UNIFYING TASK REFORMULATION PIPELINES INTO AN IMAGE EDITING HARNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work has shown that external pipelines can substantially improve image editors by reorganizing how editing requests are processed. However, existing pipelines are typically designed around specific weaknesses or failure modes. We collect and devise several pipelines, observe their behavior on five benchmarks, and identify three characteristic behaviors. Their benefits are request-dependent and complementary; unnecessary interventions can degrade otherwise successful edits; and naively combining multiple pipeline guidances can introduce conflicts rather than accumulate their benefits. Motivated by these observations, we propose a unified image-editing harness that selectively configures task reformulation for each request. Specifically, a two-stage router selects an appropriate reformulation, on-demand skills provide pipeline-specific guidance, and shared execution controls manage visual context, tool interventions, and recovery. We evaluate the resulting system with FLUX.2-klein-9B as the primary editor across five image-editing benchmarks, against a broad set of strong baselines, together with module ablations and cost analysis. Without updating the primary editor's parameters, our harness substantially improves its editing performance and approaches the quality of Nano Banana Pro at a lower inference cost.

1 INTRODUCTION

General-purpose image editors increasingly support diverse visual modifications through natural-language instructions (Brooks et al., 2023; Sheynin et al., 2024; Black Forest Labs et al., 2025; Wu et al., 2025). Yet directly supplying the original image and instruction can still fail when a request requires precise localization, implicit reasoning, or substantial spatial change (Ye et al., 2025; Pu et al., 2025; Zhang et al., 2026). External agents and harnesses offer a complementary way to improve editing without modifying the editor itself, by reorganizing model calls, intermediate conditions, tools, and feedback. Some systems assemble heterogeneous generation and editing tools, as in GenArtist (Wang et al., 2024), while others compose specialized editing adapters on a shared backbone, as in IEAP (Hu et al., 2025). In this work, we focus on a complementary setting: given a general-purpose editor, how can an external harness reorganize the editing task so that the editor can better use its existing capabilities?

Recent systems have explored several effective ways to reorganize an editing request. RePlan grounds an instruction into region-aligned plans and controls execution through attention-region injection (Qu et al., 2026). EditThinker iteratively critiques intermediate results and refines editing instructions (Li et al., 2025), while Agent Banana uses layer-based editing and context management to support fidelity and multi-turn consistency (Ye et al., 2026). Figure 1(a) contrasts representative mechanisms: instruction refinement, sequential editor composition, and region control. Although these systems differ substantially in implementation, each is largely organized around a predefined family of transformations applied to the original editing problem. Consequently, the choice of how to reformulate a request is primarily embedded in the system design, rather than made adaptively for each request. A reformulation that is effective for one failure mode may therefore be less suitable for another, while extending a system with additional reformulations requires deciding how their guidance and execution should interact. This raises three questions: *Do different task reformulations provide complementary benefits? If so, is there value in unifying them within one system? And, if they are unified, how should the system select and execute them without introducing conflicting guidance or unnecessary interventions?*

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

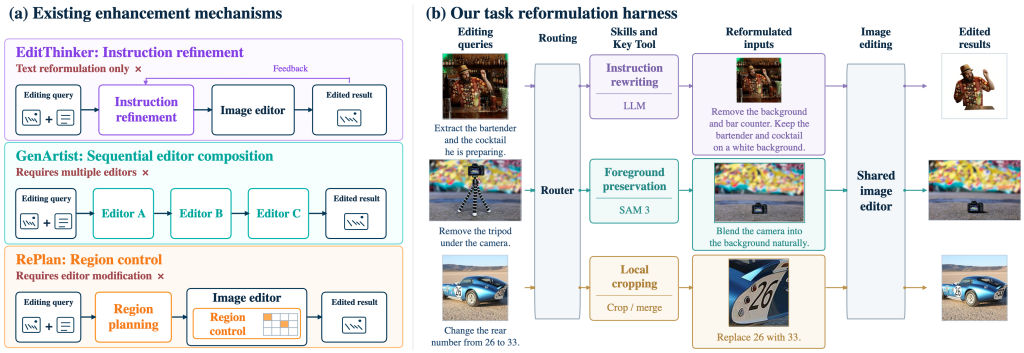

Figure 1: Existing enhancement mechanisms and our task-reformulation harness. (a) EditThinker (Li et al., 2025), GenArtist (Wang et al., 2024), and RePlan (Qu et al., 2026) illustrate instruction refinement, editor composition, and region control, respectively. (b) Our harness routes each query to a suitable skill and prepares tailored inputs for a shared image editor. Examples illustrate instruction rewriting, foreground preservation, and local cropping.

We study these questions through the lens of *task reformulation*: reorganizing the textual conditions, visual conditions, or operation sequence of an editing request to construct intermediate editing problems while preserving its intended final goal. A diagnostic pilot comparing REWRITE, LOCAL, and SPATIAL with DIRECT editing reveals three properties. First, different reformulations recover different direct-editing failures, with no single pipeline uniformly preferable across requests. Second, unnecessary interventions can degrade edits that already succeed under direct execution. Third, combining all guidance in one prompt does not match the recovery coverage of separately executed pipelines. Together, these observations motivate *selective rather than additive* unification.

Based on these findings, we propose a unified image-editing harness that represents heterogeneous reformulation pipelines through a common *Task Configuration* interface. A request-level router selects a configuration and activates its corresponding skill. Guided by this skill, a multimodal planner prepares editing conditions and integrates results within a shared execution loop, separating *what reformulation to use* from *how to execute it*. Figure 1(b) illustrates this routing and skill-guided execution around an unchanged primary editor.

We use FLUX.2-klein-9B (Black Forest Labs, 2026) as the primary editor and evaluate our harness on five image-editing benchmarks against a broad set of strong baselines, including both agentic editing systems and proprietary editors. We further conduct module-level ablations to examine the contributions of routing, skills, and execution control, together with a detailed cost analysis of the resulting system. Across these evaluations, our harness consistently improves the underlying FLUX editor and enables it to approach the editing performance of Nano Banana Pro (Google, 2026b) while operating at substantially lower inference cost.

Our contributions are threefold:

- We systematically study multiple task-reformulation pipelines and reveal their request-dependent behavior: different reformulations recover complementary failures of direct editing, while unnecessary interventions can introduce regressions and naively combining all pipeline guidance can lead to conflicts rather than their cumulative benefit.

- Motivated by these observations, we introduce a unified image-editing harness built around request-level routing and configuration-specific skills, enabling heterogeneous reformulation pipelines to be selectively invoked and coordinated within a shared execution framework.

- We conduct extensive experiments on five image-editing benchmarks against a broad range of strong baselines, together with module ablations and cost analysis. Our harness substantially improves FLUX.2-klein-9B, enabling it to achieve performance close to Nano Banana Pro at a lower inference cost.

2 Related Work

2.1 Enhancing General-Purpose Image Editors

Instruction-based editing benefits from training data that cover diverse requests and editing conditions, including manually annotated examples in MagicBrush (Zhang et al., 2023) and large-scale collections in UltraEdit and AnyEdit (Zhao et al., 2024; Yu et al., 2025). These efforts improve the editor through its training data; our focus is on reorganizing requests at inference time around an existing editor.

Recent works improve general-purpose editors without modifying their parameters by reorganizing the editing problem: EditThinker (Li et al., 2025) refines instructions through iterative critique, and RePlan (Qu et al., 2026) decomposes requests into region-aware plans. We view these as forms of *task reformulation* that better expose the capabilities of an existing editor, yet each is designed around a particular weakness or strategy. In contrast, we study multiple reformulation pipelines jointly and investigate when each should be applied.

2.2 Workflow Design for Agentic Image Editing

Another line of work organizes complex editing workflows using planners, tools, and feedback: Visual ChatGPT (Wu et al., 2023) connects language-model planning with visual foundation models, GenArtist (Wang et al., 2024) orchestrates heterogeneous visual tools, IEAP (Hu et al., 2025) programs sequences of specialized editing adapters, Agent Banana (Ye et al., 2026) adopts layer-based representations and context management, and JarvisEvo (Lin et al., 2026) combines iterative tool-based editing with editor–evaluator optimization.

These workflows also relate to general agent execution: ReAct (Yao et al., 2023) interleaves reasoning with actions, Self-Refine (Madaan et al., 2023) iterates feedback and revision, and Reflexion (Shinn et al., 2023) retains verbal feedback to guide subsequent attempts. Our work instead focuses on selecting among alternative task reformulations: request-level routing chooses a strategy, while the planner adapts concrete tool calls and editing conditions from intermediate feedback within a shared harness.

3 Analyzing Enhancement Pipelines

Existing enhancement pipelines improve image editors through different forms of intervention, but are typically designed for particular weaknesses or failure modes. Before introducing our unified framework, we study three questions: *when does intervention help or hurt, are different pipelines complementary, and can their benefits be obtained by simply combining their guidance?*

3.1 Enhancement Pipelines and Diagnostic Setup

We consider three representative enhancement pipelines. Rewrite refines the textual instruction before editing; Local crops and edits a target region before reintegrating it into the image; and Spatial decomposes spatially demanding requests into segmentation, rearrangement, and repair. We retain Direct, which applies the original instruction to the original image, as the baseline.

Despite their different implementations, these pipelines share a common principle: they reorganize the textual conditions, visual conditions, or editing operations presented to a fixed editor. We refer to this shared mechanism as *task reformulation*.

Our diagnostic study contains 480 requests drawn from five editing benchmarks. For each request, we collect two independent Direct outputs and one output from each enhancement pipeline, using FLUX.2-klein-9B as the editor and GPT-5.6 Luna as the planner. The analysis is restricted to these four configurations and excludes Reference. Outputs are evaluated three times by Qwen3.8-27B and averaged for analysis. We additionally report *Direct-Oracle-Bo2*, the retrospective better of the two direct samples, as a diagnostic reference for repeated direct sampling.

3.2 Recovering Direct-Editing Failures

Let s_i^A and $s_i^{A_2}$ be the direct scores. Diagnostic thresholds are $\tau_i = 3$ for ImgEdit-Bench's Basic-Edit and Understanding-Grounding-Editing suites (ImgEdit-Basic and ImgEdit-UGE in tables) and IV-Edit, and $\tau_i = 0.3$ for PICABench and VIBE. The consistently failed set is

$$\mathcal{F} = \{i : \max(s_i^A, s_i^{A_2}) < \tau_i\}. \tag{1}$$

The assisted pool recovers a failure if at least one output from Spatial, Local, or Rewrite reaches its threshold ($s \geq \tau$). Table 1 shows that the assisted pool recovers most persistent direct-editing failures, with recovery observed across all evaluated benchmarks. Thus, reformulating the task can expose capabilities that remain unused under repeated direct sampling from the same editor. The Aggregated Context estimates come from a separate cohort and are discussed in Section 3.4. Pooled recovery measures retrospective coverage across collected outputs, not realized routing performance or gains at matched computation.

Table 1: Recovery of direct-editing failures. n: requests; parentheses: recovered/failure counts.

Set	n	Bo2 failures	Aggregated Context (est., %)	Assisted pool (%)
ImgEdit-Basic	100	28	42.9	67.9 (19/28)
ImgEdit-UGE	45	17	17.6	70.6 (12/17)
PICABench	100	12	58.3	100.0 (12/12)
IV-Edit	100	31	71.0	93.5 (29/31)
VIBE	135	43	74.4	97.7 (42/43)
Total	480	131	58.0	87.0 (114/131)

3.3 Complementary Benefits and Intervention Risks

Table 2 reports three diagnostics. *Best-result share* distributes highest-score credit among tied pipelines, using Direct-Oracle-Bo2 for Direct. *Rescue* measures successful edits on the direct-failure subset; *harm* measures regressions on the direct-success subset. Requests outside these subsets contribute only to best-result share.

Table 2: Pipeline complementarity and intervention risks. Parentheses indicate case counts.

Pipeline	Best share (%)	Rescue (%)	Harm (%)
Direct (Oracle-Bo2)	20.1	—	—
Spatial	23.7	55.0 (72)	9.0 (24)
Local	30.6	71.8 (94)	3.7 (10)
Rewrite	25.6	66.4 (87)	7.5 (20)

No single pipeline is preferable across all requests. Local has the largest best-result share and the strongest individual rescue, but Spatial and Rewrite also account for substantial shares of the best outputs. Direct remains competitive on a subset of requests, indicating that reformulation is not always needed. All assisted pipelines also harm some otherwise successful direct edits, including Local despite its comparatively low harm rate. The same intervention can therefore be useful for one request and counterproductive for another. These observations motivate retaining alternative configurations, including Direct, and checking the applicability of auxiliary operations. Individual rescue uses $s > \tau$, whereas pooled recovery in Table 1 uses $s \geq \tau$; their numerical gap therefore does not isolate coverage attributable to complementarity.

3.4 From Complementary Pipelines to a Unified Harness

Useful individual pipelines do not specify how a planner should coordinate their guidance. Our *Aggregated Context* comparison combines the pipeline and tool instructions into a single prompt for direct editing, without routing or an iterative tool-execution loop. We examine whether this form of prompt-level integration retains the recovery potential of the individual reformulation pipelines.

Table 1 compares estimated Aggregated Context recovery with the observed assisted-pool recovery. The Aggregated Context estimates use per-benchmark recovery rates from a separate cohort, scored using three-pass means. These rates are projected onto the diagnostic subset's Bo2-failure counts, with the implied recovery counts rounded to the nearest integer before aggregation. The assisted pool shows consistently broader recovery than Aggregated Context, with the clearest separation on ImgEdit-Bench (UGE). This pattern suggests that combining guidance into one prompt does not fully realize the recovery potential of the distinct reformulation pipelines.

The comparison is descriptive: Aggregated Context is estimated on a different cohort, while the assisted pool reflects success by at least one of three separately executed pipelines. It therefore does not establish a matched-computation advantage or isolate conflicts among instructions from

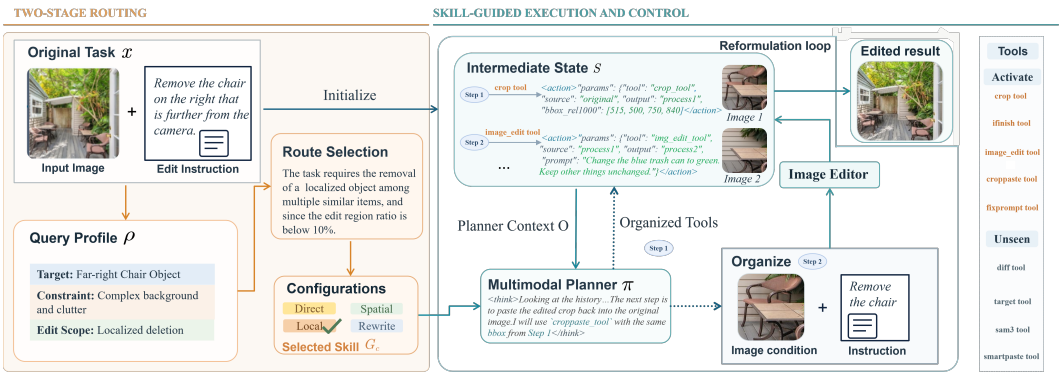

Figure 2: Unified task-reformulation harness. Left: query profiling and route selection activate a configuration-specific skill. Right: the planner uses this skill and the state-derived context to organize existing materials, optionally invoking the fixed editor to generate new ones. The numbered steps illustrate a Local execution trace, not mandatory stages of every reformulation loop.

differences in execution and sampling. Guidance conflicts remain a possible explanation rather than a demonstrated mechanism. Together with the request-dependent benefits and intervention risks, these observations motivate a harness that selects one configuration and its corresponding skill per request, then manages execution through shared controls. The next section describes the framework; Section 5.4 analyzes individual execution controls.

4 Unified Task-Reformulation Harness

Section 3 motivates query-dependent reformulation and selective pipeline guidance. Figure 2 summarizes our harness: two-stage routing selects a configuration and its skill, followed by skill-guided execution over a shared state with optional calls to a fixed editor.

4.1 Unified Formulation of Task Reformulation

Original task. An editing request consists of a source image I_0, an instruction p_0, and optional visual conditions $\mathcal{V}$, jointly written as $x = (I_0, p_0, \mathcal{V})$. The goal is to produce an image that satisfies the requested modification while preserving content that should remain unchanged. The task x remains fixed throughout execution as the reference for evaluating completion.

A shared view of pipelines. The pipelines in Section 3 differ in procedure, but share a common mechanism: they reorganize textual conditions, visual conditions, or editing operations around the same fixed image editor E. We call this mechanism *task reformulation*. A pipeline preserves the user's objective while changing the editor's inputs and how its output is integrated. We unify these procedures through a state that holds their working information and a skill that guides their transformations.

Unified execution state. Different pipelines require different intermediate information: Rewrite uses revised instructions and their history; Local uses crops, image coordinates, and edited regions; Spatial uses masks and placement information for rearrangement and repair. At reformulation iteration t, we represent this information in a common state:

$$S_t = \big(x, I_t^{\text{cur}}, \mathcal{A}_t, H_t^{\text{text}}, \mathcal{Y}_t\big). \tag{2}$$

Here, I_t^{cur} is the current working image; $\mathcal{A}_t$ stores intermediate artifacts such as crops, masks, coordinates, and placement information; H_t^{text} records instructions, actions, tool feedback, and verification outcomes; and $\mathcal{Y}_t$ retains candidate outputs for terminal comparison. Each pipeline reads and updates the components it needs, without requiring a separate state representation. The intermediate state in Figure 2 illustrates a Local trace with tool-action records, crop coordinates, and intermediate images. We initialize $S_0 = \text{Init}(x)$ with $I_0^{\text{cur}} = I_0$.

Pipeline-specific skills. The configurations are $\mathcal{C} = \{c_{\text{Direct}}, c_{\text{Rewrite}}, c_{\text{Local}}, c_{\text{Spatial}}\}$. For a configuration $c \in \mathcal{C}$, its corresponding skill provides guidance G_c on preferred transformations, relevant

tools, and result integration. Direct retains the original editing inputs; Rewrite emphasizes instruction refinement; Local emphasizes crop–edit–paste operations; and Spatial emphasizes segmentation, rearrangement, and repair. Thus, the state unifies the information used by different pipelines, while skills express their distinct ways of reformulating the task. A skill does not mandate a fixed sequence or every associated tool: a multimodal planner π determines concrete actions from the state and feedback.

Skill-guided state transitions. Each reformulation iteration organizes existing instructions and visual artifacts, optionally invoking the editor to generate new material:

$$S_{t+1} = \begin{cases} \mathrm{Organize}(S_t; \pi, G_c), & \text{without an editor call,} \\ \mathrm{Organize}\Big(S_t, E(\widetilde{I}_t, \widetilde{p}_t); \pi, G_c\Big), & \text{with an editor call.} \end{cases} \tag{3}$$

Here, $\mathrm{Organize}$ forms the next state by selecting, transforming, or incorporating materials under the planner and selected skill, using auxiliary tools $\mathcal{T}_{\mathrm{aux}}$. The first branch uses only existing materials, through operations such as instruction refinement, cropping, or paste-back. In the second branch, the planner first prepares the image condition $\widetilde{I}_t$ and instruction $\widetilde{p}_t$ from S_t, then organizes the newly generated output $J_t = E(\widetilde{I}_t, \widetilde{p}_t)$ alongside existing materials and records the associated feedback. In Figure 2, cropping yields Image 1 without an editor call; editing this crop produces Image 2, which is stored for later paste-back without another editor call. Thus, t counts reformulation iterations rather than editor calls. Direct reduces to a single call $E(I_0, p_0)$ whose output is the final result. We next describe skill selection and execution control.

4.2 Task Configuration via Two-Stage Routing

Different query difficulties call for different reformulations, and some requests need no intervention. Our router therefore first characterizes the query and then selects an execution route.

Query profiling. Given the input image–instruction pair, an edit profiler P_{profile} produces a structured query profile $\rho = P_{\mathrm{profile}}(I_0, p_0)$. The profile summarizes the *edit target* (the object or region to manipulate), *key constraints* (such as ambiguity, structural dependencies, or background coupling), and *effective edit scope* (localized or scene-level). It makes factors relevant to potential direct-editing failures explicit for route selection, rather than leaving them implicit in the raw inputs.

Route selection and skill activation. The second stage uses the profile to choose among Direct, Rewrite, Local, and Spatial. The full profiling-and-selection procedure returns $c = r_{\mathrm{2stage}}(x) \in \mathcal{C}$. The harness then loads only the selected configuration's skill guidance G_c. The example in Figure 2 selects Local for a small, localized removal amid similar objects and a complex background. Selection occurs once per request: c and G_c remain fixed, while concrete instructions and tool calls adapt to the state. Configurations specify preferred strategies, not exclusive tool sets; Local and Spatial can also clarify instructions. The interface also accepts supplied configurations, allowing routing and execution to be examined separately.

4.3 Skill-Guided Execution and Control

Shared controls manage the planner's context, auxiliary operations, and termination throughout the state transitions of Equation (3).

Context management before planning. Before each planning decision, the harness projects the stored state S into the planner's visible context O. Omitting iteration indices for brevity, we write:

$$O = \mathrm{Context}(S) = (p_0, I_0, I^{\mathrm{cur}}, \mathcal{V}, H^{\mathrm{text}}). \tag{4}$$

The planner receives O together with G_c: the original task, current image, and textual records are retained, but historical intermediate image attachments are removed. The intermediate images shown inside the state in Figure 2 remain stored artifacts, not historical attachments replayed in O. They remain accessible for tool execution and integration, while candidate images are retained for terminal comparison.

Applicability checks and recovery. During preparation and integration, the harness checks auxiliary operations and provides corrective feedback. For Local, an oversized first crop is invalidated, I_0 is restored, and the planner continues with the full image; repeated text-based segmentation requests trigger guidance toward point-based or difference-based segmentation. This feedback updates the state and subsequent actions without changing the selected configuration or skill.

Verification and termination. When the planner requests completion verification, a vision-language model evaluates the current image against the original task x; a positive verdict returns the result, and otherwise the recorded feedback guides continued execution within a ten-step agent budget counting auxiliary operations as well as editor calls. If completion remains unverified at the limit, the harness makes a terminal direct-editing attempt with the original instruction p_0, and an online selector compares this candidate with retained outputs to choose the final image, without using offline benchmark scores. The terminal attempt is a recovery action, not a new routing or skill-selection decision. Algorithm 1 provides pseudocode.

5 Experiments

We evaluate editing quality, configuration routing, execution controls, transfer across editors and planners, and execution cost. Pipeline-level opportunity is examined in Section 3. Dashes indicate unreported measurements.

5.1 Experimental Setup

Benchmarks and evaluation. We evaluate the Basic-Edit and Understanding-Grounding-Editing (UGE) suites of ImgEdit-Bench (Ye et al., 2025), PICABench (Pu et al., 2025), IV-Edit (Qu et al., 2026), and VIBE (Zhang et al., 2026). All experiments are conducted on the full sets of each benchmark, scored with their respective official evaluation protocols. The judge model is Qwen3.8-27B (Qwen Team, 2026) in non-thinking mode, with hyperparameters set to the official defaults. ImgEdit-Bench and IV-Edit use 1–5 scores; PICABench and VIBE use 0–1 scores, displayed as percentages in this section. IV-Edit reports the Average of four dimensions.

Systems and implementation. Our primary editor is FLUX.2-klein-9B (Black Forest Labs, 2026), hereafter referred to as FLUX.2-klein, and the planner is GPT-5.6 Luna (OpenAI, 2026). The main comparison includes FLUX.2-klein (Direct), EditThinker (Li et al., 2025), RePlan (Qu et al., 2026), GenArtist (Wang et al., 2024), and Nano Banana Pro (Google, 2026b). EditThinker is an arXiv 2025 preprint, RePlan was accepted at ECCV 2026, and GenArtist was accepted as a NeurIPS 2024 Spotlight. Selector-Bo2, which uses a GPT-5.6 Luna selector to choose between two direct samples with randomized candidate order, serves as the reference for the routing analysis. Appendix A documents the remaining implementation identifiers.

5.2 Main Results and Qualitative Analysis

Main comparison. Relative to direct editing with the same editor, the harness yields consistent gains on all five benchmarks, with the largest improvement on VIBE ($44.94\% \rightarrow 74.32\%$). It also exceeds EditThinker, RePlan, and GenArtist on every reported benchmark, with GenArtist trailing on most benchmarks. Compared with Nano Banana Pro, the harness achieves the best score on four of five benchmarks, while Nano Banana Pro remains ahead on VIBE (76.51% versus 74.32%). These results indicate that a 9B primary editor with an external planner approaches the quality of a leading proprietary model. Section 5.6 reports the corresponding cost estimates.

Table 3: Main results (higher is better). PICABench and VIBE are percentages; IV-Edit reports Average.

Method	ImgEdit-Basic	ImgEdit-UGE	PICABench	IV-Edit	VIBE
FLUX.2-klein Direct	3.421	3.052	51.24%	3.056	44.94%
EditThinker	3.662	2.978	57.13%	3.275	41.49%
RePlan	3.055	3.089	48.58%	3.210	33.57%
GenArtist	2.171	2.022	35.74%	2.680	48.75%
Nano Banana Pro	3.627	3.568	57.89%	3.605	**76.51%**
Ours	**3.789**	**3.697**	**59.82%**	**3.725**	74.32%

Table 4: Routing hit rates on the five benchmarks.

Benchmark	n	Hit rate
ImgEdit-Basic	737	74.9%
ImgEdit-UGE	45	**86.7%**
PICABench	900	81.2%
IV-Edit	804	77.2%
VIBE	900	83.6%
TOTAL	**3386**	**79.6%**

Table 5: Ablation results. PICABench and VIBE are percentages; IV-Edit reports Average.

Ablation	ImgEdit-Basic	ImgEdit-UGE	PICABench	IV-Edit	VIBE
Full Harness	**3.789**	**3.697**	**59.82%**	**3.725**	74.32%
Single-stage Router	3.742	3.675	58.74%	3.692	**75.35%**
Aggregated Context	3.554	3.044	58.01%	3.578	61.66%
Full History	3.557	3.433	58.57%	3.602	63.94%
No Terminal Selection	3.522	3.578	54.26%	3.615	64.15%

Qualitative analysis. Figure 3 compares final outputs on four requests involving clothing removal, stand removal, the appearance of bitten pastries, and person relocation. Several baselines leave requested objects in place or introduce inconsistent support relationships, such as a floating plant after stand removal. In the displayed examples, the harness removes the jacket and tie, places the plant on the floor, depicts bitten pastries, and relocates the target person.

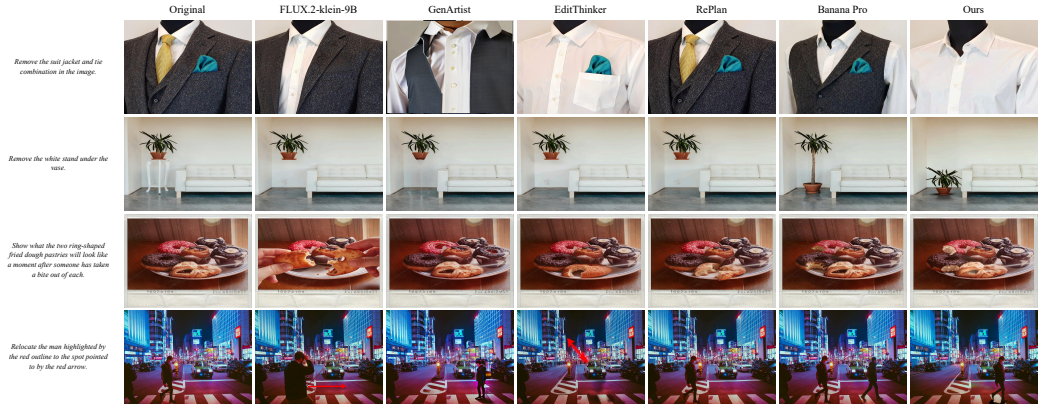

Figure 3: Qualitative comparisons on four editing requests. Instructions are shown on the left; columns show the original image and outputs from FLUX.2-klein (Direct), GenArtist, EditThinker, RePlan, Nano Banana Pro, and our harness.

5.3 Routing Analysis

For task x_i, the router selects $c_i = r(x_i)$ once, and that configuration is retained throughout execution. Let s_i^c be the measured score under configuration c, and let b_i be the score of the output chosen by the online Selector-Bo2 baseline. Over the full set of requests $\mathcal{D}_r$, the hit rate is

$$H_r = \frac{100}{|\mathcal{D}_r|} \sum_{i \in \mathcal{D}_r} \mathbf{1}\left[s_i^{r(x_i)} \geq b_i \right]. \tag{5}$$

The router receives only task inputs; offline scores are used to evaluate its decisions. Thus, H_r measures how often the selected configuration matches or exceeds Selector-Bo2, rather than agreement with a configuration label or an oracle best route. Table 4 reports the captioned router's 79.6% overall hit rate, compared with 76.5% without captions. These results indicate that the selected configuration often meets or exceeds a strong direct-sampling reference. Per-benchmark hit counts for the captioned router are detailed in Appendix B. The corresponding framework-level quality results are reported in Table 5.

5.4 Framework and Execution-Control Ablations

Framework variants. Full Harness is our complete method: two-stage routing with image captions, configuration-specific skills, and execution controls. Single-stage Router removes the captioning stage and classifies the raw request directly, leaving the rest of the harness unchanged. Aggregated Context merges all tool instructions into a single prompt for direct editing, without routing or a

tool-execution loop. Full History and No Terminal Selection are execution-control ablations, defined in the following two paragraphs. Full Harness attains the best score on four of five benchmarks, and removing any component degrades results, with the captioned two-stage routing contributing the largest share of the gap.

Visual context and tool intervention. Full History differs from Full Harness only in the visual context visible to the planner: it attaches every historical intermediate image, whereas Full Harness retains only the source and current images together with textual records (Equation (4)). Compressed context is better than full history on all five benchmarks, most clearly on ImgEdit-UGE (3.697 versus 3.433), indicating that retaining historical intermediate images is unnecessary and can hurt editing quality. A separate ImgEdit-UGE diagnostic improves from 3.522 to 3.632 when crop rollback and segmentation redirection are enabled together (Appendix B).

Terminal output handling. No Terminal Selection disables the terminal candidate comparison and returns the last output directly. It is lower than the compressed-context reference on every benchmark, by up to 0.119 on ImgEdit-UGE, which supports the utility of terminal output selection.

5.5 Generalization across Editors and Planners

Editor transfer replaces FLUX.2-klein with Boogu-Image-0.1-Edit-Turbo (Chen et al., 2026) or a Banana editor, retaining GPT-5.6 Luna and the harness design apart from required API adaptation. Planner transfer keeps FLUX.2-klein fixed and replaces GPT-5.6 Luna with Qwen3.8. Table 6 reports the transfer results on ImgEdit-UGE.

Table 6: Editor and planner transfer on ImgEdit-UGE.

Setting	Score
Editor transfer	
Boogu Direct	3.067
Boogu + Harness	3.484
Banana Direct	3.150
Banana + Harness	**3.778**
Planner transfer	
GPT-5.6 Luna	**3.697**
Qwen3.8	3.618

The harness improves both transfer editors over their direct baselines, with the Banana editor matching the primary setting in absolute score. Replacing the planner with Qwen3.8 remains effective at a modest gap to GPT-5.6 Luna. These results support transfer beyond the primary editor and planner on this benchmark; they do not establish the same gains across all five datasets.

5.6 Cost Analysis

Average per-case costs across the five benchmarks are reported in Table 9 in Appendix C. The API cost of one request comprises planning calls, a one-time captioning-and-routing charge, and API tool calls; the editor and geometric tools run locally. With GPU time monetized, the combined estimates range from USD 0.0068 to 0.0098 per request, roughly $1/14$–$1/20$ of the Nano Banana Pro charge of USD 0.134 per image.

6 Conclusion

We present an image editing harness that organizes textual and visual task reformulation pipelines within a common configuration and execution framework. The pilot identifies request-dependent recovery opportunities and intervention risks, motivating selective configuration and controlled execution. Two-stage routing and skill guidance configure each request, while shared controls manage context, tool use, and recovery around an unchanged primary editor. Across five evaluation sets, the reported system improves over direct editing and online best-of-two selection, achieving competitive quality with a 9B primary editor. Remaining limitations include imperfect routing, reliance on model-based verification, and the additional execution overhead of assisted editing.

AI use statement

We used generative AI tools to assist with manuscript preparation, including revising text for clarity and readability, organizing the related-work discussion, identifying relevant literature, checking

and formatting citations and references, refining figure and table layouts, and adjusting LaTeX typesetting. The authors take full responsibility for the final manuscript, including the accuracy of its scientific claims, references, and reported results.

REFERENCES

Black Forest Labs. FLUX.2 [klein] 9B: Official model card. Hugging Face model card, 2026. URL https://huggingface.co/black-forest-labs/FLUX.2-klein-9B. Accessed September 25, 2026.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. arXiv preprint arXiv:2506.15742, 2025. URL https://arxiv.org/abs/2506.15742.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2211.09800.

Guoxuan Chen, Chufeng Xiao, Haoran Yang, Siyue Xie, Binxiao Huang, Ming Zhang, Cheuk Him Chau, Xinyu Fu, Yingzhao Lian, Tom S. Y. Li, Jintao Lin, Bowen Dong, Zian Qian, Yuhao Liu, Yuxuan Hu, Weikang Shi, Bin Zou, Bowen Zheng, Haoxuan Che, Chang Chen, Yuyang He, Heyang Sun, Tianyu Huang, Chong Hou Choi, Cheng Gong, Han Shi, Haoli Bai, Xihui Liu, Hongsheng Li, Qifeng Chen, Chao Huang, Rui Liu, and Chenyang Lei. Boogu-Image-0.1: Boosting open agentic multimodal generation via understanding under a minimal budget. arXiv preprint arXiv:2607.13125, 2026. URL https://arxiv.org/abs/2607.13125.

Google. Gemini Developer API pricing. Google AI for Developers, 2026a. URL https://ai.google.dev/gemini-api/docs/pricing. Gemini 3 Pro Image standard image-output pricing. Accessed September 26, 2026.

Google. Gemini 3 Pro Image (Nano Banana Pro): Official model documentation. Google AI for Developers, 2026b. URL https://ai.google.dev/gemini-api/docs/models/gemini-3-pro-image. Accessed September 26, 2026.

Yujia Hu, Songhua Liu, Zhenxiong Tan, Xingyi Yang, and Xinchao Wang. Image editing as programs with diffusion models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.04158.

Hongyu Li, Manyuan Zhang, Dian Zheng, Ziyu Guo, Yimeng Jia, Kaituo Feng, Hao Yu, Yexin Liu, Yan Feng, Peng Pei, Xunliang Cai, Linjiang Huang, Hongsheng Li, and Si Liu. EditThinker: Unlocking iterative reasoning for any image editor. arXiv preprint arXiv:2512.05965, 2025. URL https://arxiv.org/abs/2512.05965.

Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, Wenxun Dai, Xinghao Ding, Chunyu Wang, and Qinglin Lu. JarvisEvo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2511.23002.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.17651.

OpenAI. GPT-5.6 Luna: Official model documentation. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed September 26, 2026.

Yuandong Pu, Le Zhuo, Songhao Han, Jinbo Xing, Kaiwen Zhu, Shuo Cao, Bin Fu, Si Liu, Hongsheng Li, Yu Qiao, Wenlong Zhang, Xi Chen, and Yihao Liu. PICABench: How far are we from physically realistic image editing? arXiv preprint arXiv:2510.17681, 2025. URL https://arxiv.org/abs/2510.17681.

Tianyuan Qu, Lei Ke, Xiaohang Zhan, Longxiang Tang, Yuqi Liu, Bohao Peng, Bei Yu, Dong Yu, and Jiaya Jia. RePlan: Reasoning-guided region planning for complex instruction-based image editing. In *European Conference on Computer Vision (ECCV)*, 2026. URL https://arxiv.org/abs/2512.16864.

Qwen Team. Qwen3.8-27B: Official model card. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B. Accessed September 25, 2026.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu Edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.10089.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.11366.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. GenArtist: Multimodal LLM as an agent for unified image generation and editing. In *Advances in Neural Information Processing Systems*, volume 37, pp. 128374–128395. Curran Associates, Inc., 2024. doi: 10.52202/079017-4077. URL https://arxiv.org/abs/2407.05600.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671, 2023. URL https://arxiv.org/abs/2303.04671.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image technical report. arXiv preprint arXiv:2508.02324, 2025. URL https://arxiv.org/abs/2508.02324.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Ruijie Ye, Jiayi Zhang, Zhuoxin Liu, Zihao Zhu, Siyuan Yang, Li Li, Tianfu Fu, Franck Dernoncourt, Yue Zhao, Jiacheng Zhu, Ryan Rossi, Wenhao Chai, and Zhengzhong Tu. Agent Banana: High-fidelity image editing with agentic thinking and tooling. arXiv preprint arXiv:2602.09084, 2026. URL https://arxiv.org/abs/2602.09084.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. ImgEdit: A unified image editing dataset and benchmark. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2505.20275.

Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. AnyEdit: Mastering unified high-quality image editing for any idea. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2411.15738.

Huanyu Zhang, Xuehai Bai, Chengzu Li, Chen Liang, Haochen Tian, Haodong Li, Ruichuan An, Yifan Zhang, Anna Korhonen, Zhang Zhang, Liang Wang, and Tieniu Tan. How well do models follow visual instructions? VIBE: A systematic benchmark for visual instruction-driven image editing. arXiv preprint arXiv:2602.01851, 2026. URL https://arxiv.org/abs/2602.01851.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.10012.

Haozhe Zhao, Xiaojian Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. UltraEdit: Instruction-based fine-grained image editing at scale. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.05282.