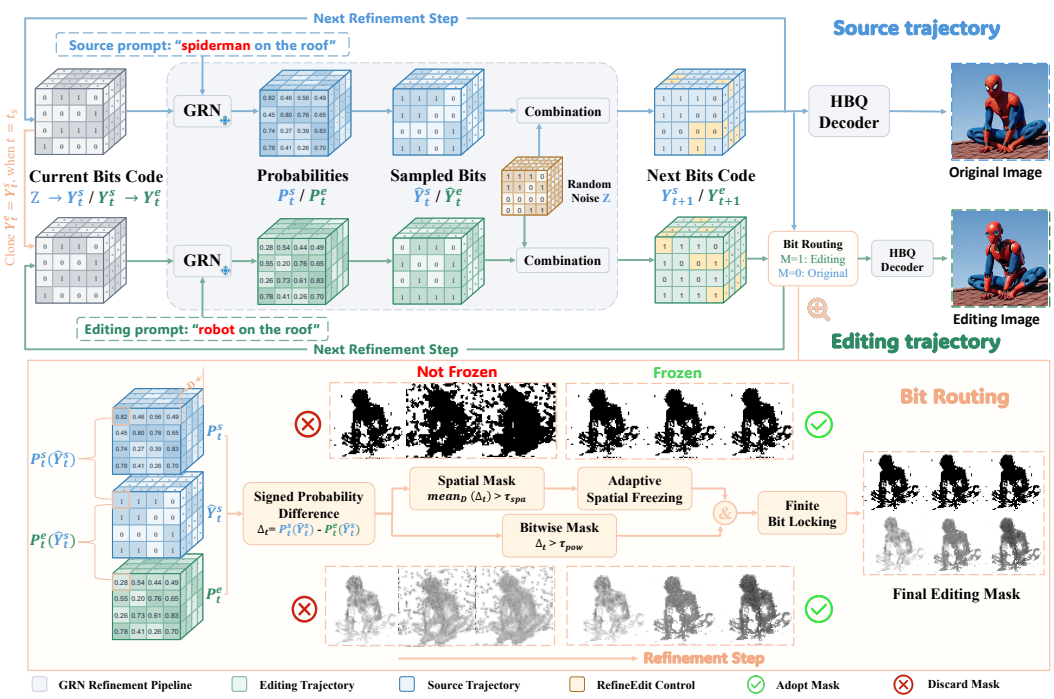


Figure 3: Overview of ***RefineEdit***. The editing branch inherits the source state at the switch step. Both branches retain GRN's global prediction and random refinement, while bit routing selects editing updates or copies the evolving source state.

4 METHOD

In this section, we present ***RefineEdit***, a training-free prompt-to-prompt image editing framework built on GRN. Section 4.1 describes how we derive editing masks from aligned bit probabilities and use them to route source and editing updates. Section 4.2 introduces spatial and temporal stabilization to limit mask expansion and sustain bitwise editing. Figure 3 provides an overview.

4.1 REFINEMENT-GUIDED BIT ROUTING

Given a source prompt c^s and an editing prompt c^e, RefineEdit first runs source refinement under c^s. At the switch step t_s, we initialize the editing branch from the current source state, $Y^e_{t_s} = Y^s_{t_s}$. Both branches then continue refinement under their corresponding prompts, using the same fixed random code Z and refinement schedule. The editing branch therefore inherits the intermediate structure from the source branch. We use differences between the two branches' bit probabilities as editing evidence to identify coordinates that may depart from the source.

Editing evidence. At each editing step, the two branches produce P^s_t and P^e_t with Eq. 1. We denote the source-sampled bit at coordinate (n, d) as $\widehat{y}_{t,n,d} = \widehat{Y}^s_{t,n,d}$. Its signed probability difference is:

$$\Delta_{t,n,d} = P^s_t(n, d, \widehat{y}_{t,n,d}) - P^e_t(n, d, \widehat{y}_{t,n,d}). \tag{3}$$

A positive value means that the editing branch assigns less probability to the source-sampled bit, providing evidence for allowing that coordinate to depart from the source. The branches share the same input only at the switch step. During the following refinement steps, the probability differences reflect both the respective prompts and the evolving branch states. We use these differences to determine where editing is allowed and which bits may change.

Spatial and bitwise mask selection. We express the two decisions described above through instantaneous spatial and bitwise masks. Averaging the evidence across all D bits at each position gives the spatial score $q_{t,n} = D^{-1} \sum_{d=1}^{D} \Delta_{t,n,d}$. The spatial mask $a_t \in \{0, 1\}^N$ and bitwise mask

$m_t \in \{0,1\}^{N\times D}$ are then defined as:

$$a_{t,n} = \mathbf{1}[q_{t,n} > \tau_{\text{spa}}], \qquad m_{t,n,d} = a_{t,n}\,\mathbf{1}[\Delta_{t,n,d} > \tau_{\text{pow}}], \tag{4}$$

where $\mathbf{1}[\cdot]$ equals 1 when its condition holds and 0 otherwise. The spatial threshold selects positions, while the power threshold determines which bits within those positions receive editing permission. For a fixed spatial mask and fixed probabilities, lowering τ_{pow} relaxes bit selection. Changing these selections alters subsequent branch states and predictions, so bitwise control can also influence later spatial masks. These masks specify editing permissions, which we apply through source-anchored routing.

Source-anchored routing. Standard GRN refinement in Eq. 2 produces the next source state Y^s_{t+1} and an editing proposal $\widetilde{Y}^e_{t+1}$. We then route their bits using m_t:

$$Y^e_{t+1} = m_t \odot \widetilde{Y}^e_{t+1} + (1 - m_t) \odot Y^s_{t+1}. \tag{5}$$

Selected coordinates take values from the editing proposal; all other coordinates copy the source state at the same refinement step. Both branches therefore retain global GRN refinement; the mask controls which updates enter the next editing state. At the final step, we route the sampled source and editing predictions and decode the resulting HBQ code, without mixing in Z.

Eq. 4 selects masks from the current predictions. As the predictions evolve, the spatial mask may expand into unrelated regions, and previously selected bits may lose editing permission. Section 4.2 addresses these issues by two novel techniques before the masks are used in routing.

4.2 Mask stabilization

Adaptive spatial freezing (AdaSF). As mentioned above, mask expansion can introduce changes in unrelated regions. To address this problem, we adapt the freezing decision to each sample using its initial editing response. This response determines whether to freeze the spatial mask at the switch step or continue updating it. It is formulated as:

$$\Omega = \{n \mid a_{t_s,n} = 1\}, \qquad r = \frac{\sum_{n\in\Omega}(q_{t_s,n} - \tau_{\text{spa}})}{\max(1, |\Omega|)}, \tag{6}$$

Here, Ω contains the positions selected at the switch step, and $|\Omega|$ is their number. The response r measures their average score margin above τ_{spa}, with $r = 0$ when no position is selected. When $r \geq \tau_{\text{frz}}$, we retain the initial spatial mask by setting $a_t = a_{t_s}$ for all $t \geq t_s$. Otherwise, a_t continues to follow Eq. 4. We use $\tau_{\text{frz}} = 2\tau_{\text{spa}}$ by default.

The decision is made once at the switch step and affects only spatial selection; bitwise selection and content refinement continue. Even within a frozen spatial mask, changes in bit probabilities can deactivate previously selected bits.

Finite bit locking (FBL). To reduce these interruptions, FBL retains a bit's editing permission if it was selected by the instantaneous bitwise mask in Eq. 4 at least once within the latest K steps. This replaces the instantaneous rule for m_t with:

$$m_{t,n,d} = \max_{\max(t_s,t-K+1)\leq j\leq t} \{a_{j,n}\,\mathbf{1}[\Delta_{j,n,d} > \tau_{\text{pow}}]\}\,. \tag{7}$$

Here, j indexes the latest finite K editing steps, including the current step. The maximum implements a logical OR over their instantaneous activations. Setting $K = 1$ recovers the instantaneous rule. When spatial selection remains adaptive, a recently activated bit may remain editable even if its position is excluded by the current spatial mask. Thus, the spatial mask gates instantaneous activations, while m_t also retains recent ones.

Locking preserves editing permission, not the bit value. Selected bits therefore continue to undergo prediction and random refinement. A bit loses permission once it has not passed the selection tests for K consecutive steps.

Overall, bit routing, AdaSF, and FBL balance source preservation with continued refinement of edited content. Bit routing anchors unselected bits to the evolving source, while AdaSF and FBL address unnecessary mask expansion and interruptions to bitwise editing. The complete training-free editing procedure is given in Algorithm 1.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUPS

Benchmarks. ***RefineEdit*** uses a pretrained GRN without additional training and generates images at 1024×1024 resolution. We evaluate it using the source and editing prompt pairs from nine editing categories of PIE-Bench (Ju et al., 2024): object replacement, addition and removal, content and pose modification, color and material modification, background modification, and style transfer. All prompts are taken directly from PIE-Bench. For each prompt pair, GRN generates a source image from the source prompt. All baselines edit this same source image using the corresponding editing prompt, providing a common source reference across methods. We use Grounded-SAM (Ren et al., 2024) to obtain foreground masks for regional evaluation. These masks are shared across methods and are separate from the dynamic masks predicted by ***RefineEdit***. Since the baselines produce 512×512 images in our setup, we resize the GRN source images and ***RefineEdit*** outputs to 512×512 before evaluation. The evaluation masks are aligned to the same resolution. We additionally evaluate on EditEval v2 (Huang et al., 2025) using the same settings from PIE-Bench without retuning. Details of the PIE-Bench and EditEval v2 evaluations are provided in Appendices D.1 and D.4, respectively.

Evaluation Metrics. We evaluate structural consistency, preservation of unedited content, and alignment with the editing prompt. Structure Distance (Tumanyan et al., 2022) measures structural differences between source and edited images. PSNR (Wang et al., 2004), LPIPS (Zhang et al., 2018), MSE (Wang & Bovik, 2009), and SSIM (Wang et al., 2004) measure content preservation in unedited regions. We assess semantic alignment using CLIP (Radford et al., 2021) similarity between the editing prompt and the edited image, computed over both the whole image and the designated editing region. We denote these two scores by $\mathrm{CLIP}_{\mathrm{tgt}}$ and $\mathrm{CLIP}_{\mathrm{edit}}$, respectively. All methods use the same evaluation images, masks, and metric implementations. Metrics details are provided in Appendix D.2.

Table 1: Quantitative comparison across nine editing categories of PIE-Bench. Methods are grouped by generation paradigm. **Bold** and underlined values indicate the best and second-best results, respectively. Background colors represent a diverging heatmap normalized by *performance*: Blue for worst, white for average, and Pink for best.

Method	Model	Struct. Dist.↓	Background Preservation PSNR↑	LPIPS↓	MSE↓	SSIM↑	CLIP Sim. Whole↑	Edited↑
Diffusion Methods								
P2P (Hertz et al., 2023)	SD 1.4	0.0572	19.02	0.149	0.0172	0.771	24.88	22.26
MasaCtrl (Cao et al., 2023)	SD 1.4	0.0297	22.25	0.092	0.0088	0.838	24.30	21.78
Pix2Pix-Zero (Parmar et al., 2023)	SD 1.4	0.0798	19.45	0.170	0.0184	0.771	23.94	21.92
PnP (Tumanyan et al., 2023)	SD 1.5	0.0271	22.80	0.091	0.0074	0.842	25.51	22.74
PnP-DirectInv (Ju et al., 2024)	SD 1.5	<u>0.0229</u>	23.37	0.081	0.0064	0.852	25.57	22.77
LEDits++ (Brack et al., 2024)	SD 1.5	0.0414	20.59	0.100	0.0120	0.820	<u>26.22</u>	<u>23.29</u>
ChordEdit (Lu et al., 2026)	SD-Turbo	0.0266	23.02	0.097	0.0071	0.812	24.94	22.41
Flow Methods								
FlowEdit (Kulikov et al., 2025)	FLUX.1-dev	**0.0162**	<u>25.03</u>	<u>0.062</u>	<u>0.0046</u>	<u>0.902</u>	24.85	21.95
RF-Inversion (Rout et al., 2025)	FLUX.1-dev	0.0484	19.73	0.176	0.0148	0.711	25.04	22.44
GRN Method								
RefineEdit (Ours)	GRN	0.0266	**30.36**	**0.032**	**0.0045**	**0.950**	**26.59**	**23.44**

Comparison Methods. We compare ***RefineEdit*** with nine training-free image editing baselines, grouped by the generative backbones used in our evaluation. Flow baselines include FlowEdit (Kulikov et al., 2025) and RF-Inversion (Rout et al., 2025). Diffusion baselines include MasaCtrl (Cao et al., 2023), Prompt-to-Prompt (P2P) (Hertz et al., 2023), Pix2Pix-Zero Parmar et al. (2023), PnP (Tumanyan et al., 2023), PnP-DirInv (Ju et al., 2024), LEDits++ (Brack et al., 2024), and ChordEdit (Lu et al., 2026). Comparison methods details are provided in Appendix D.3.

Implementation details. ***RefineEdit*** uses a frozen pretrained GRN without attention control. We select t_s, τ_{spa}, and τ_{pow} by editing category, while fixing $K = 4$ and $\tau_{\mathrm{frz}} = 2\tau_{\mathrm{spa}}$. Each inference run uses a single NVIDIA A100 GPU. Category-specific settings are provided in Appendix B.2.

5.2 Experimental Results

Figure 1 presents a *qualitatively* comparison of representative editing results. ***RefineEdit*** replaces spiderman with a robot while retaining the crouching pose and roof layout. LEDits++ and ChordEdit alter the pose or roof, whereas FlowEdit largely retains the source appearance. ***RefineEdit*** also adds a goat while preserving the mountain scene, removes sunglasses while retaining the car interior, and changes a woman's gaze without disturbing the bouquet. For color editing, it turns the butterfly green while retaining the surrounding composition. These examples illustrate localized changes with limited disruption to unrelated content. More *qualitatively* results are provided in Appendix F.

Table 1 reports *quantitative comparison* results over all nine editing categories. ***RefineEdit*** achieves the best scores on all four background-preservation metrics and both CLIP metrics. Compared with FlowEdit, it improves PSNR by 5.33 dB. Its edited-region CLIP score exceeds LEDits++ by 0.15, while whole-image alignment also ranks first. The gains in MSE and CLIP are modest, and FlowEdit remains stronger in Structure Distance. Overall, these results show strong preservation of unedited content without sacrificing alignment with editing prompts. On EditEval v2, ***RefineEdit*** also achieves strong content preservation and competitive alignment with editing prompts (Appendix D.4).

5.3 Ablations and analysis

Latency. We measure runtime on a single NVIDIA A100 GPU. Each editing operation is repeated ten times, and we report the average runtime excluding I/O time. As shown in Table 2, ***RefineEdit*** takes 26.77 seconds per edit at 1024×1024 resolution. Its runtime is comparable to FlowEdit and lower than RF-Inversion, while providing approximately threefold speedup over PnP and PnP-DirectInv.

Table 2: Runtime comparison with two diffusion and two flow editing methods. Type denotes the generation paradigm. Time is the average runtime per editing operation over ten runs, excluding I/O time.

Method	Type	Resolution	Time (s)↓
PnP	Diffusion	$1K$	79.70
PnP-DirectInv	Diffusion	$1K$	79.65
RF-Inversion	Flow	$1K$	32.05
FlowEdit	Flow	$1K$	27.15
RefineEdit (Ours)	GRN	$1K$	**26.77**

Effects of hyper-parameters. We qualitatively examine the three core parameters: the switch step t_s, the spatial threshold τ_{spa}, and the bitwise threshold τ_{pow}. Figure 4 shows the trade-off in choosing when to begin editing. Branching too early introduces target semantics before a clear source layout has emerged, allowing changes to spread into the background. Branching too late better preserves the source structure but leaves the intended transformation incomplete. This is consistent with a more established visual state being harder to change and fewer remaining steps for editable target refinement; the qualitative comparison alone does not distinguish these effects. In Figure 4, $t_s = 18$ provides a suitable balance between editability and preservation.

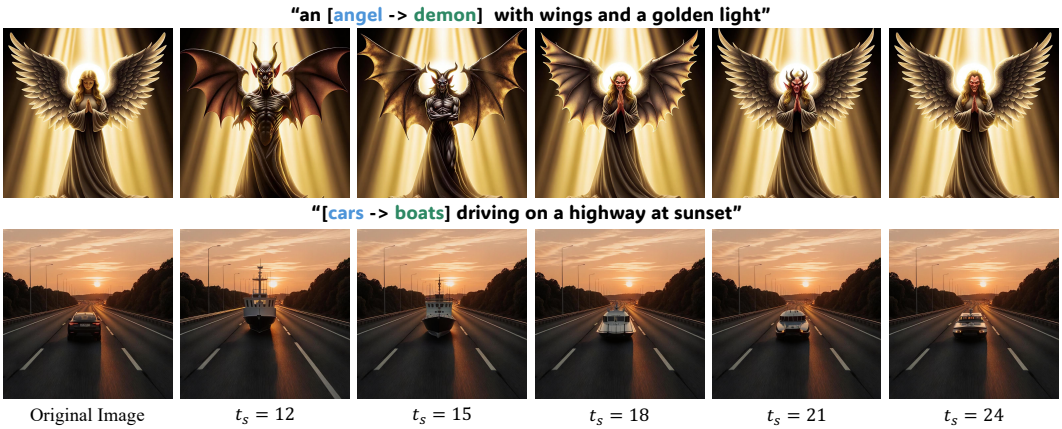


Figure 4: Effect of the t_s. Earlier branching permits stronger changes but can disturb the background, whereas later branching retains more source appearance and may leave the edit incomplete.



(a) (b)

Figure 5: Joint effects of spatial and bitwise thresholds. (a) Edited images. (b) Corresponding bit-density maps, where darker pixels indicate more active bits. Columns use $\tau_{\text{spa}} = 0.1, 0.2, 0.3$ from left to right; rows use $\tau_{\text{pow}} = 0.12, 0.16, 0.20$ from top to bottom.

With the switch step fixed, Figure 5 shows how the two thresholds control spatial coverage and bitwise edit strength. Increasing τ_{spa} from left to right reduces the selected region in Figure 5 (b), leaving more bread unchanged in Figure 5 (a). Increasing τ_{pow} from top to bottom reduces the fraction of active bits, producing lighter density maps and weaker edits. The thresholds interact through refinement: bit selection changes the editing state and the probability evidence used for later spatial masks. In these examples, stronger bitwise editing also expands the mask, highlighting the balance between editing coverage and background preservation. Controlled quantitative comparisons are provided in Appendix B.3.

Effects of mask stabilization. Figure 6 compares variants without *AdaSF* or *FBL*, with other settings unchanged. Without *AdaSF*, the editing mask spreads into the roof and background near the robot's hands, introducing unwanted changes. Without *FBL*, the mask remains more localized, but the fingers are not fully edited. Together, the two mechanisms balance spatial control with continued refinement of edited content. Quantitative results are provided in Appendix B.4.

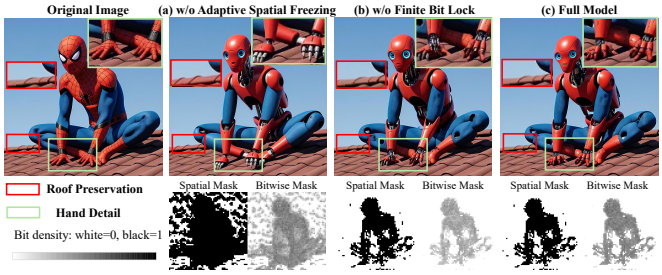


Figure 6: Qualitative mechanism ablation. Removing freezing edits the roof around the robot; removing bit locking leaves the fingers incompletely edited. Spatial masks and bit-density maps show the corresponding changes in editing support.

6 Conclusion

In this paper, we presented ***RefineEdit***, a training-free framework that adapts GRN to prompt-to-prompt image editing. Starting from an intermediate source state, it uses aligned bit probability differences to guide editing while preserving unselected source coordinates. It incorporates two stabilization techniques, *adaptive spatial freezing* and *finite bit locking*. *Adaptive spatial freezing* limits unnecessary expansion of the editing region, while *finite bit locking* keeps recently selected bits editable across refinement steps, reducing interruptions to the refinement of edited content. Experiments across nine PIE-Bench editing categories show strong preservation of unedited content and alignment with editing prompts. These results support our central insight that *refinement is inherently editable*: edit localization and content generation can evolve within the same process. Exploring this insight beyond GRN could open new directions for training-free image editing.

AI USE STATEMENT

Generative AI tools were employed exclusively for language polishing and proofreading. All AI-assisted revisions were carefully reviewed and verified by the authors, who assume full responsibility for the final content and integrity of this work.

REFERENCES

Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18208–18218, June 2022.

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 (CVPR)*, pp. 7877–7888, June 2025.

Manuel Brack, Felix Friedrich, Katharia Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinario Passos. Ledits++: Limitless image editing using text-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8861–8870, June 2024.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402. IEEE, 2023.

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 *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22503–22513. IEEE, 2023.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

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

Fengyi Fu, Lei Zhang, Mengqi Huang, and Zhendong Mao. Feededit: Text-based image editing with dynamic feedback regulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2661–2670, June 2025.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15733–15744, June 2025.

Jian Han, Jinlai Liu, Jiahuan Wang, Bingyue Peng, and Zehuan Yuan. Generative refinement networks for visual synthesis. In *European Conference on Computer Vision*, pp. 566–585. Springer, 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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_CDixzkzeyb.

Taihang Hu, Linxuan Li, Kai Wang, Yaxing Wang, Jian Yang, and Ming-Ming Cheng. Anchor token matching: Implicit structure locking for training-free ar image editing. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18166–18176. IEEE, 2025.

Yi Huang, Jiancheng Huang, Yifan Liu, Mingfu Yan, Jiaxi Lv, Jianzhuang Liu, Wei Xiong, He Zhang, Liangliang Cao, and Shifeng Chen. Diffusion model-based image editing: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4409–4437, 2025. doi: 10.1109/TPAMI.2025.3541625.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Pnp inversion: Boosting diffusion-based editing with 3 lines of code. In *International Conference on Learning Representations*, volume 2024, pp. 23395–23422, 2024.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, Krishna Somandepalli, Hassan Akbari, Yair Alon, Yong Cheng, Josh Dillon, Agrim Gupta, Meera Hahn, Anja Hauth, David Hendon, Alonso Martinez, David Minnen, Mikhail Sirotenko, Kihyuk Sohn, Xuan Yang, Hartwig Adam, Ming-Hsuan Yang, Irfan Essa, Huisheng Wang, David A. Ross, Bryan Seybold, and Lu Jiang. Videopoet: a large language model for zero-shot video generation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Senmao Li, Joost Van De Weijer, Taihang Hu, Fahad Shahbaz Khan, Qibin Hou, Yaxing Wang, Jian Yang, and Ming-Ming Cheng. Stylediffusion: Prompt-embedding inversion for text-based editing. *Computational Visual Media*, 12(3):743–763, 2026. doi: 10.26599/CVM.2025.9450462.

Liangsi Lu, Xuhang Chen, Minzhe Guo, Shichu Li, Jingchao Wang, and Yang Shi. Chordedit: One-step low-energy transport for image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14398–14407, June 2026.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–11, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, volume 2024, pp. 1862–1874, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8821–8831. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/ramesh21a.html`.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Hu0FSOSEyS.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Narek Tumanyan, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Splicing vit features for semantic appearance transfer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10738–10747. IEEE, 2022.

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 (CVPR)*, pp. 1921–1930, June 2023.

Jiahuan Wang, Yuxin Chen, Jun Yu, Guangming Lu, and Wenjie Pei. Editinfinity: Image editing with binary-quantized generative models. *Advances in Neural Information Processing Systems*, 38:50763–50790, 2026.

Yufei Wang, Lanqing Guo, Zhihao Li, Jiaxing Huang, Pichao Wang, Bihan Wen, and Jian Wang. Training-free text-guided image editing with visual autoregressive model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17577–17586. IEEE, 2025.

Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Tao Wu, Senmao Li, Yaxing Wang, Shiqi Yang, Kai Wang, and Joost van de Weijer. Training-free image inversion for one-step diffusion models. *Pattern Recognition*, 180:114063, 2026. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2026.114063. URL https://www.sciencedirect.com/science/article/pii/S0031320326010289.

Feng Xie, Jiagao Hu, Fuhao Li, Zepeng Wang, Yuxuan Chen, Dahua Gao, Fei Wang, and Daiguo Zhou. Grnedit: Efficient general video editing from a new binary-evidence perspective in generative refinement networks. *arXiv preprint arXiv:2608.16328*, 2026.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=AFDcYJKhND. Featured Certification.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Appendix Contents

Appendix A Statements . . . 13

A.1 Limitations . . . 13

A.2 Ethics Statement . . . 13

A.3 Reproducibility Statement . . . 14

Appendix B Implementation and Hyperparameter Settings . . . 14

B.1 Fixed GRN Settings . . . 14

B.2 RefineEdit Hyperparameters . . . 14

B.3 Controlled Hyperparameter Comparisons . . . 15

B.4 Mechanism Ablations . . . 15

Appendix C User Study . . . 16

Appendix D Evaluation Details . . . 17

D.1 Dataset . . . 17

D.2 Evaluation Metrics . . . 17

D.3 Compared Methods . . . 20

D.4 Cross-Dataset Evaluation on EditEval v2 . . . 21

Appendix E Algorithm of RefineEdit . . . 22

Appendix F Additional Qualitative Results . . . 24

A Statements

A.1 Limitations

RefineEdit focuses on editing images generated by GRN using their source refinement trajectories. It does not directly support arbitrary real images; extending the framework to this setting is a direction for future work.

A.2 Ethics Statement

Our evaluation uses PIE-Bench prompts and source images generated with GRN. Like other generative image editing methods, ***RefineEdit*** could be misused to create misleading content or manipulate visual evidence. It may also inherit biases and harmful associations from the pretrained generator. Our editing mechanisms do not address these risks. Responsible use should respect privacy, consent, and intellectual property rights, and disclose image modifications when they could otherwise be mistaken for authentic records.

A.3 Reproducibility Statement

RefineEdit uses frozen pretrained models and requires no additional training. Appendix B reports the model configuration, sampling settings, random seeds, and category-specific hyperparameters. We select category-specific hyperparameters based on PIE-Bench evaluation results to balance editing quality and source preservation. Algorithm 1 summarizes the editing procedure using the notation and equations from the main text. All compared methods share the same source images and evaluation masks. Appendix D describes image preprocessing, metric definitions, and aggregation over valid samples to clarify how the reported results are obtained.

B Implementation and Hyperparameter Settings

B.1 Fixed GRN Settings

We use the pretrained GRN-2B (Han et al., 2026) text-to-image model with the frozen HBQ tokenizer and UMT5-XXL text encoder. All network weights remain fixed. Table 3 lists the shared inference settings. Images are generated at 1024×1024 resolution. The corresponding HBQ code has 64×64 spatial positions and 256 bits per position, obtained from 64 latent channels and four quantization rounds. Generation starts from random binary labels, with each bit sampled as 0 or 1 with equal probability. The two branches share this initial random code, which is reused whenever a coordinate returns to a random label. We use separate random generators for source and editing sampling, seeded with 42 and 43, respectively.

Table 3: Fixed GRN settings used for ***RefineEdit***. These settings are shared across editing categories.

Setting	**Value**
Generative backbone	GRN-2B, T2I checkpoint 251600
Visual tokenizer	HBQ, 64 channels, 4 quantization rounds
Tokenizer checkpoint	Image/video tokenizer dated 2026-06-26
Text encoder	UMT5-XXL
Generation resolution	1024×1024
Spatial code grid	64×64
Refinement iterations	50, indexed from 0 to 49
Classifier-free guidance scale	3.0
Sampling temperature	1.1
Bit sampling	Multinomial sampling over $\{0, 1\}$
Prediction-retention schedule	Cosine schedule, maximum ratio 0.95
Schedule shift	1.0
Source / editing random seeds	42 / 43
Negative prompt	Empty string
Inference autocast precision	bfloat16
Additional attention control	None

Both branches follow the same global refinement schedule and finish at step 49. For the input to zero-based step t, the nominal prediction-retention ratio is

$$\lambda_t = 0.95 \left[1 - \cos\left(\frac{\pi t}{2 \cdot 49} \right) \right], \qquad t = 0, \ldots, 49. \tag{8}$$

The selection mask is sampled at each step, so its realized prediction fraction can differ slightly from this nominal ratio. The implementation uses the realized source fraction for the shared progress embedding. At the final step, we decode the sampled predictions after applying source/edit routing, rather than decoding the intermediate state that still contains random labels. We do not add refinement steps or restart the noise schedule after editing begins. A switch step t_s therefore leaves $50 - t_s$ editing iterations, including the branching step itself.

Table 4: Category-specific settings used in the quantitative comparison. All categories use $K = 4$ and $\tau_{\text{frz}} = 2\tau_{\text{spa}}$.

Subset	Editing category	Images	t_s	τ_{spa}	τ_{pow}
1	Object replacement	80	15	0.013	0.20
2	Object addition	80	15	0.016	0.10
3	Object removal	80	10	0.020	0.16
4	Content modification	40	12	0.016	0.18
5	Pose modification	40	12	0.018	0.16
6	Color modification	40	11	0.020	0.16
7	Material modification	40	16	0.020	0.20
8	Background modification	80	10	0.012	0.16
9	Style transfer	80	6	0.016	0.14

B.2 *RefineEdit* HYPERPARAMETERS

We select the switch step t_s, spatial threshold τ_{spa}, and bitwise threshold τ_{pow} for each editing category. Table 4 lists the configurations used in the main quantitative comparison. All images within a category share the same configuration.

FIXED MASK STABILIZATION. We set $K = 4$ for all categories. A bit selected in any of the latest four refinement steps remains eligible for editing, but its binary value can continue to change. For adaptive spatial freezing, we fix the response multiplier to 2, giving $\tau_{\text{frz}} = 2\tau_{\text{spa}}$. This fixes the multiplier, not the absolute threshold across categories. If the mean excess spatial response at the switch step reaches this threshold, we freeze the spatial mask immediately. Otherwise, the mask remains dynamic through the final step. These mechanisms are evaluated by removing each one separately, rather than by sweeping their settings.

B.3 CONTROLLED HYPERPARAMETER COMPARISONS

Table 5 reports controlled comparisons for the three core parameters. The switch-step comparison uses the object-replacement subset, while the spatial and bitwise threshold comparisons use the color-modification subset. Each block varies only the indicated parameter. The other two parameters, random seeds, GRN settings, $K = 4$, and the adaptive-freezing rule remain fixed. All entries are taken from completed grid runs, with the means verified against their per-image evaluation records. These are slices of the existing search, not additional runs.

SWITCH STEP. Later branching improves background preservation in this comparison, while both CLIP scores decrease. This supports a trade-off between retaining source structure and allowing target changes. Because the endpoint stays fixed, changing t_s changes both the inherited source state and the number of editing iterations. These results do not isolate either factor as the sole cause.

SPATIAL THRESHOLD. On the color-modification subset, increasing τ_{spa} improves background preservation but lowers edited-region CLIP. This illustrates the trade-off between restricting spatial selection and allowing the requested semantic change. The comparison uses the full AdaSF rule, so τ_{frz} scales with τ_{spa}.

BITWISE THRESHOLD. Increasing τ_{pow} restricts the bits eligible for editing at each selected position. In this comparison, all reported background metrics improve, but both whole-image and edited-region CLIP decrease. Stricter bit selection therefore favors source preservation at the expense of semantic editing.

B.4 MECHANISM ABLATIONS

We evaluate *AdaSF* and *FBL* on the object-replacement subset by disabling one mechanism at a time. Without *AdaSF*, the spatial mask is updated at every step, with *FBL* still enabled. Without

FBL, only the current step's candidate bits remain active, equivalent to $K = 1$, with *AdaSF* still enabled. All other settings remain unchanged. Figure 6 and Table 6 present the qualitative and quantitative comparisons, respectively.

Removing *FBL* improves background preservation but lowers both CLIP scores, indicating that retaining recent bit activations supports semantic editing. Removing *AdaSF* improves CLIP scores at the cost of background preservation and structural consistency. These results illustrate their complementary roles: *AdaSF* restricts unwanted spatial expansion, and *FBL* sustains editing activity across refinement steps.

Table 5: Representative controlled parameter comparisons. The switch-step comparison uses the object-replacement subset, and the threshold comparisons use the color-modification subset. Only one input hyperparameter varies within each block, with AdaSF and FBL enabled. Background metrics are evaluated outside the reference edit mask. Higher PSNR, SSIM, and CLIP and lower LPIPS are better.

Hyperparameters			**Background Preservation**			**CLIP Similarity**	
t_s	τ_{spa}	τ_{pow}	PSNR↑	LPIPS↓	SSIM↑	Whole↑	Edited↑
Varying the switch step							
15	0.016	0.14	26.08	0.042	0.939	26.04	20.12
18	0.016	0.14	29.18	0.027	0.959	25.05	19.33
20	0.016	0.14	31.14	0.019	0.969	24.34	18.57
Varying the spatial threshold							
14	0.010	0.12	29.11	0.031	0.948	26.73	21.95
14	0.014	0.12	33.25	0.023	0.965	26.54	21.55
14	0.020	0.12	36.49	0.018	0.976	26.42	21.35
Varying the bitwise threshold							
13	0.010	0.09	26.92	0.037	0.937	27.36	22.46
13	0.010	0.12	28.67	0.031	0.950	27.07	22.24
13	0.010	0.16	31.11	0.023	0.965	26.87	21.69

Table 6: Mechanism ablations on the object-replacement subset. Each variant disables one mechanism while keeping the other settings unchanged. Arrows indicate the better direction.

Method	**Structure**	**Background Preservation**			**CLIP Similarity**	
	Distance↓	**PSNR↑**	**LPIPS↓**	**SSIM↑**	**Whole↑**	**Edited↑**
Full model	0.028	30.04	0.044	0.932	26.48	23.17
w/o Finite Bit Locking	0.028	31.04	0.041	0.938	26.34	23.01
w/o Adaptive Spatial Freezing	0.033	28.76	0.050	0.920	26.59	23.44

C User Study

We conducted a user study with 20 participants to evaluate human preferences for our editing results against those of LEDits++, FlowEdit, and ChordEdit. The study comprised 20 sets of editing results (400 votes in total) randomly sampled from a pool spanning all editing categories. As illustrated in Fig. 8, participants were shown four candidate results with the method names hidden and asked to select the best one. As shown in Fig. 7, our method received a higher proportion of preference votes than each baseline.

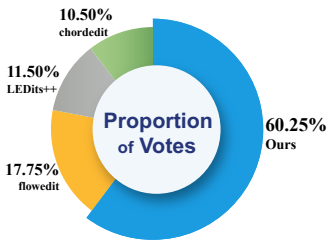


Figure 7: Human preference results from the user study.

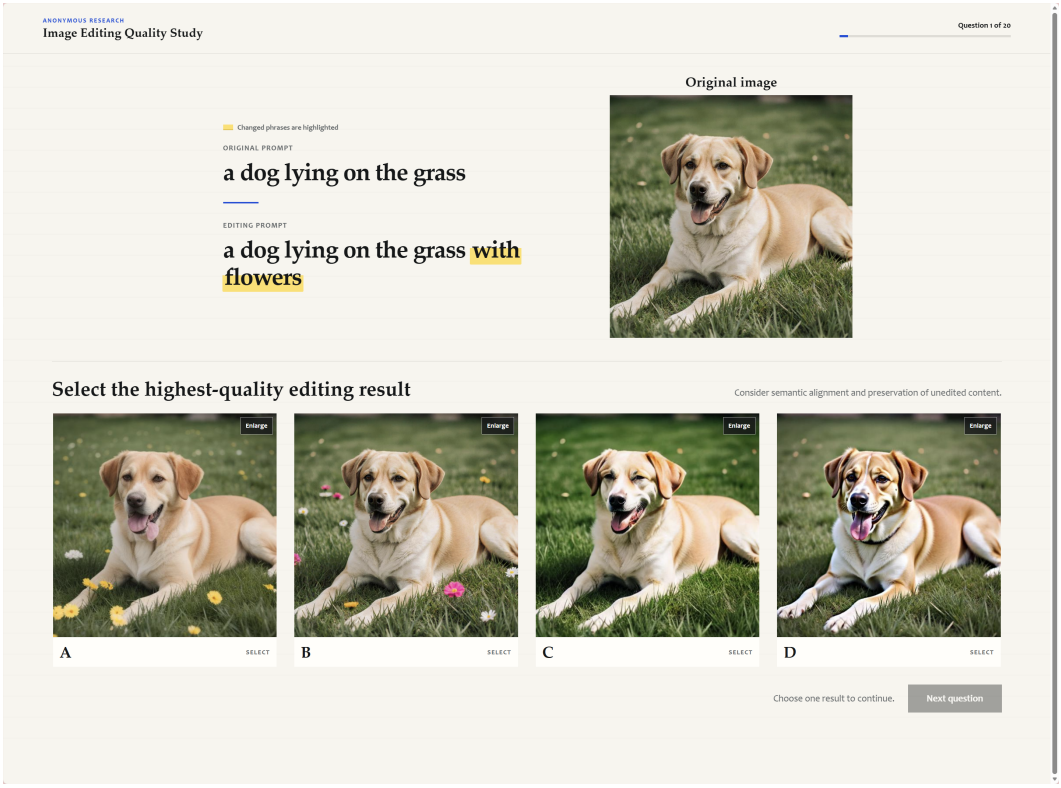


Figure 8: User study interface.

D Evaluation Details

D.1 Dataset

We construct a generation-to-editing evaluation set from PIE-Bench, containing 560 prompt pairs across nine editing categories. Each pair consists of a source prompt and an editing prompt that specifies the desired change. Table 7 summarizes the categories and their sample counts.

For each source prompt, we use GRN to generate a source image at 1024×1024 resolution. ***RefineEdit*** edits the corresponding generation trajectory, while all baselines receive the same source image. This shared source ensures that content preservation is evaluated against the same reference across methods. The source image serves as the reference for preservation, rather than as a ground-truth edited target.

We use Grounded-SAM to obtain source-aligned region annotations for evaluation. The reference editing masks specify which regions may change and are distinct from the masks predicted by ***RefineEdit***. They are used only for evaluation and are not provided to our method during inference.

Before evaluation, source and edited images are resized to 512×512 using interpolation. Reference masks are decoded at their encoded resolution and resized using nearest-neighbor interpolation to preserve their binary values. This protocol standardizes the evaluation resolution across methods.

Figure 9 presents one example from each category, including the prompt change, source image, and outputs from ***RefineEdit*** and representative baselines. These examples cover both localized object edits and broader changes to the scene or style.

D.2 Evaluation Metrics

We evaluate structural consistency, preservation of unedited content, and alignment with the editing prompt. Let I^s and I^e denote the source and edited RGB images, and let c^e denote the editing prompt. Let $M \in \{0,1\}^{H \times W}$ be the reference editing mask, where $M = 1$ indicates an editable

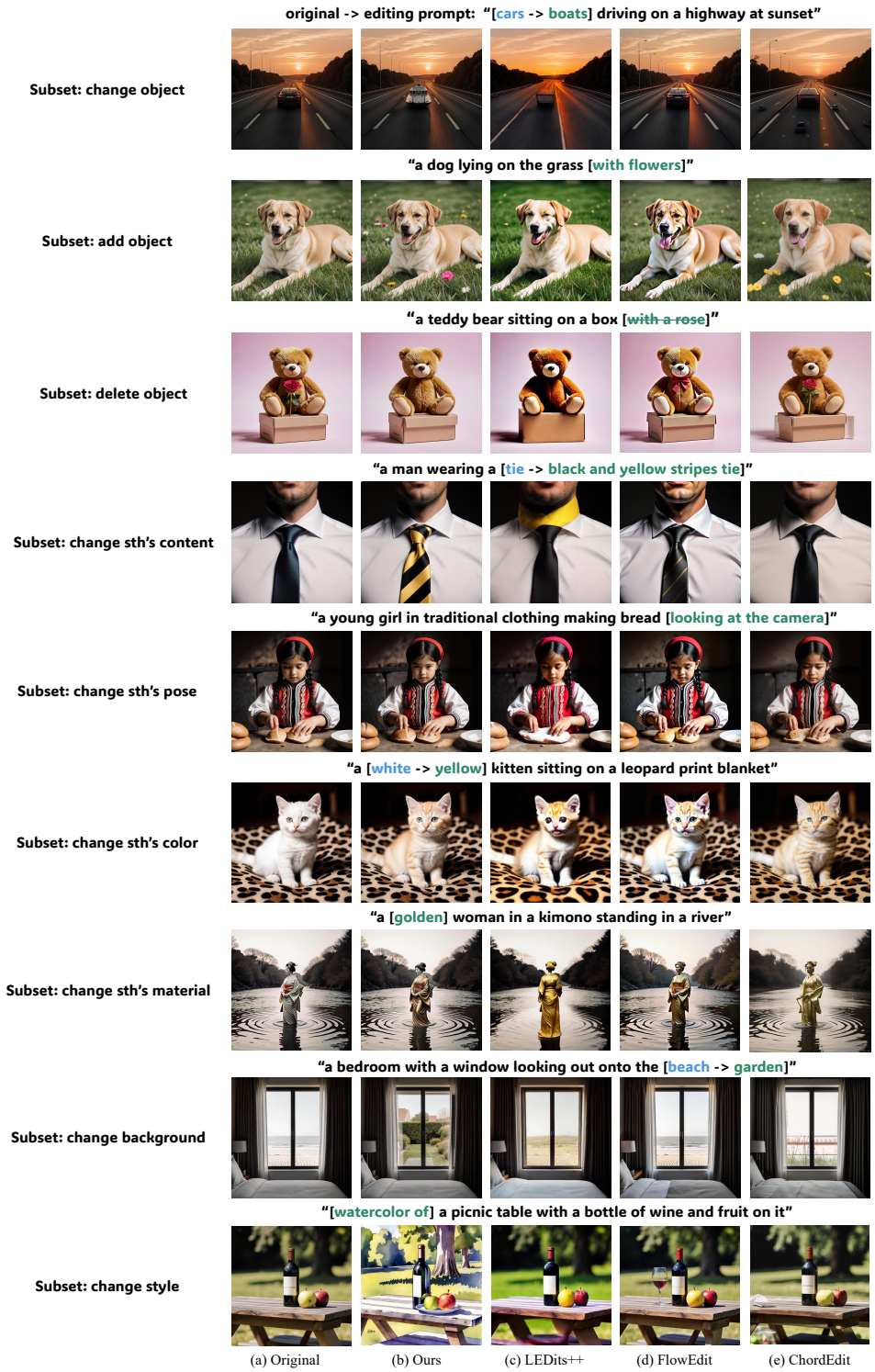


Figure 9: Examples from the nine editing categories. Each row shows a source image and the corresponding results from ***RefineEdit***, LEDits++, FlowEdit, and ChordEdit. Prompt annotations indicate the content to replace, add, or remove.

Table 7: Composition of our PIE-Bench-derived evaluation set. Examples summarize the edits illustrated in Fig. 9.

Category	Samples	Description and example
Object replacement	80	Replace an object with another category, such as cars with boats.
Object addition	80	Introduce a new object, such as flowers on the grass beside a dog.
Object removal	80	Remove an existing object, such as a rose held by a teddy bear.
Content modification	40	Modify an object's attributes or internal appearance, such as adding black and yellow stripes to a tie.
Pose modification	40	Change an object's pose, action, or orientation, such as making a girl look toward the camera.
Color modification	40	Change an object's color, such as turning a white kitten yellow.
Material modification	40	Change an object's material appearance, such as giving a person a golden appearance.
Background modification	80	Change the surrounding scene, such as replacing a beach view with a garden.
Style transfer	80	Change the image's overall visual style, such as converting a photograph into a watercolor painting.

region. The images used to evaluate preservation are

$$U = (1 - M) \odot I^s, \qquad V = (1 - M) \odot I^e, \tag{9}$$

where the mask is broadcast across RGB channels. Thus, editable regions are set to zero in both images.

Although we refer to these measures as background preservation, they evaluate all content outside the editing mask. For background modification, this preserved content may instead be the foreground subject.

STRUCTURE DISTANCE. (TUMANYAN ET AL., 2022) Structure Distance measures changes in the relationships between image tokens using a pretrained DINO ViT-B/8. Let $k_i(I)$ denote the concatenated attention-key features of token i from the final Transformer layer, following the preprocessing used by our evaluation implementation. The token self-similarity matrix is

$$A_{ij}(I) = \frac{k_i(I)^\top k_j(I)}{\max(\|k_i(I)\|_2 \|k_j(I)\|_2, \epsilon)}, \tag{10}$$

where ϵ ensures numerical stability. Structure Distance is the mean squared difference between the source and edited self-similarity matrices:

$$D_{\text{structure}} = \frac{1}{T^2} \sum_{i=1}^{T} \sum_{j=1}^{T} \left(A_{ij}(I^s) - A_{ij}(I^e)\right)^2, \tag{11}$$

where T is the number of image tokens, including the class token used by the implementation. This metric is computed on the full images. Lower values indicate stronger structural consistency.

MEAN SQUARED ERROR (MSE). (WANG & BOVIK, 2009) MSE measures pixel differences in the masked preservation images. With RGB values normalized to $[0, 1]$, it is computed as

$$\text{MSE} = \frac{1}{3HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{3} \left(U_{hwc} - V_{hwc}\right)^2. \tag{12}$$

Following the evaluation implementation, the denominator is the size of the full RGB image, rather than the number of unmasked coordinates. Lower values indicate better preservation.

PEAK SIGNAL-TO-NOISE RATIO (PSNR). (WANG ET AL., 2004) PSNR expresses pixel reconstruction fidelity on a logarithmic scale. For images normalized to $[0, 1]$, it is

$$\text{PSNR} = 10 \log_{10} \left(\frac{1}{\text{MSE}}\right), \tag{13}$$

where MSE is defined in Eq. 12. Higher PSNR indicates better preservation of the source content.

LEARNED PERCEPTUAL IMAGE PATCH SIMILARITY (LPIPS). (ZHANG ET AL., 2018) LPIPS measures perceptual differences using pretrained visual features. We use the SqueezeNet implementation. Let $\phi_\ell(U)_{hw}$ and $\phi_\ell(V)_{hw}$ be the channel-normalized features at layer ℓ and spatial position (h, w). LPIPS is

$$\mathrm{LPIPS} = \sum_\ell \frac{1}{H_\ell W_\ell} \sum_{h,w} w_\ell^\top \left(\phi_\ell(U)_{hw} - \phi_\ell(V)_{hw}\right)^{\odot 2}, \tag{14}$$

where w_ℓ contains learned channel weights and $\odot 2$ denotes element-wise squaring. The masked images are mapped from $[0, 1]$ to $[-1, 1]$ before LPIPS preprocessing. The metric is evaluated on the complete masked images, without additional normalization by the preserved area. Lower values indicate greater perceptual similarity.

STRUCTURAL SIMILARITY INDEX MEASURE (SSIM). (WANG ET AL., 2004) SSIM compares local luminance, contrast, and structure. For corresponding local windows u and v from U and V,

$$\mathrm{SSIM}(u, v) = \frac{(2\mu_u\mu_v + C_1)(2\sigma_{uv} + C_2)}{(\mu_u^2 + \mu_v^2 + C_1)(\sigma_u^2 + \sigma_v^2 + C_2)}, \tag{15}$$

where μ_u and μ_v are local means, σ_u^2 and σ_v^2 are local variances, and σ_{uv} is their covariance. For the unit intensity range, we use $C_1 = 0.01^2$ and $C_2 = 0.03^2$. Local statistics use an 11×11 Gaussian window with standard deviation 1.5. The final score averages the local SSIM values over the masked images and RGB channels. Higher values indicate better preservation.

CLIP SIMILARITY. (RADFORD ET AL., 2021) We measure semantic alignment using CLIP ViT-L/14. Let f_I and f_T denote its image and text encoders, including their respective preprocessing. The image-text score is

$$C(I, c) = 100 \max\left(0,\ \frac{f_I(I)^\top f_T(c)}{\|f_I(I)\|_2 \|f_T(c)\|_2}\right). \tag{16}$$

We report alignment with the editing prompt for both the full image and the edited region:

$$\mathrm{CLIP}_{\mathrm{tgt}} = C(I^e, c^e), \qquad \mathrm{CLIP}_{\mathrm{edit}} = C(M \odot I^e, c^e). \tag{17}$$

For edited-region CLIP, pixels outside the reference editing mask are set to black before CLIP preprocessing; the image is not cropped to the mask's bounding box. Both scores use the complete editing prompt. Higher values indicate stronger target alignment.

The evaluator additionally records source-image alignment, $C(I^s, c^s)$, where c^s is the source prompt. This score describes the generated source image and is not an editing-performance measure.

AGGREGATION. Metrics are computed per image and averaged over finite values using the same rule for all methods. For metric m, let $\mathcal{V}_m$ contain samples with finite scores. The reported mean is

$$\overline{m} = \frac{1}{|\mathcal{V}_m|} \sum_{i \in \mathcal{V}_m} m_i. \tag{18}$$

Each valid image receives equal weight, rather than each category. Undefined scores and infinite PSNR values are excluded, not replaced by zero. Valid counts can therefore differ across metrics and methods. PSNR is averaged per image, rather than computed from the pooled MSE.

D.3 COMPARED METHODS

We compare ***RefineEdit*** with nine training-free editing methods based on pretrained diffusion or rectified-flow models. We group the baselines by the generative backbones used in our evaluation. Diffusion methods use SD 1.4, SD 1.5, or SD-Turbo, while flow methods use FLUX.1-dev.

Diffusion Editing Methods

PROMPT-TO-PROMPT (P2P). (HERTZ ET AL., 2023) Prompt-to-Prompt controls the cross-attention maps that connect text tokens to image regions. By replacing, retaining, or reweighting these maps during generation, it introduces changes specified by the editing prompt while preserving the source layout.

MASACTRL. (CAO ET AL., 2023) MasaCtrl replaces diffusion self-attention with mutual self-attention, allowing the editing branch to retrieve related content and textures from the source branch. Masks derived from cross-attention help distinguish foreground from background and reduce incorrect feature matching. This supports changes in pose and shape while preserving the source object's appearance.

PIX2PIX-ZERO. (PARMAR ET AL., 2023) Pix2Pix-Zero represents a semantic edit as a direction in the text-embedding space. It uses cross-attention guidance to retain the source attention maps during editing, helping preserve content outside the intended change. The method requires no task-specific training.

PLUG-AND-PLAY (PNP). (TUMANYAN ET AL., 2023) PnP extracts spatial features and self-attention information from the source diffusion trajectory and injects them into generation conditioned on the editing prompt. These features preserve the source layout while the editing prompt guides changes in appearance. The method requires no additional training or fine-tuning.

PNP WITH DIRECT INVERSION (PNP-DIRINV). (JU ET AL., 2024) This baseline combines PnP with Direct Inversion, which separates source reconstruction from editing. The source branch preserves the input, while the editing branch introduces the requested changes. PnP provides additional structure control through feature injection. We evaluate this combination separately from standard PnP.

LEDITS++. (BRACK ET AL., 2024) LEDits++ combines an efficient, optimization-free inversion procedure with semantic editing guidance. Its implicit masks restrict changes to relevant image regions. The framework also supports multiple simultaneous edits.

CHORDEDIT. (LU ET AL., 2026) ChordEdit formulates editing as transport between distributions defined by the source and editing prompts. Using dynamic optimal transport, it derives a smoothed, low-energy editing field that reduces trajectory variance. This enables editing through a single large integration step without additional training. We evaluate ChordEdit with the SD-Turbo backbone.

Flow Editing Methods

FLOWEDIT. (KULIKOV ET AL., 2025) FlowEdit constructs an ordinary differential equation that directly transports images between distributions defined by the source and editing prompts. It combines the corresponding model fields to form an editing trajectory without first recovering a noise latent. The method requires neither inversion nor test-time optimization.

RF-INVERSION. (ROUT ET AL., 2025) RF-Inversion supports reconstruction and editing with pretrained rectified-flow models. It derives a controlled inversion field using a linear quadratic regulator and relates the resulting dynamics to a rectified stochastic differential equation. The controlled trajectory enables editing while retaining source content, without additional model training.

D.4 CROSS-DATASET EVALUATION ON EDITEVAL V2

Evaluation setup. We conduct an additional evaluation on EditEval v2 (Huang et al., 2025) to assess whether the category-specific settings selected on PIE-Bench transfer to another dataset without retuning. EditEval v2 contains 150 source and editing prompt pairs across seven categories: object replacement, object addition, object removal, action change, texture change, background change, and style change. Each category uses the corresponding PIE-Bench configuration in Table 4. Action and texture changes use the settings for pose and material modification, respectively. All other GRN settings remain unchanged.

We follow the generation-to-editing protocol described in Appendix D.1. GRN generates a source image from each source prompt, and all methods edit this same image using the editing prompt. Thus, the evaluation uses EditEval prompt pairs with source images produced by GRN, rather than the original benchmark photographs. Image preprocessing, regional evaluation, and metric aggregation follow the same protocol as our PIE-Bench experiments.

Results. Table 8 highlights the balance between content preservation and semantic editing. ***RefineEdit*** achieves the best PSNR, LPIPS, and SSIM, together with the second-best whole-image and edited-region CLIP scores among the evaluated methods. Although LEDits++ achieves higher CLIP scores, this advantage is accompanied by substantially poorer performance on all four background-preservation metrics. FlowEdit obtains lower Structure Distance and MSE, but shows weaker alignment with editing prompts and lower preservation performance on PSNR, LPIPS, and SSIM. In comparison, ***RefineEdit*** combines competitive prompt alignment with leading performance on three complementary preservation metrics. This balance is maintained on EditEval using the PIE-Bench settings without retuning, supporting their transferability across datasets.

Table 8: Quantitative comparison across seven editing categories of EditEval v2 under our generation-to-editing protocol. ***RefineEdit*** reuses the corresponding PIE-Bench settings without retuning. Methods are grouped by generation paradigm. **Bold** and underlined values indicate the best and second-best results, respectively. Cell colors indicate normalized performance within each metric, from blue for worst through white to pink for best.

Method	Model	Struct. Dist.↓	Background Preservation PSNR↑	LPIPS↓	MSE↓	SSIM↑	CLIP Sim. Whole↑	Edited↑
Diffusion Methods								
P2P	SD 1.4	0.049	17.97	0.168	0.0198	0.701	23.89	20.37
MasaCtrl	SD 1.4	0.028	20.76	0.105	0.0112	0.776	22.85	19.64
Pix2Pix-Zero	SD 1.4	0.079	16.90	0.228	0.0278	0.677	23.15	19.58
PnP	SD 1.5	0.025	20.60	0.109	0.0107	0.773	24.33	20.69
PnP-DirectInv	SD 1.5	0.024	20.82	0.108	0.0103	0.774	24.45	20.69
LEDits++	SD 1.5	0.044	18.94	0.124	0.0158	0.750	**25.74**	**21.64**
ChordEdit	SD-Turbo	0.030	20.40	0.139	0.0120	0.717	24.00	20.22
Flow Methods								
FlowEdit	FLUX.1-dev	**0.019**	22.06	0.092	**0.0083**	0.843	23.72	20.05
RF-Inversion	FLUX.1-dev	0.045	18.07	0.228	0.0195	0.610	23.59	20.06
GRN Method								
RefineEdit (Ours)	GRN	0.030	**24.60**	**0.065**	0.0112	**0.892**	25.12	21.19

E ALGORITHM OF *RefineEdit*

Algorithm 1 summarizes ***RefineEdit*** using the notation introduced in the main text. $\mathrm{SampleBits}(P)$ samples each binary coordinate from its predicted distribution over $\{0, 1\}$. Bernoulli sampling is applied independently to each coordinate. The two branches share a fixed random code Z and the same refinement schedule, but use separate random streams. For brevity, G_θ includes the standard GRN inference settings for guidance and temperature. The decoder $\mathcal{D}$ includes conversion from HBQ bits to image features.

The algorithm performs T prediction steps indexed from 0 to $T-1$. For $t < T-1$, λ_{t+1} controls the prediction fraction used to construct the next input. At $t = T-1$, we update the editing mask and directly decode the routed sampled predictions, without constructing another input state or using λ_T. In the algorithm, $\bar{a}_t$ denotes spatial selection before freezing, and $\bar{m}_t$ denotes instantaneous bit selection before locking. The initial response r is computed once at t_s and remains unchanged throughout subsequent steps.

Algorithm 1: ***RefineEdit***: Training-Free Prompt-to-Prompt Image Editing

Input: Frozen GRN G_θ and HBQ decoder $\mathcal{D}$; source prompt c^s and editing prompt c^e; total prediction steps $T \geq 2$, switch step $0 \leq t_s < T$, and prediction-ratio schedule $\{\lambda_t\}_{t=0}^{T-1}$ with $\lambda_0 = 0$; thresholds τ_{spa}, τ_{pow}, and τ_{frz}; bit-lock duration $K \geq 1$.

Output: Source image I^s and edited image I^e.

$Z_{n,d} \sim \text{Bernoulli}(1/2)$ for every binary coordinate (n, d);
$Y_0^s \leftarrow Z$;
// Source-only refinement
for $t = 0, \ldots, t_s - 1$ **do**
  $P_t^s \leftarrow \text{softmax}(G_\theta(Y_t^s, c^s, t))$; // Eq. 1
  $\widehat{Y}_t^s \leftarrow \text{SampleBits}(P_t^s)$;
  $S_{t,n,d}^s \sim \text{Bernoulli}(\lambda_{t+1})$;
  $Y_{t+1}^s \leftarrow S_t^s \odot \widehat{Y}_t^s + (1 - S_t^s) \odot Z$; // Eq. 2
$Y_{t_s}^e \leftarrow Y_{t_s}^s$; // Shared-state branching; Sec. 4.1
// Coupled edit localization and refinement
for $t = t_s, \ldots, T - 1$ **do**
  for $u \in \{s, e\}$ **do**
    $P_t^u \leftarrow \text{softmax}(G_\theta(Y_t^u, c^u, t))$; // Eq. 1
    $\widehat{Y}_t^u \leftarrow \text{SampleBits}(P_t^u)$;
  // Compare probabilities of the same source-sampled bit
  $\widehat{y}_{t,n,d} \leftarrow \widehat{Y}_{t,n,d}^s$;
  $\Delta_{t,n,d} \leftarrow P_t^s(n, d, \widehat{y}_{t,n,d}) - P_t^e(n, d, \widehat{y}_{t,n,d})$; // Eq. 3
  $q_{t,n} \leftarrow \frac{1}{D} \sum_{d=1}^{D} \Delta_{t,n,d}$;
  $\bar{a}_{t,n} \leftarrow \mathbf{1}[q_{t,n} > \tau_{\text{spa}}]$; // Eq. 4
  // AdaSF: compute the initial response only once
  if $t = t_s$ **then**
    $\Omega \leftarrow \{n \mid \bar{a}_{t_s,n} = 1\}$;
    $r \leftarrow \frac{\sum_{n \in \Omega}(q_{t_s,n} - \tau_{\text{spa}})}{\max(1, |\Omega|)}$; // Eq. 6
  // Apply the fixed freezing decision at every editing step
  if $r \geq \tau_{\text{frz}}$ **then**
    $a_t \leftarrow \bar{a}_{t_s}$; // Reuse the initial spatial mask
  else
    $a_t \leftarrow \bar{a}_t$; // Use the current spatial mask
  $\bar{m}_{t,n,d} \leftarrow a_{t,n}\, \mathbf{1}[\Delta_{t,n,d} > \tau_{\text{pow}}]$; // Eq. 4
  // FBL: retain recent editing permissions
  $m_t \leftarrow \bigvee_{j=\max(t_s, t-K+1)}^{t} \bar{m}_j$; // Eq. 7
  // Construct the next input only before the final step
  if $t < T - 1$ **then**
    for $u \in \{s, e\}$ **do**
      $S_{t,n,d}^u \sim \text{Bernoulli}(\lambda_{t+1})$;
      $\widetilde{Y}_{t+1}^u \leftarrow S_t^u \odot \widehat{Y}_t^u + (1 - S_t^u) \odot Z$; // Eq. 2
    $Y_{t+1}^s \leftarrow \widetilde{Y}_{t+1}^s$;
    $Y_{t+1}^e \leftarrow m_t \odot \widetilde{Y}_{t+1}^e + (1 - m_t) \odot Y_{t+1}^s$; // Eq. 5
// Decode final sampled predictions without mixing with Z
$I^s \leftarrow \mathcal{D}(\widehat{Y}_{T-1}^s)$;
$I^e \leftarrow \mathcal{D}\Big(m_{T-1} \odot \widehat{Y}_{T-1}^e + (1 - m_{T-1}) \odot \widehat{Y}_{T-1}^s\Big)$;
return I^s, I^e;

F Additional Qualitative Results

We provide additional qualitative comparisons across the nine editing categories in our evaluation set. Figures 10–18 show how different methods introduce the requested changes while preserving content unrelated to the editing prompt. These examples complement the quantitative evaluation in the main text.

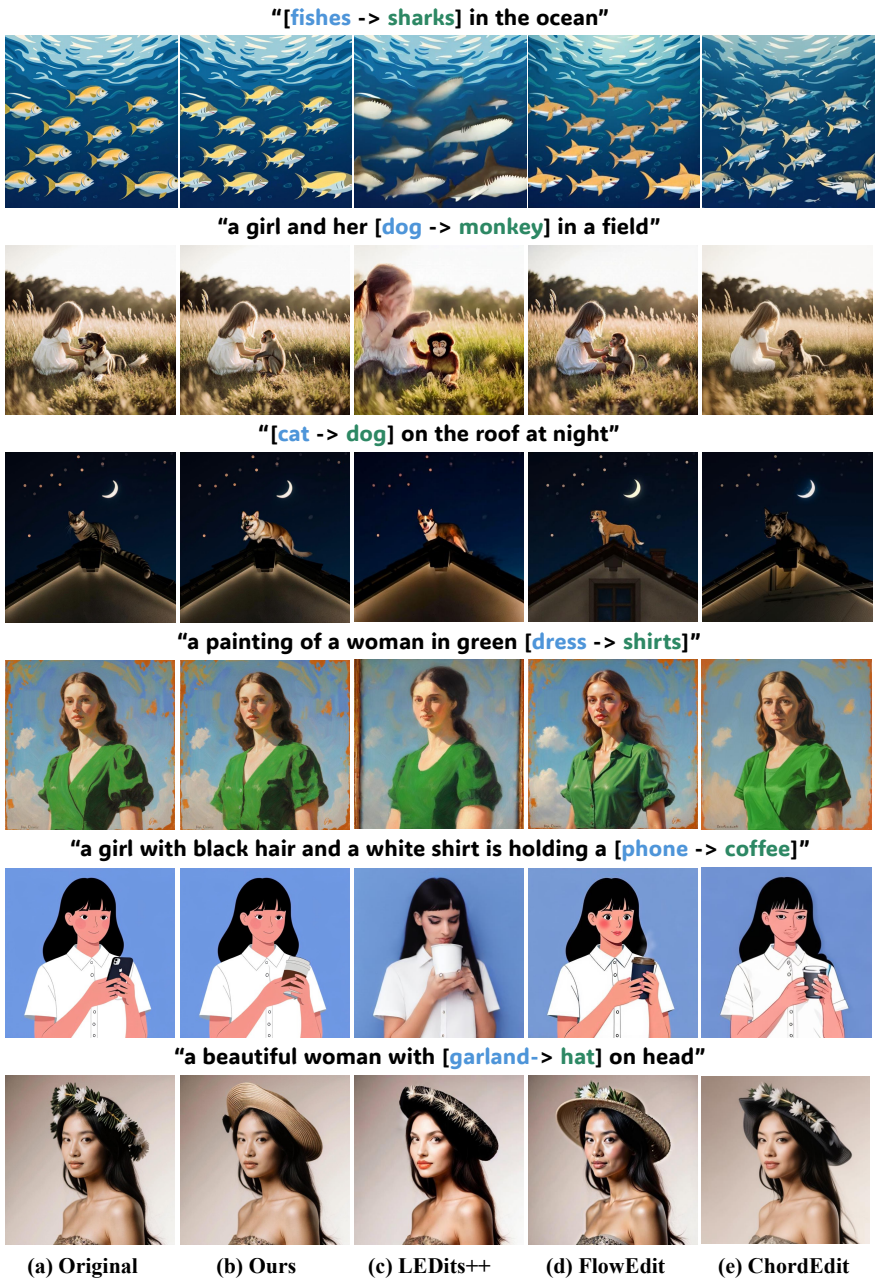


Figure 10: Qualitative comparison on object replacement. The editing prompt replaces a source object with another category while leaving unrelated scene content unchanged.

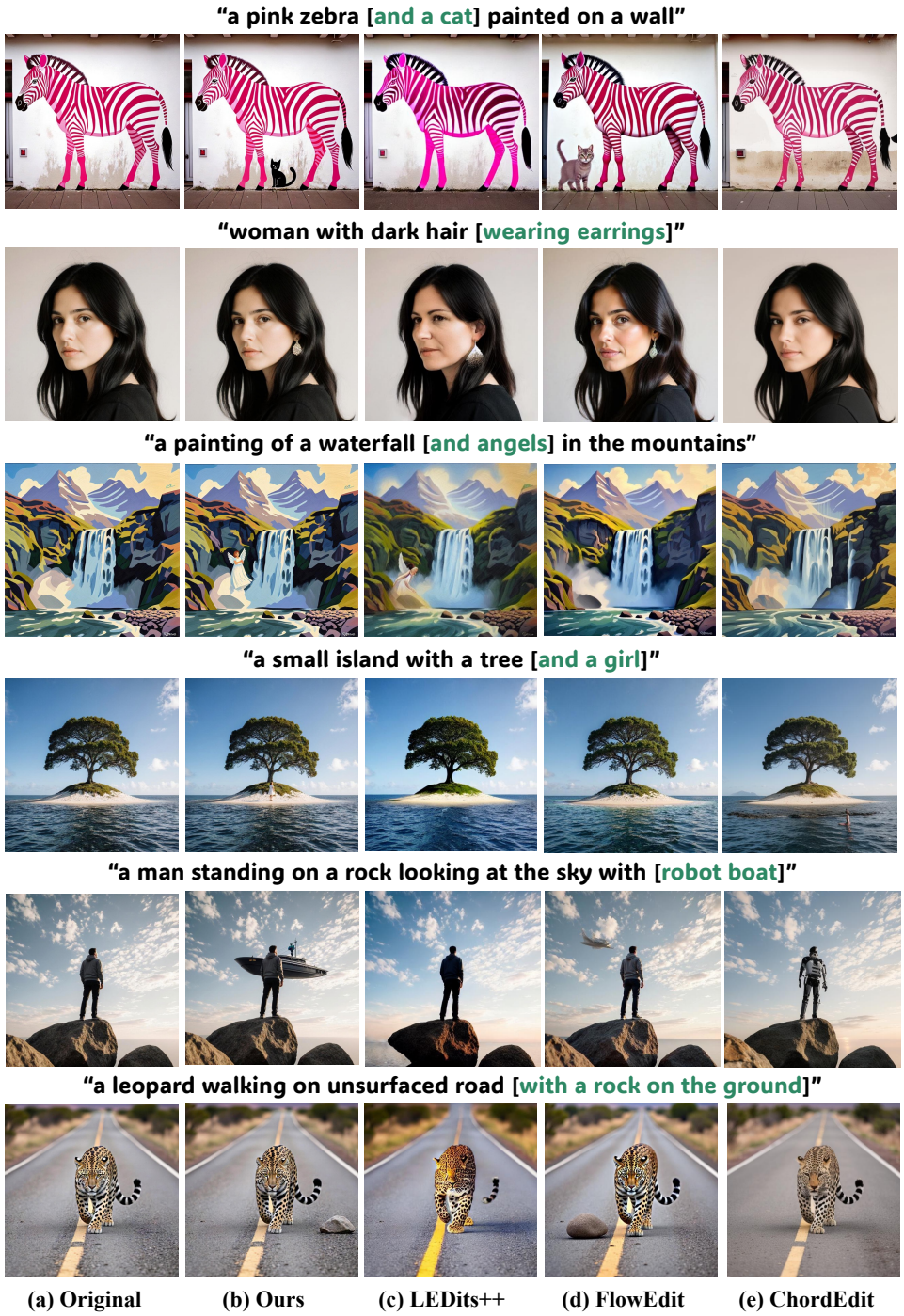


Figure 11: Qualitative comparison on object addition. The task introduces new objects specified by the editing prompt while preserving existing content and integrating the additions into the scene.



Figure 12: Qualitative comparison on object removal. The specified object should be removed and its former region filled coherently, without unnecessary changes elsewhere.

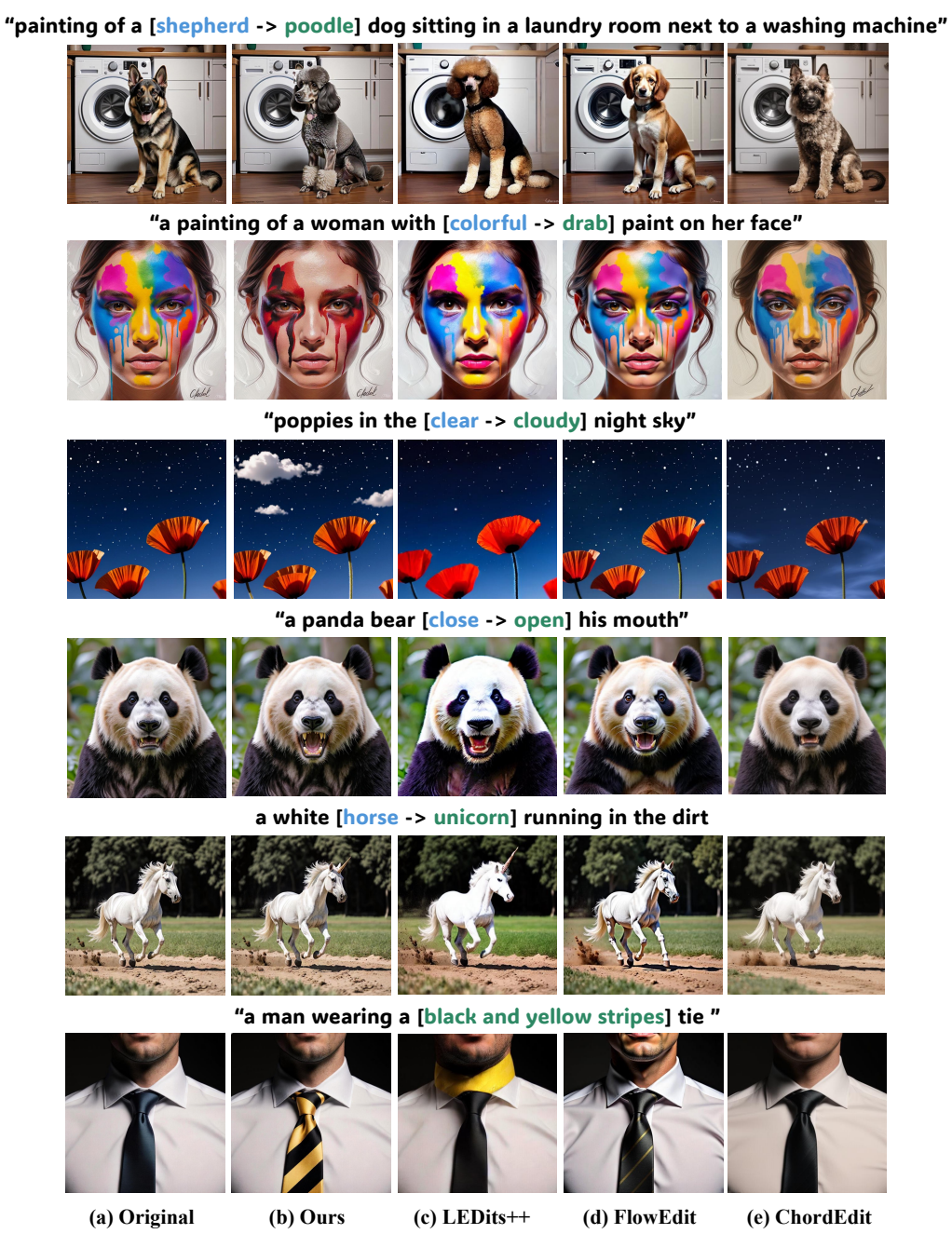


Figure 13: Qualitative comparison on content modification. The task changes an object's attributes or internal appearance, such as its expression or pattern, while retaining unrelated object and scene characteristics.



Figure 14: Qualitative comparison on pose modification. The editing prompt changes a subject's pose, action, or orientation. Successful editing requires the corresponding structural changes while maintaining a coherent subject and scene.

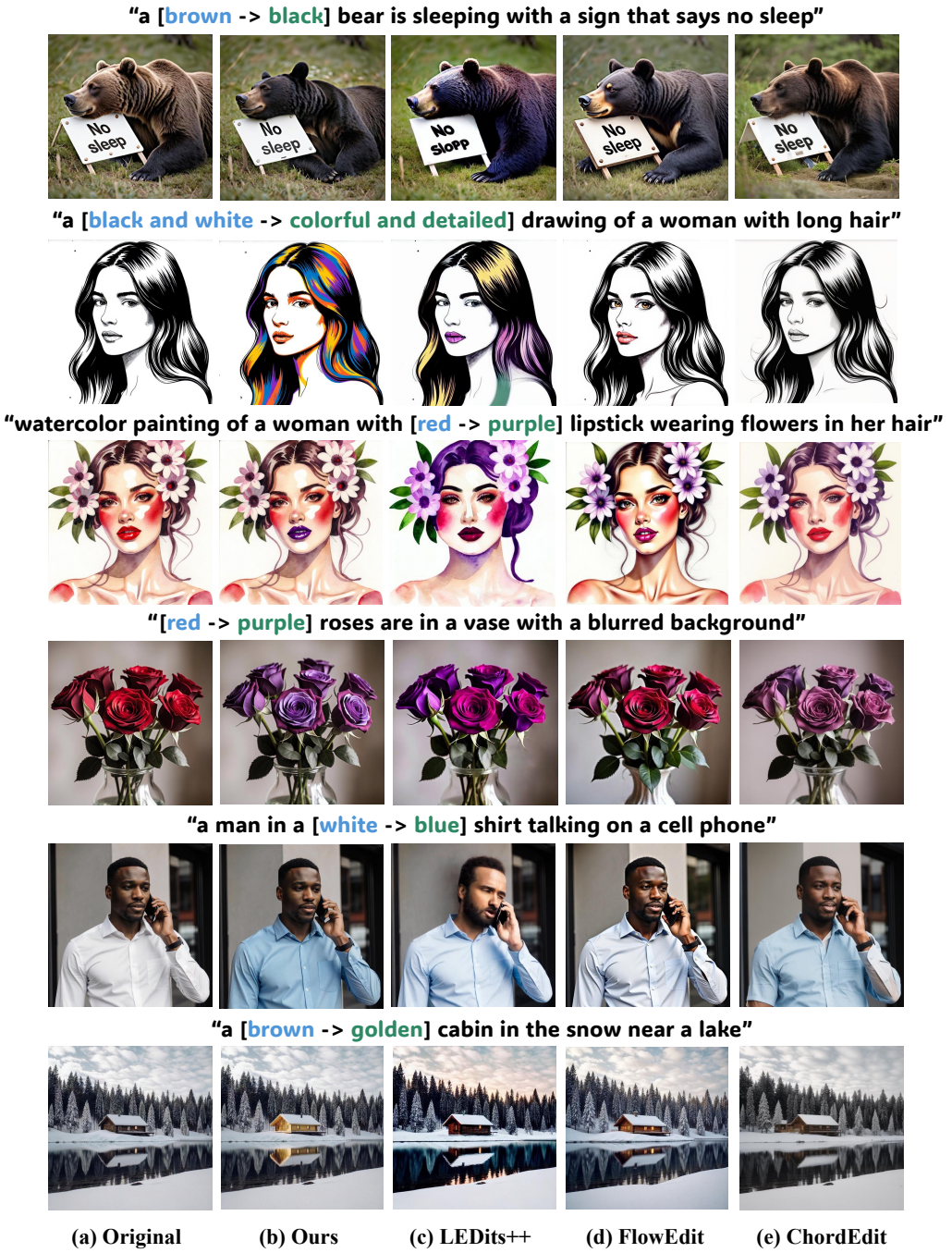


Figure 15: Qualitative comparison on color modification. The specified color change should remain localized to the target object while preserving its shape and the appearance of unrelated regions.



Figure 16: Qualitative comparison on material modification. The task changes the target object's material appearance, including its texture and reflectance, while retaining its overall form and surrounding content.

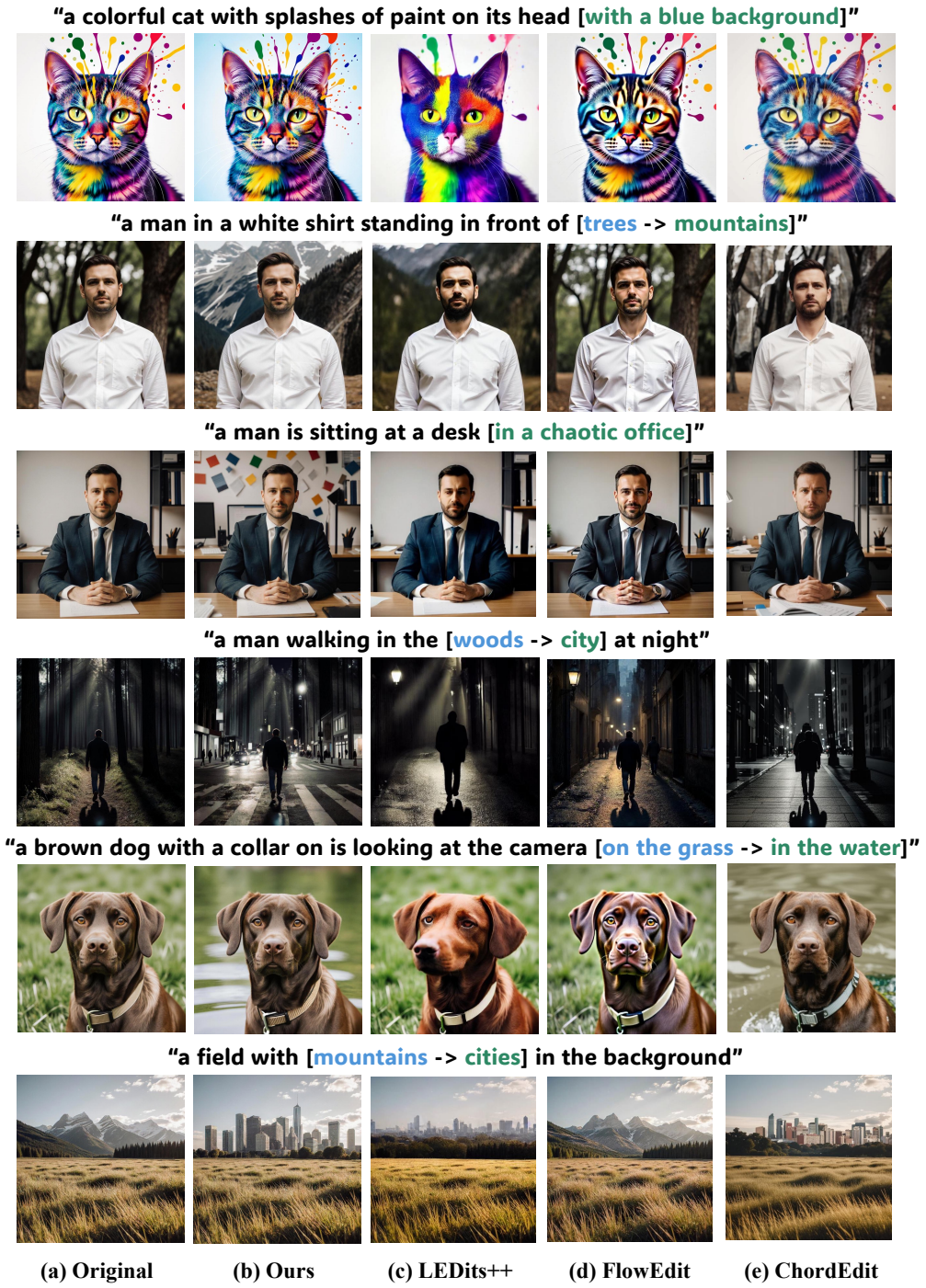


Figure 17: Qualitative comparison on background modification. The surrounding scene is changed according to the editing prompt, while the foreground subject should remain consistent with the source image.

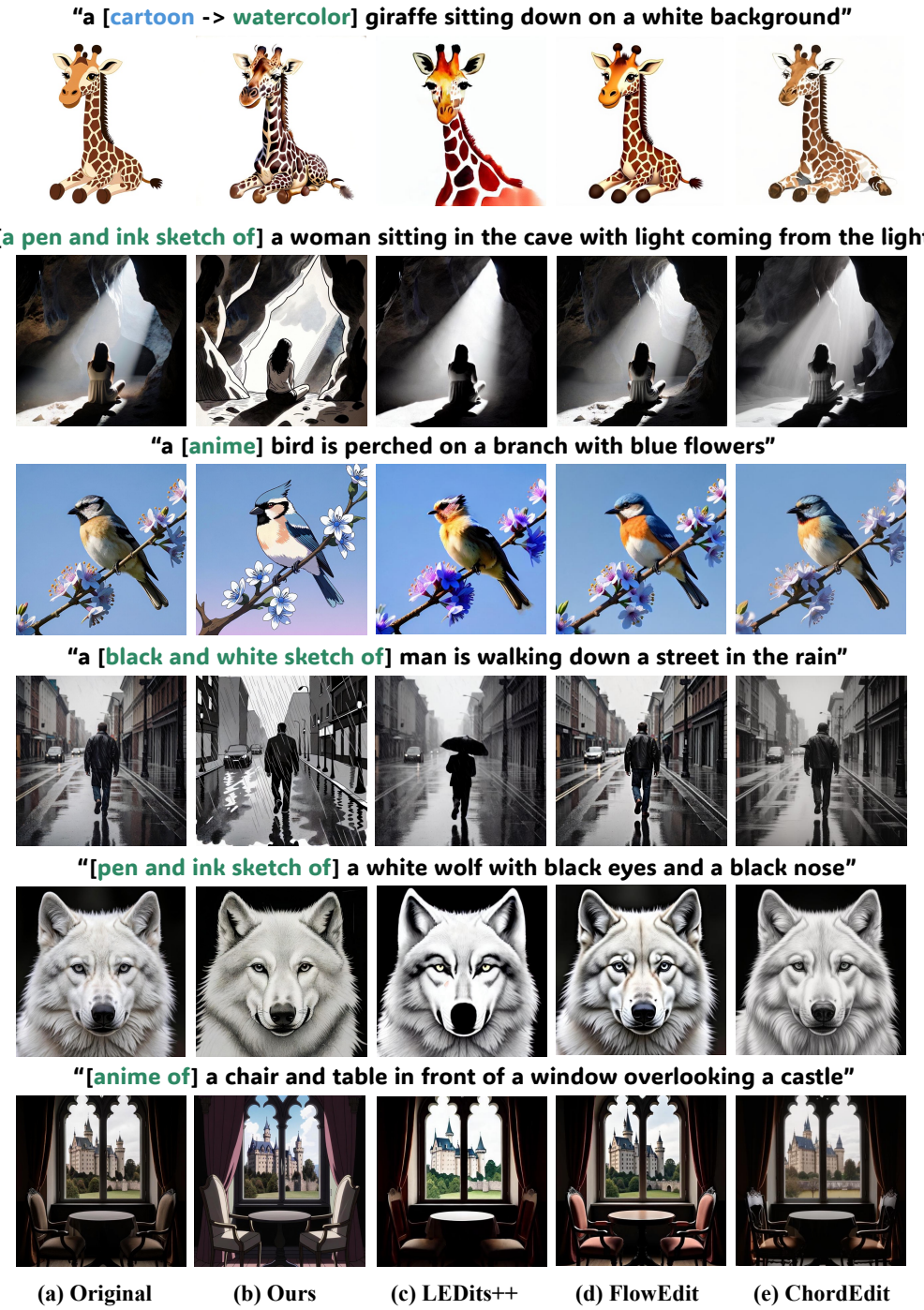


Figure 18: Qualitative comparison on style transfer. The target style is applied across the image while preserving recognizable scene content and composition. Unlike localized editing, this task permits changes throughout the image.