GMO-E^{2}DIT: Grounded Multi-Operation Editing for E-Commerce Images

Zipeng Guo[1*], **Xiaoan Liu**[1,2,*], **Lichen Ma**[1,4,*†], **Cheng Wang**[3], **Yu He**[1], **Xiaolong Fu**[1], **Jingling Fu**[1], **Xinyuan Shan**[1], **Shaojie Guo**[1], **Luohang Liu**[1], **Yan Li**[1], **Junshi Huang**[1‡]

[1]JD.com, [2]Wuhan University, [3]Sun Yat-sen University,
[4]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
{guozp8888, malichen2020, junshi.huang}@gmail.com

Abstract

Real-world e-commerce image editing often requires multiple, localized, and auditable operations rather than global restyling. This compositional nature poses a dual challenge: models must precisely apply all requested edits to the correct regions while preserving unmodified content, even under ambiguous instructions. Existing one-shot editors conflate intent resolution, spatial grounding, and synthesis into a single step, frequently resulting in partial execution failures, which is unacceptable for commercial scenarios. To address this, we introduce GMO-E^{2}DIT, an agentic editing framework that couples a Vision-Language Model (VLM) with a mask-conditioned image editor to tackle structured multi-turn task completion. Given an underspecified instruction, the VLM agent constructs a region-grounded edit agenda, effectively decoupling cognitive reasoning from generative rendering. The framework then executes sub-programs via operation-aware masks and references, utilizing a reflection-driven loop to inspect intermediate results and determine the subsequent state. This iterative mechanism reliably preserves safe partial progress, retries unfinished operations, and recovers from errors. Furthermore, we develop a unified data pipeline providing aligned supervision for planning, execution, and reflection, alongside EComEditBench, a comprehensive benchmark for instruction-driven evaluation. Extensive experiments demonstrate that GMO-E^{2}DIT achieves competitive performance compared to strong closed-source models, yielding superior instruction accuracy and edit fidelity over existing baselines.

1 Introduction

Instruction-driven image editing has become a natural interface for visual content creation, allowing models to perform diverse semantic modifications on input images according to free-form user instructions [1–11]. For e-commerce product images, however, mere visual plausibility is insufficient. Many visual elements such as logos, price tags, and decorative patches often carry business significance. As shown in Fig. 1 (a), tasks like "move the right-side yellow sticker to the left-side area", "exchange the bottom-left patch with the bottom-right patch" or "remove the center product sticker and the top-right sticker" require precise, accountable, and auditable local modifications rather than global restyling.

Current one-shot editors struggle to meet these requirements. User instructions are often fuzzy and compositional, referencing ambiguous regions or requesting multiple simultaneous edits. Existing

*Equal contribution.
†Project Lead.
‡Corresponding Author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

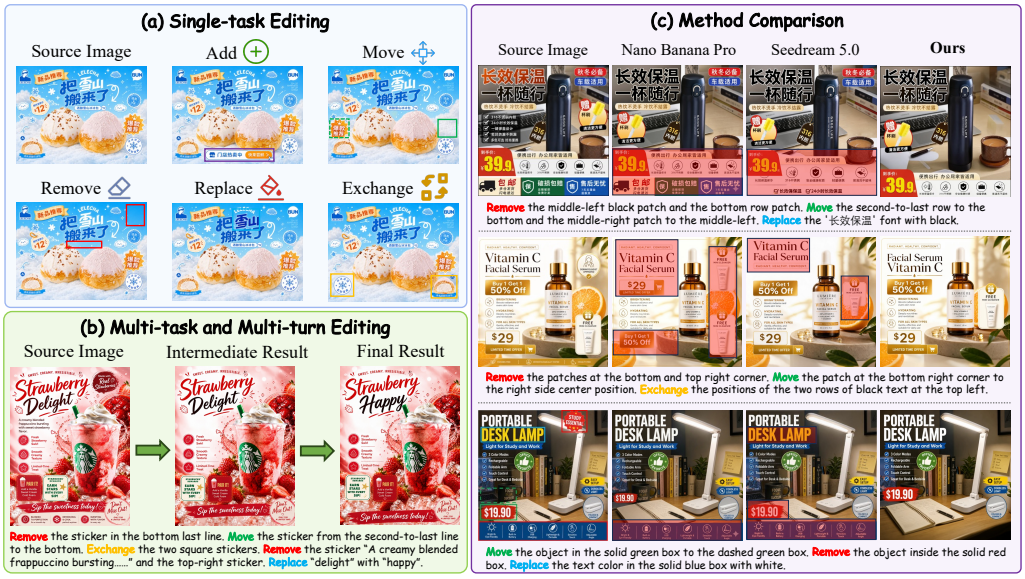

Figure 1: Qualitative demonstration. (a) Single-task editing operations. (b) Complex multi-task and multi-turn editing capabilities. (c) Method comparison demonstrating our approach outperforms strong baselines in executing compositional instructions while preserving unedited regions.

editors typically merge intent resolution, spatial grounding, operation binding, and rendering into a single generation step. Consequently, partial failures are common: some local edits may be correct, while others are missed or erroneously modified. For commercial applications, such partially incorrect outputs are more problematic than incomplete ones. Mask- or box-guided editors[12–18] provide more control, but practical users rarely supply such annotations, and manual markings can interfere with the product image itself. Although reasoning-guided methods using VLMs and MLLMs can decompose instructions and ground edits [19–23], they remain largely auxiliary, operating around a fixed generator.

To better tackle the challenges of multi-task, local-operation editing in e-commerce images, we reformulate the problem as multi-operation task completion. The key difficulty lies not only in interpreting natural-language instructions but also in managing a complete editing session until all fine-grained requirements are fulfilled. To this end, we propose GMO-E^{2}DIT, a framework in which a VLM acts as both a strategic planner and a reflective judge. Specifically, GMO-E^{2}DIT converts underspecified user instructions into a structured edit agenda, explicitly specifying operation types (e.g., *move, add, remove*) and their spatial-semantic constraints, thereby separating cognitive reasoning from generative rendering. This design enables robust multi-turn interaction, where a reflection-driven loop allows the agent to autonomously assess each output and determine the state of each operation, indicating whether it is *success, continue* or *rollback*, and ensuring reliable task decomposition and execution in complex e-commerce scenarios.

To support training and evaluation, we construct a unified e-commerce data pipeline and dataset that provides supervision for planning, execution, and reflection from structured metadata, following the principle that complex visual editing benefits from task-specific data [21, 24, 25]. The pipeline extracts editable elements from product and scene images, generates controllable source-target pairs, and provides aligned supervision for the VLM agent and mask-conditioned editor. We further introduce EComEditBench, a benchmark for fuzzy-instruction editing, where the system infers the full edit agenda, and precise-instruction editing, where annotated regions isolate execution quality. The benchmark covers *add, remove, move, exchange, replace* and mixed multi-operation compositions, evaluating instruction accuracy, region fidelity, and content preservation.

In summary, our contributions are threefold. (1) We formalize e-commerce editing as grounded multi-operation task completion, serving as a paradigm for multi-task, local-operation editing in e-commerce images and providing a practical framework for handling complex, compositional edits. (2) We propose GMO-E^{2}DIT, a VLM-based agent that integrates planning and reflection for multi-turn editing, interacting with a mask-conditioned editor to achieve robust and controllable results. (3) We develop a unified data pipeline and EComEditBench for systematic training and

evaluation, demonstrating substantial improvements in instruction adherence, region fidelity, content preservation, and reliability across multi-operation editing tasks.

2 Related Work

Instruction-based image editing. Instruction-based image editing aims to modify an input image according to natural-language instructions while preserving irrelevant content. Early diffusion-based methods edit images through attention control, region discovery, or instruction-to-image supervision [1, 26, 27]. Later datasets and MLLM-assisted editors improve instruction diversity and semantic understanding [2, 3, 28], while recent unified and in-context models further support open-ended editing, multimodal generation, and reference-conditioned synthesis [4, 5, 29–33]. Beyond general-purpose editing, specialized tasks such as text-centric editing and poster creation pose unique challenges in maintaining glyph fidelity, layout consistency, and entity preservation [24, 34]. E-commerce image editing further requires precise local manipulation of commercial elements, including badges, labels, stickers, and product patches. Accordingly, GMO-E^{2}DIT formulates e-commerce editing as grounded multi-operation execution, where executable edit programs explicitly bind operations, regions, and attributes to guide mask-conditioned local rendering.

Grounded planning for editing. Reliable grounding is essential for complex editing. Open-vocabulary detectors and promptable segmentation models provide region-level priors [35, 36], and recent VLM/MLLM-guided methods use such priors to localize targets, decompose instructions, or inject spatial constraints into diffusion editors [37–39]. For example, MCIE introduces spatial- and background-aware attention for complex instruction editing [21], RePlan employs a VLM planner to produce region-aligned guidance for diffusion execution [19], and GoT formulates semantic-spatial reasoning chains with coordinates for generation and editing [20]. These methods demonstrate the value of planning before synthesis, yet they mainly focus on identifying where to edit. E-commerce editing additionally requires binding operations to roles, e.g., moving a patch from a source to a target, exchanging two elements, or inserting an external reference. GMO-E^{2}DIT addresses this by representing each edit as a typed program entry with source regions, target regions, attributes, and optional patch references, which are compiled into operation-aware masks for execution.

Reward feedback editing. Feedback signals are increasingly used to improve instruction following in image generation and editing. VLM-as-judge protocols provide scalable evaluation, while reward models and RL-based post-training optimize instruction fidelity, visual quality, and content preservation [40–42]. Recent reasoning-centric editors further introduce explicit critique or reflection. EditThinker iteratively critiques editing results and refines prompts for subsequent attempts [22], and ThinkRL-Edit expands RL exploration from denoising to reasoning trajectories through planning, reflection, and checklist-style rewards [23]. Such methods highlight the importance of deliberation, but their feedback is usually prompt-level or globally scored. In contrast, the reflection module in GMO-E^{2}DIT is program-level: it classifies each intermediate result as `success`, `continue`, or `rollback`. This allows the system to keep correct partial edits, retry only missing operations, and discard outputs that damage the product image, enabling reliable multi-operation execution.

3 Method

3.1 Problem definition

Given a source e-commerce image $x \in [0,1]^{H\times W\times 3}$ and a natural-language editing instruction u, our goal is to produce an edited image y that completes all requested local operations while preserving non-target regions. We represent the latent intent as a structured *edit agenda*,

$$\mathcal{U} = \{p_i\}_{i=1}^{K}, \qquad p_i = (\tau_i, b_i^{\mathrm{src}}, b_i^{\mathrm{dst}}, a_i, h_i), \tag{1}$$

where i denotes the task order, τ_i the operation type, b_i^{src} and b_i^{dst} optional source and target boxes, a_i operation-specific attributes, and h_i a short textual hint. The operations include addition, removal, movement, exchange, and local replacements. This role-aware design separates reasoning from rendering: the VLM decides *what* and *where* to edit, while the editor handles localized synthesis.

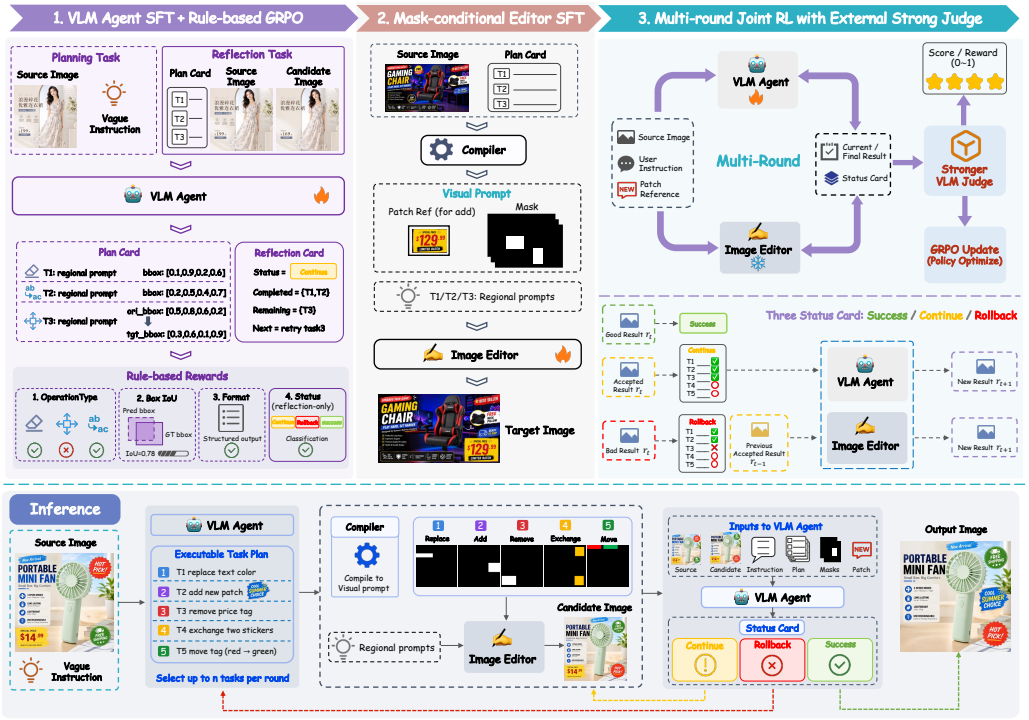

Figure 2: Overview of our framework. Training involves three stages: 1) SFT and rule-based GRPO for the VLM agent; 2) SFT for the mask-conditioned editor; and 3) multi-round joint RL guided by an external VLM judge. During inference, the agent and editor interact in a closed loop, iteratively executing masked visual prompts and refining results until the instruction is complete.

3.2 Agent–Editor Framework

Our framework contains a VLM editing agent $\mathcal{G}_\phi$ and a mask-conditioned image editor $\mathcal{E}_\theta$. The VLM agent works in two modes: *planning* and *reflection*. In the planning mode, it converts the fuzzy instruction u, together with the source image $x \in [0,1]^{H\times W\times 3}$, into a structured edit agenda,

$$\hat{\mathcal{U}}_0 = \mathcal{G}_\phi^{\text{plan}}(x_0, u). \tag{2}$$

At editing round t, a scheduler selects a small sub-program $\mathcal{Q}_t \subseteq \hat{\mathcal{U}}_{t-1}$ from the remaining agenda. A deterministic compiler $\mathcal{C}$ converts $\mathcal{Q}_t$ into a canonical prompt $\tilde{u}_t$, a small set of operation-aware masks $\mathcal{M}_t$, and optional patch references $\mathcal{A}_t$, based on which the intermediate image $\hat{y}_t$ is generated by editor $\mathcal{E}_\theta$,

$$(\tilde{u}_t, \mathcal{M}_t, \mathcal{A}_t) = \mathcal{C}(x_t, u, \mathcal{Q}_t), \qquad \hat{y}_t = \mathcal{E}_\theta(x_t, \tilde{u}_t, \mathcal{M}_t, \mathcal{A}_t). \tag{3}$$

After the editor generates an intermediate result, the same VLM agent switches to reflection mode:

$$(\hat{f}_t, \hat{s}_t) = \mathcal{G}_\phi^{\text{reflect}}(x_t, \hat{y}_t, \mathcal{U}_{t-1}, \mathcal{Q}_t), \qquad \hat{s}_t \in \{\texttt{success}, \texttt{continue}, \texttt{rollback}\}. \tag{4}$$

$$(x_{t+1}, \hat{\mathcal{U}}_t) = \begin{cases} (\hat{y}_t, \varnothing), & \hat{s}_t = \texttt{success}, \\ (\hat{y}_t, \textsc{Remaining}(\hat{f}_t)), & \hat{s}_t = \texttt{continue}, \\ (x_t, \textsc{Replan}(\hat{f}_t, \hat{\mathcal{U}}_{t-1}, u)), & \hat{s}_t = \texttt{rollback}. \end{cases} \tag{5}$$

The reflection output evaluates the result of each editing round and assigns one of three states: `success`, `continue`, or `rollback`. If all tasks are successfully completed (`success`), the system stops, as no further edits are required. If some tasks remain but no critical errors occurred (`continue`), the agent continues with the remaining tasks. If significant errors occur (`rollback`), the system rolls back to the input or last safe image and retries with a revised sub-program. This mechanism ensures multi-turn editing preserves safe progress while preventing errors from propagating.

3.3 Training and Inference

As shown in Fig. 2, our framework trains a VLM agent and a mask-conditioned editor for reliable multi-turn, multi-operation editing. The agent is first trained to generate structured edit agendas and diagnose intermediate states, while the editor learns to execute these programs with explicit visual grounding. In a joint training stage, the agent interacts with the frozen editor and receives rewards for both plan accuracy and reflection, learning when to continue or roll back. At test time, the system iteratively applies the agenda with reflection-guided retries to achieve robust, multi-turn editing.

Stage 1: Supervised and Reinforcement Training of the Agent. We first train the VLM agent $\mathcal{G}_\phi$ with supervised fine-tuning on both planning and reflection data. Planning samples map an image-instruction pair (x, u) to a edit agenda $\mathcal{U}$, while reflection samples map an intermediate editing state $(x_t, \hat{y}_t, \mathcal{U}_{t-1}, \mathcal{Q}_t)$ to a diagnosis f_t and a state label s_t. The SFT objective is

$$\mathcal{L}_{\text{agent-sft}}(\phi) = -\mathbb{E}_{(q,o)\sim\mathcal{D}_{\text{agent}}} \sum_{l=1}^{|o|} \log \pi_\phi(o_l \mid q, o_{<l}). \tag{6}$$

To further improve structured planning and state diagnosis, we apply GRPO to the VLM agent. For each query q, the current policy samples a group of responses $\{o_g\}_{g=1}^{G}$, and the advantage is normalized within the group. The optimization objective follows

$$\mathcal{J}_{\text{GRPO}}(\phi) = \mathbb{E}_{q,\{o_g\}} \frac{1}{G} \sum_{g=1}^{G} \min\left(r_g A_g, \text{clip}(r_g, 1-\epsilon, 1+\epsilon) A_g\right) - \beta D_{\text{KL}}(\pi_\phi \| \pi_{\text{ref}}), \tag{7}$$

where $r_g = \pi_\phi(o_g \mid q)/\pi_{\text{old}}(o_g \mid q)$ and π_{ref} is the reference policy. Notably, we define two groups of verifiable rewards. For planning, the reward evaluates whether the predicted agenda is properly structured, whether the actions follow the annotated order and operation types, and whether the predicted boxes align with the target boxes. For reflection, the reward assesses whether the predicted result f_t is correctly formatted, whether the state label $\hat{s}_t$ corresponds to `success`, `continue`, or `rollback`, and whether the subsequent tasks after the current state match the ground truth.

Stage 2: Training the Mask-Conditioned Editor. The editor $\mathcal{E}_\theta$ is trained to execute localized programs produced by the agent. Each training sample contains the current image x_t, the compiled instruction $\tilde{u}_t$, operation-aware masks $\mathcal{M}_t$, optional patch references $\mathcal{A}_t$, and the target edited image y_t. By providing visual grounding and role information, the editor is relieved from inferring fuzzy hint and can focus entirely on local synthesis. For a flow-matching editor, the training objective is

$$\mathcal{L}_{\text{edit}} = \mathbb{E}\left[\left\| v_\theta(z_t, t, x_t, \tilde{u}_t, \mathcal{M}_t, \mathcal{A}_t) - v^\star \right\|_2^2\right], \tag{8}$$

where z_t is the noisy latent state and $v^\star$ is the target velocity. Diffusion-based editors can use the analogous noise-prediction objective. Since region roles are expressed by masks and patch references, the editor can focus on faithful local rendering, content preservation, and patch consistency.

Stage 3: Joint Reinforcement Training with Editor Feedback. Finally, the agent is trained in the multi-round editing loop while keeping the editor $\mathcal{E}_\theta$ frozen. For each instruction, the policy samples a group of trajectories. In each trajectory, $\mathcal{G}_\phi$ first predicts $\hat{\mathcal{U}}_0$; the scheduler selects $\mathcal{Q}_t$; the frozen editor generates $\hat{y}_t$; and the updated context $(x_t, \hat{y}_t, \mathcal{U}_{t-1}, \mathcal{Q}_t)$ is fed back to the agent for reflection. If $\hat{s}_t =$ `continue`, the next round starts from $\hat{y}_t$ and the remaining tasks in $\hat{f}_t$; if $\hat{s}_t =$ `rollback`, the corrupted result is discarded and the next round restarts from the previous safe image.

The closed-loop reward integrates verifiable reward with model-based evaluation. During planning turns, the agent is rewarded for maintaining the correct structure, action order and types, and bounding-box alignment as established in Stage 1. During reflection turns, a stronger VLM judge evaluates both the edited image and the subsequent tasks selected by the agent. Notably, premature `success` decisions on incomplete or incorrect images are heavily penalized to prevent early termination. Rewards are assigned to the token spans responsible for generating the corresponding planning or reflection outputs, and the sampled trajectories are optimized using the GRPO objective. This stage trains the agent not only to generate accurate plans but also to determine when it is safe to continue from an intermediate result and when to roll back, thereby enabling reliable multi-turn execution.

Inference. At test time, the same loop is run for at most T rounds. Starting from $x_0 = x$, the system repeatedly updates $(x_{t+1}, \hat{\mathcal{U}}_t)$ according to Eq. (5). Execution terminates when $\hat{s}_t =$ `success`.

Otherwise, it continues from $\hat{y}_t$ for recoverable unfinished operations or rolls back to x_t for harmful failures. In this way, the structured agenda provides multi-operation capability, and reflection-guided retry provides multi-turn reliability.

4 Dataset Construction and Benchmark

To enhance complex instruction following, we propose a data synthesis pipeline focused on compositional operations, spatial grounding, and iterative feedback. Moving beyond isolated edits, our approach simulates multifaceted e-commerce workflows by integrating tasks such as patch insertion, region relocation, and attribute modification (e.g., text, color, and style). This provides rigorous supervision for the model's spatial localization, generative fidelity, and instruction adherence.

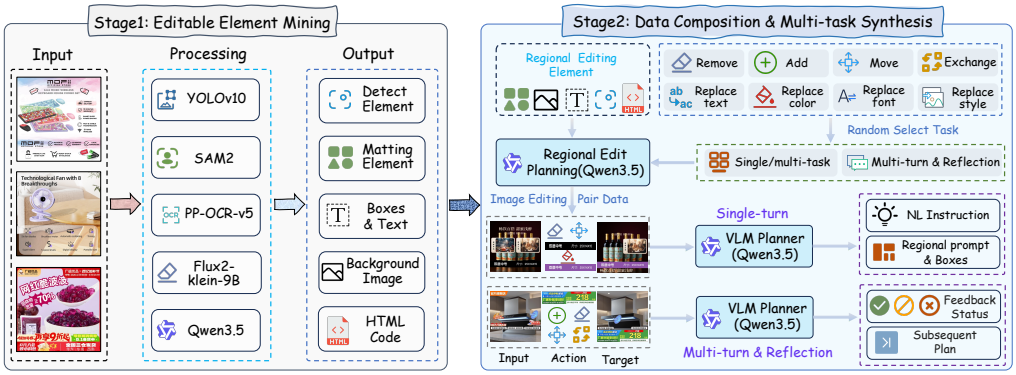

Figure 3: Overview of the data synthesis pipeline. Stage 1 extracts structured visual and textual assets, while Stage 2 recomposes them to synthesize grounded multi-task and multi-turn training pairs.

4.1 Editable Element Mining

As illustrated in Fig. 3, the pipeline accepts a diverse array of e-commerce posters as input. These images are characterized by intricate compositional layouts, integrating multi-layered visual and textual components. We utilize a sophisticated multi-stage processing pipeline to decompose each input image into its constituent semantic layers and structure. The pipeline generates a comprehensive suite of aligned, editable representations for each input image. This includes pixel-level masks and spatial bounding boxes, high-fidelity matted assets (e.g., products and icons), structured textual metadata with precise coordinates, a semantically-consistent inpainted background, and a hierarchical HTML representation that captures the poster's underlying layout and semantic logic.

4.2 Data Composition And Multi-task Synthesis

Unified Synthesis of Image Editing Data. We synthesize training samples by integrating three core components: the raw product image, a generatively-inpainted background, and structured elements extracted via our perception pipeline. Specifically, we localize semantic components—such as stickers and text blocks—using object detection, followed by alpha-matting to isolate them as high-fidelity, transparent assets. Each asset is stored as a structured tuple encompassing its semantic category, bounding box, and pixel-level mask.

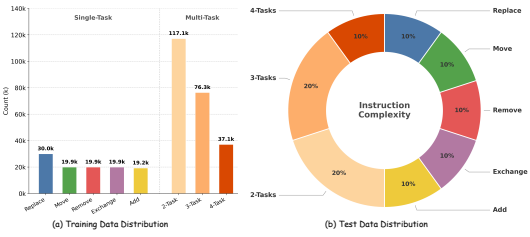

Figure 4: Overview of training and test data. (a) Sample counts for single-task and multi-task instructions in training. (b) Evaluation set composition by instruction complexity, highlighting benchmark diversity.

To bridge these discrete assets with complex editing intents, we introduce a dual-mode manipulation strategy. While geometric operations (e.g., addition or relocation) are applied directly, local replacement requires modifying a region's semantic content or style while preserving its spatial and layout integrity. To ensure controllability and interpretability, we adopt HTML as a structured intermediate representation for replacement, effectively decoupling style attributes from spatial constraints.

Finally, these elements are programmatically recomposed based on edit programs orchestrated by the Regional Edit Planner (Qwen3.5), yielding precisely aligned source-target pairs for grounded e-commerce editing.

Synthesis of Structured Data for Multi-task Planning. Leveraging the visual differences between source and target pairs, we utilize inverse reasoning to generate training triplets for the VLM planner. Each triplet consists of a natural language instruction, a grounded region mask, and a spatial prompt. We further expand the scope of our planning data by incorporating multi-task regional editing operations, such as *addition*, *removal*, *movement*, *exchange*, and *replacement*. By mapping high-level linguistic directives to these fine-grained spatial constraints, our pipeline establishes a unified data contract that facilitates robust, multi-operation image editing.

Synthesis of Iterative Data for Multi-turn Reflection. Solely training for image generation is insufficient for stable multi-turn editing, as models may partially execute tasks or produce spatial errors. Thus, we construct a reflection dataset to train a feedback model for state diagnosis and decision-making. We categorize feedback into three states: (1) *Success*: All tasks are successfully completed; (2) *Continue*: Tasks are partially finished without fatal errors, allowing the system to continue with the remaining objectives; and (3) *Rollback*: Significant errors (e.g., misplacement or scaling issues) occur, triggering a rollback to the previous input and task decomposition for a retry.

4.3 Benchmark

Task Setting. We introduce EComEditBench, a held-out benchmark for grounded multi-operation e-commerce editing. Following recent task-specific benchmark design in editing and long-context generation [25, 19, 34], it evaluates not only whether an output looks plausible, but whether all requested local edits are executed on the correct regions while the rest of the product remains intact. Fig. 4 showcases the benchmark's composition, which consists of 1,000 samples distributed across five atomic operations: `add`, `remove`, `move`, `exchange`, and `replace`. The benchmark is balanced between 500 single-operation and 500 multi-operation cases, enabling evaluation of both primitive local edits and compositional editing under multiple region constraints. Additional details are provided in Appendix C.

Instruction protocols. We evaluate models under fuzzy and precise instruction protocols. The fuzzy protocol provides only the source image and a natural-language request, testing end-to-end instruction understanding, operation binding, and spatial grounding. The precise protocol provides structured region-level edit descriptions, optionally with insertion patches, to isolate localized editing once grounding is given.

VLM-based scoring. Following recent benchmarks, we use a fixed VLM-as-a-Judge protocol that evaluates the source image, edited result, and optional references, and assigns three integer scores in $[1, 10]$. **Instruction Accuracy (IA)** measures whether all requested edits are applied to the correct objects and locations. **Edit Fidelity (EF)** evaluates the visual quality and semantic correctness of edited regions. **Background Preservation (BP)** measures whether non-target regions remain unchanged. We report both fine-grained task-group scores and overall averages.

5 Experiment

Experimental Setup. We train GMO-E^{2}DIT on our constructed e-commerce editing dataset with supervision for region planning, mask-conditioned editing, and visual reflection. We evaluate on EComEditBench using fuzzy and precise instruction protocols, which respectively test end-to-end instruction understanding and localized editing execution. Following VLM-as-judge practice [40], we use Gemini-3 Pro to evaluate each output using the source image, instruction, model result, and optional region or patch references. More details can be found in the appendix D.

5.1 Experimental Results

Quantitative Results. Table 1 reports the main comparison under fuzzy- and precise-instruction settings. In the fuzzy setting, GMO-E^{2}DIT achieves the best overall IA and EF scores (8.66 and 8.25) and the second-best BP score (8.87), substantially outperforming open-source baselines such as Qwen-Image-Edit-2511 ($4.52/4.12/4.91$) and FLUX2-klein-base-9B ($5.16/4.53/6.65$) in IA/EF/BP.

Table 1: Quantitative results on EComEditBench under fuzzy-instruction and precise-instruction settings. Scores are on a 1–10 scale; higher is better. **Bold** denotes the best and underlined denotes the second-best result.

	Method	Single Task																Multi Task									Overall		
		Add			Replace			Remove			Move			Exchange			2 Tasks			3 Tasks			4 Tasks					
		IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP	IA	EF	BP
Fuzzy	**Closed-source methods**																												
	Nano Banana 2	**9.69**	9.55	9.75	9.04	9.01	9.57	**9.67**	**9.52**	8.48	8.21	7.80	8.87	7.61	7.15	8.59	8.30	7.96	8.59	**8.27**	**7.99**	**8.72**	7.49	7.20	8.22	8.48	8.21	8.81
	Nano Banana Pro	**9.69**	**9.73**	9.93	9.19	9.14	9.80	9.32	9.16	**9.08**	7.93	7.61	**9.18**	7.30	6.89	8.81	7.95	7.71	**9.05**	7.55	7.33	8.37	7.19	7.18	**8.82**	8.16	7.98	**9.04**
	Seedream 5.0	9.22	8.80	8.46	**9.20**	9.14	9.49	9.52	9.24	8.39	8.82	**8.45**	8.39	8.07	7.51	8.74	8.34	7.88	8.34	7.88	7.45	7.67	7.68	7.30	7.47	8.49	8.11	8.30
	Wan2.7-Image-Pro	9.64	9.50	9.77	9.18	**9.15**	9.64	9.48	9.29	8.71	7.22	6.88	8.52	3.74	3.61	8.25	6.88	6.65	8.55	6.21	6.04	8.34	4.98	4.92	7.94	7.04	6.87	8.66
	Open-source methods																												
	FireRed-Image-Edit-1.1	1.35	1.11	1.64	6.82	5.89	4.67	8.08	7.83	4.08	4.65	3.73	3.06	2.06	1.84	2.20	2.34	1.91	1.63	1.91	1.65	1.46	1.25	1.14	1.18	3.27	2.87	2.30
	Qwen-Image-Edit-2511	4.88	3.85	5.29	8.09	8.14	7.63	8.42	8.07	6.03	4.90	4.49	5.17	2.41	2.21	4.48	3.87	3.43	4.04	3.21	2.75	4.33	2.31	2.12	3.72	4.52	4.12	4.91
	FLUX2-klein-base-9B	7.61	5.40	8.17	7.34	7.10	8.38	8.97	8.79	8.20	5.49	4.49	7.80	2.02	1.91	6.78	4.68	3.97	6.01	4.16	3.70	5.58	2.51	2.24	3.96	5.16	4.53	6.65
	GMO-E^{2}DIT (ours)	9.62	9.31	**9.95**	9.04	9.00	**9.88**	8.96	8.83	9.06	**8.87**	8.14	8.56	**9.07**	**8.54**	**9.30**	**8.67**	**8.13**	8.74	8.00	7.47	8.12	**7.75**	**7.44**	8.06	**8.66**	**8.25**	8.87
Precise	**Closed-source methods**																												
	Nano Banana 2	9.82	9.72	**9.92**	9.09	9.01	**9.92**	9.54	9.45	9.19	8.50	8.32	9.17	9.32	9.06	9.59	8.86	8.63	9.27	9.30	9.10	9.44	9.07	8.95	9.19	9.17	9.00	9.44
	Nano Banana Pro	9.84	9.79	9.83	9.16	8.98	9.46	9.47	9.43	9.45	8.17	7.93	9.27	9.26	8.98	9.47	9.10	8.91	9.55	9.50	9.28	9.72	8.71	8.58	8.87	9.18	9.01	9.49
	Seedream 5.0	9.74	9.46	9.68	8.65	8.46	9.44	**9.82**	**9.71**	9.57	8.79	8.50	9.15	9.02	8.80	9.53	9.33	9.05	9.56	9.24	8.91	9.31	9.75	9.51	9.73	9.29	9.04	9.49
	Wan2.7-Image-Pro	9.18	8.99	9.19	8.78	8.40	9.26	8.86	8.85	8.58	7.31	7.10	9.14	7.40	7.24	8.79	8.53	8.32	9.29	8.39	8.17	9.04	8.66	8.58	9.15	8.40	8.21	9.07
	Open-source methods																												
	FireRed-Image-Edit-1.1	1.18	1.09	1.47	4.78	4.34	2.27	6.58	6.56	2.33	1.86	1.77	1.54	1.54	1.44	1.26	1.81	1.68	1.64	1.23	1.21	1.14	1.00	1.00	1.00	2.30	2.20	1.54
	Qwen-Image-Edit-2511	3.00	2.41	5.16	7.99	7.77	6.91	8.14	7.90	5.41	7.23	6.75	7.87	5.16	4.42	5.67	5.01	4.54	5.26	4.48	4.17	4.93	2.91	2.81	3.38	5.34	4.95	5.48
	FLUX2-klein-base-9B	5.84	4.75	6.32	6.40	6.32	5.68	8.25	8.18	6.94	4.52	4.40	9.11	4.06	3.78	5.19	5.58	5.10	6.66	5.97	5.57	7.02	4.35	4.26	4.64	5.65	5.30	6.52
	GMO-E^{2}DIT (ours)	**9.88**	**9.91**	9.89	**9.19**	**9.19**	9.65	**9.82**	9.70	**9.61**	**9.96**	**9.72**	**9.78**	**9.92**	**9.73**	**9.95**	**9.78**	**9.51**	**9.89**	**9.91**	**9.63**	**9.80**	**9.92**	**9.67**	**9.75**	**9.82**	**9.62**	**9.85**



Figure 5: Qualitative comparison on image editing tasks. Highlighted regions indicate areas where the editing instruction is not fully satisfied. Our method better follows the target edit while preserving irrelevant image regions.

The gains are especially clear on role-sensitive operations, including `move` (8.87/8.14/8.56) and `exchange` (9.07/8.54/9.30), suggesting that explicit agenda planning and operation-aware masks reduce source–target confusion. GMO-E^{2}DIT also leads IA and EF on compositional two- and four-operation cases. Although closed-source models such as Nano Banana Pro better preserve backgrounds, they still lag behind GMO-E^{2}DIT in instruction completion and edit fidelity.

In the precise setting, where region ambiguity is largely removed, GMO-E^{2}DIT further improves to 9.82/9.62/9.85 overall in IA/EF/BP. It outperforms strong closed-source baselines including Nano Banana Pro (9.18/9.01/9.49) and Seedream 5.0 (9.29/9.04/9.49), with consistent gains across operation types. The advantage remains pronounced on source–target-sensitive edits such as `move` (9.96/9.72/9.78) and `exchange` (9.92/9.73/9.95), and performance stays high as the number of requested operations increases. These results validate the separation of fuzzy instruction understanding from grounded local execution.

Qualitative Results. Fig. 5 shows representative fuzzy and precise editing examples. Existing editors can often generate visually plausible e-commerce posters, but they remain brittle when a request

Table 2: Ablation study of direct editing baselines and our staged agent–editor pipeline. Best and second-best results are highlighted in **bold** and underlined.

Method	Training Stage	Single-task			Multi-task			Overall		
		IA	EF	BP	IA	EF	BP	IA	EF	BP
w/o VLM agent										
Baseline image editor (w/o training)	-	6.286	5.538	7.866	3.783	3.303	5.183	5.160	4.530	6.650
Baseline image editor + fuzzy-instruction SFT	-	8.540	8.260	8.884	7.070	6.728	7.383	7.885	7.557	8.156
w/ VLM agent and mask-conditioned editor										
VLM agent (w/o training)	Stage 2	7.946	7.612	8.855	6.017	5.540	7.488	6.883	6.463	8.082
VLM agent SFT	Stages 1–2	9.109	8.734	**9.387**	7.884	7.397	8.146	8.528	8.079	8.768
VLM agent SFT + GRPO	Stages 1–2	**9.126**	**8.814**	9.338	7.973	7.482	8.195	8.618	8.200	8.801
VLM agent SFT + GRPO + joint agent–editor RL	Stages 1–3	9.112	8.764	9.382	**8.137**	**7.682**	**8.308**	**8.664**	**8.247**	**8.871**

contains multiple region-specific operations. Under fuzzy prompts, competing methods frequently complete only part of the request, confuse source and target roles in `move` or `exchange`, or introduce undesired artifacts such as visible boxes. Under precise prompts, where region information is already available, several baselines still miss local removals or replacements and sometimes alter non-target product or background regions. In contrast, GMO-E^{2}DIT decomposes each request into a grounded edit agenda and executes the sub-operations with operation-aware masks, leading to more complete fulfillment of `add`, `remove`, `move`, `exchange`, and `replace` instructions while better preserving the original layout, product appearance, and irrelevant background content.

User Study. We conduct a head-to-head human evaluation on 1,000 fuzzy-instruction cases against Nano Banana Pro, Seedream 5.0, and Wan2.7-Image-Pro. As shown in Fig. 6, GMO-E^{2}DIT is consistently preferred for instruction following (36.3%, 40.4%, and 46.6% vs. 28.5%, 26.6%, and 21.2%) and also slightly favored for consistency (27.5%, 36.6%, and 26.3% vs. 26.5%, 25.0%, and 20.5%). This demonstrates that our framework can effectively adhere to multiple task instructions while maintaining high visual consistency throughout multi-task editing processes.

5.2 Ablation Study

Table 2 studies the contribution of the staged agent–editor pipeline in GMO-E^{2}DIT. The upper block evaluates direct image editing without a VLM agent. The baseline image editor performs reasonably on single-operation edits but drops sharply on multi-operation cases, with only 3.783 IA and 3.303 EF. Fuzzy-instruction SFT substantially improves the overall scores from 5.160/4.530/6.650 to 7.885/7.557/8.156 in IA/EF/BP, showing that task-specific supervision is important for localized editing and background preservation. The lower block further introduces VLM-agent planning with our mask-conditioned editor. Even without agent training, the VLM agent improves compositional handling, reaching 6.017 IA and 5.540 EF on multi-operation cases. After SFT, the agent raises the overall score to 8.528/8.079/8.768, confirming that learned planning and grounding are central to reliable multi-operation editing. Adding GRPO yields the best single-task IA and EF (9.126 and 8.814), while joint agent–editor RL achieves the strongest multi-task and overall results, reaching 8.137/7.682/8.308 on multi-operation cases and 8.664/8.247/8.871 overall. This indicates that GRPO improves local decision quality, whereas joint training better aligns the planner and editor for compositional execution.

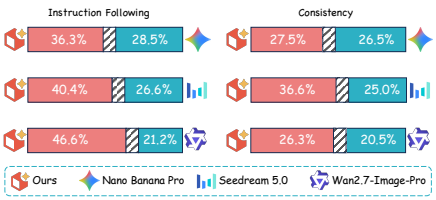

Figure 6: User study between our method and closed-source models regarding Instruction Following and Consistency.

To evaluate the reflection-driven multi-turn mechanism, we compare different strategies in Table 3. Relative to the Stage 1 one-turn baseline, Stage 3 one-turn inference improves IA and EF from 6.89/5.98 to 7.04/6.23, highlighting the benefit of the jointly optimized agent-editor policy. Using a fixed-turn strategy further increases IA/EF to 7.21/6.37, showing that additional execution rounds help recover incomplete operations. Our full reflection-based policy achieves the highest scores across all metrics, with IA, EF, and BP, confirming that adaptive state diagnosis, which allows

Table 3: Ablation study on multi-turn editing.

Model	IA	EF	BP
Stage1-one-turn	6.89	5.98	7.86
Stage3-one-turn	7.04	6.23	8.14
Stage3-fixed-turn	7.21	6.37	8.08
Stage3-reflection	**7.29**	**6.47**	**8.42**

the agent to decide whether to stop, continue, or roll back, is more robust than using a fixed number of refinement rounds. More details are presented in Appendix E.1.

6 Conclusion

In this paper, we have presented GMO-E^{2}DIT, a novel agentic framework that redefines e-commerce image editing from a single-step generation task into a systematic, multi-operation task completion process. By decoupling cognitive reasoning from generative execution, our framework leverages a VLM-based agent to translate fuzzy user instructions into structured agendas and manages the editing workflow through a self-reflective loop. This design effectively addresses the brittleness of existing one-shot editors when faced with complex, compositional requests typical of real-world e-commerce scenarios. Furthermore, we introduced a unified data pipeline and the EComEditBench benchmark to facilitate the training and evaluation of grounded, multi-turn editing. Our experimental results demonstrate that GMO-E^{2}DIT consistently outperforms both state-of-the-art open-source and proprietary models, particularly in terms of instruction adherence, regional fidelity, and long-term reliability through its rollback-and-retry mechanism.

References

[1] 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*, pages 18392–18402, 2023.

[2] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *International Conference on Learning Representations*, 2024.

[3] 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.

[4] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[5] Black Forest Labs. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[6] Yingjing Xu, Jie Kong, Jiazhi Wang, Xiao Pan, Bo Lin, and Qiang Liu. Insightedit: Towards better instruction following for image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2694–2703, 2025.

[7] Yiran Zhao, Yaoqi Ye, Xiang Liu, Michael Qizhe Shieh, and Trung Bui. Imageedit-r1: Boosting multi-agent image editing via reinforcement learning. *arXiv preprint arXiv:2603.08059*, 2026.

[8] Shichao Ma, Yunhe Guo, Jiahao Su, Qihe Huang, Zhengyang Zhou, and Yang Wang. Talk2image: A multi-agent system for multi-turn image generation and editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32437–32445, 2026.

[9] Zongjian Li, Zheyuan Liu, Qihui Zhang, Bin Lin, Feize Wu, Shenghai Yuan, Zhiyuan Yan, Yang Ye, Wangbo Yu, Yuwei Niu, et al. Uniworld-v2: Reinforce image editing with diffusion negative-aware finetuning and mllm implicit feedback. *arXiv preprint arXiv:2510.16888*, 2025.

[10] Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. Enabling instructional image editing with in-context generation in large scale diffusion transformer. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[11] Shiyi Zhang, Yiji Cheng, Tiankai Hang, Zijin Yin, Runze He, Yu Xu, Wenxun Dai, Yunlong Lin, Chunyu Wang, Qinglin Lu, et al. Meta-cot: Enhancing granularity and generalization in image editing. *arXiv preprint arXiv:2604.24625*, 2026.

[12] Lichen Ma, Tiezhu Yue, Pei Fu, Yujie Zhong, Kai Zhou, Xiaoming Wei, and Jie Hu. Chargen: High accurate character-level visual text generation model with multimodal encoder. *arXiv preprint arXiv:2412.17225*, 2024.

[13] Xin Yu, Tianyu Wang, Soo Ye Kim, Paul Guerrero, Xi Chen, Qing Liu, Zhe Lin, and Xiaojuan Qi. Objectmover: Generative object movement with video prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17682–17691, 2025.

[14] Yikai Wang, Chenjie Cao, Junqiu Yu, Ke Fan, Xiangyang Xue, and Yanwei Fu. Towards enhanced image inpainting: Mitigating unwanted object insertion and preserving color consistency. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23237–23248, 2025.

[15] Xuan Ju, Xian Liu, Xintao Wang, Yuxuan Bian, Ying Shan, and Qiang Xu. Brushnet: A plug-and-play image inpainting model with decomposed dual-branch diffusion. In *European Conference on Computer Vision*, pages 150–168. Springer, 2024.

[16] Liangbin Xie, Daniil Pakhomov, Zhonghao Wang, Zongze Wu, Ziyan Chen, Yuqian Zhou, Haitian Zheng, Zhifei Zhang, Zhe Lin, Jiantao Zhou, et al. Turbofill: adapting few-step text-to-image model for fast image inpainting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7613–7622, 2025.

[17] Zipeng Guo, Lichen Ma, Xiaolong Fu, Gaojing Zhou, Lan Yang, Yuchen Zhou, Linkai Liu, Yu He, Ximan Liu, Shiping Dong, et al. Repainter: Empowering e-commerce object removal via spatial-matting reinforcement learning. *arXiv preprint arXiv:2510.07721*, 2025.

[18] Lichen Ma, Xiaolong Fu, Gaojing Zhou, Zipeng Guo, Ting Zhu, Yichun Liu, Yu Shi, Jason Li, and Junshi Huang. Um-text: A unified multimodal model for image understanding. *arXiv preprint arXiv:2601.08321*, 2026.

[19] 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. *arXiv preprint arXiv:2512.16864*, 2025.

[20] Rongyao Fang, Chengqi Duan, Kun Wang, Linjiang Huang, Hao Li, Shilin Yan, Hao Tian, Xingyu Zeng, Rui Zhao, Jifeng Dai, Xihui Liu, and Hongsheng Li. Got: Unleashing reasoning capability of multimodal large language model for visual generation and editing. *arXiv preprint arXiv:2503.10639*, 2025.

[21] Xuehai Bai, Xiaoling Gu, Akide Liu, Hangjie Yuan, YiFan Zhang, and Jack Ma. Mcie: Multimodal llm-driven complex instruction image editing with spatial guidance. *arXiv preprint arXiv:2602.07993*, 2026.

[22] 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.

[23] Hengjia Li, Liming Jiang, Qing Yan, Yizhi Song, Hao Kang, Zichuan Liu, Xin Lu, Boxi Wu, and Deng Cai. Thinkrl-edit: Thinking in reinforcement learning for reasoning-centric image editing. *arXiv preprint arXiv:2601.03467*, 2026.

[24] Sixiang Chen, Jianyu Lai, Jialin Gao, Hengyu Shi, Zhongying Liu, Tian Ye, Junfeng Luo, Xiaoming Wei, and Lei Zhu. Posteromni: Generalized artistic poster creation via task distillation and unified reward feedback. *arXiv preprint arXiv:2602.12127*, 2026.

[25] Zhekai Chen, Yuqing Wang, Manyuan Zhang, and Xihui Liu. Macro: Advancing multi-reference image generation with structured long-context data. *arXiv preprint arXiv:2603.25319*, 2026.

[26] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

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

[28] 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.

[29] Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. In *Advances in Neural Information Processing Systems*, 2025.

[30] Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

[31] Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, Ze Liu, Ziyi Xia, Chaofan Li, Haoge Deng, Jiahao Wang, Kun Luo, Bo Zhang, Defu Lian, Xinlong Wang, Zhongyuan Wang, Tiejun Huang, and Zheng Liu. Omnigen2: Towards instruction-aligned multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025.

[32] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[33] Black Forest Labs. FLUX.2: Next generation image generation. `https://bfl.ai/models/flux-2`, 2025. Accessed: 2026-04-24.

[34] Hui Zhang, Juntao Liu, Zongkai Liu, Liqiang Niu, Fandong Meng, Zuxuan Wu, and Yu-Gang Jiang. Weedit: A dataset, benchmark and glyph-guided framework for text-centric image editing. *arXiv preprint arXiv:2603.11593*, 2026.

[35] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, 2024.

[36] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[37] Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and Ying Shan. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[38] Jun Zhou, Jiahao Li, Zunnan Xu, Hanhui Li, Yiji Cheng, Fa-Ting Hong, Qin Lin, Qinglin Lu, and Xiaodan Liang. Fireedit: Fine-grained instruction-based image editing via region-aware vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[39] Chun-Hsiao Yeh, Yilin Wang, Nanxuan Zhao, Richard Zhang, Yuheng Li, Yi Ma, and Krishna Kumar Singh. Beyond simple edits: X-planner for complex instruction-based image editing. *arXiv preprint arXiv:2507.05259*, 2025.

[40] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023.

[41] Keming Wu, Sicong Jiang, Max Ku, Ping Nie, Minghao Liu, and Wenhu Chen. Editreward: A human-aligned reward model for instruction-guided image editing. *arXiv preprint arXiv:2509.26346*, 2025.

[42] Xiangyu Zhao, Peiyuan Zhang, Junming Lin, Tianhao Liang, Yuchen Duan, Shengyuan Ding, Changyao Tian, Yuhang Zang, Junchi Yan, and Xue Yang. Trust your critic: Robust reward modeling and reinforcement learning for faithful image editing and generation. *arXiv preprint arXiv:2603.12247*, 2026.

[43] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *Advances in neural information processing systems*, 37:107984–108011, 2024.

[44] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[45] Cheng Cui, Yubo Zhang, Ting Sun, Xueqing Wang, Hongen Liu, Manhui Lin, Yue Zhang, Tingquan Gao, Changda Zhou, Jiaxuan Liu, et al. Pp-ocrv5: A specialized 5m-parameter model rivaling billion-parameter vision-language models on ocr tasks. *arXiv preprint arXiv:2603.24373*, 2026.

[46] Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

[47] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Appendix

In this work, we propose GMO-E^{2}DIT, an agentic framework for grounded multi-operation e-commerce image editing. Due to space constraints in the main manuscript, several technical details, data construction processes, and additional qualitative results were omitted. The purpose of this supplementary document is to provide these comprehensive details to facilitate a deeper understanding and reproducibility of our work. Specifically, this material provides the following contents:

- **Method Details (Section A):** Detailed explanations and mathematical formulations of the reward designs for both the initial planning and multi-turn joint reinforcement learning stages;
- **Data Pipeline Details (Section 2):** Comprehensive technical details of the multi-stage perception and generation pipeline utilized for editable element mining and structural synthesis;
- **Benchmark Details (Section C):** Elaboration on the task settings, precise and fuzzy evaluation protocols, and VLM-based scoring criteria of the constructed benchmark;
- **Experimental Details (Section D):** Additional training hyperparameters, optimization strategies, and baseline configurations for both the VLM agent and the mask-conditioned editor;
- **More Experimental Results (Section E):** Extended qualitative comparisons demonstrating the advantages of multi-turn instruction execution over single-turn baselines;
- **Limitations (Section F):** A discussion of the current boundaries and limitations of our proposed framework.

A Method Details

A.1 Stage 1 Reward Design: Planning and Reflection Rewards

During Stage 1, the agent acquires fundamental planning and diagnostic capabilities through supervised fine-tuning (SFT) and initial reinforcement learning. The reward signals during this phase are applied at a single-turn level.

Planning Rewards (R_{plan}) comprise four verifiable metrics designed to evaluate the generated edit agenda:

$$R_{\text{plan}} = w_1 R_{\text{JSON}} + w_2 R_{\text{order}} + w_3 R_{\text{type}} + w_4 R_{\text{bbox}} \tag{9}$$

where:

- R_{JSON} verifies that the generated edit agenda is strictly parsable and accurately populates all requisite structural fields.
- R_{order} evaluates the logical sequencing of operations, heavily penalizing illogical temporal sequences (e.g., executing an `add` operation before a prerequisite `remove` operation).
- R_{type} assesses the accuracy of the predicted operation types (e.g., `move`, `add`, `remove`) against the ground truth.
- R_{bbox} measures the spatial alignment of the predicted source and target bounding boxes. This is quantified using the Intersection over Union (IoU) between the predicted bounding box B_p and the ground-truth box B_{gt}:

$$\text{IoU}(B_p, B_{gt}) = \frac{\text{Area}(B_p \cap B_{gt})}{\text{Area}(B_p \cup B_{gt})} \tag{10}$$

The coefficients w_1, w_2, w_3, and w_4 are hyperparameters that balance the contribution of each reward component. The default weights are set as $w_1 = 0.3$, $w_2 = 0.2$, $w_3 = 0.2$, and $w_4 = 0.3$.

Reflection Rewards (R_{reflect}) evaluate the agent's diagnostic phase using three task-oriented signals:

$$R_{\text{reflect}} = w_5 R_{\text{format}} + w_6 R_{\text{status}} + w_7 R_{\text{next}} \tag{11}$$

where:

- R_{format} ensures that the structured diagnostic reflection strictly adheres to the required JSON schema.
- R_{status} evaluates the accuracy of the predicted intermediate state classification (`success`, `continue`, or `rollback`).
- R_{next} assesses whether the agent correctly identifies the remaining task IDs when the current status is not `success`.

The coefficients w_5, w_6, and w_7 are hyperparameters that balance the contribution of each reward component. The default weights are set as $w_5 = 0.4$, $w_6 = 0.3$, and $w_7 = 0.3$.

Additionally, we incorporate Chain-of-Thought (CoT) reasoning during the reflection phase to explicitly guide the agent through intermediate logical deliberation before outputting the final state diagnosis.

A.2 Stage 3 Reward Design: Multi-turn Joint Reinforcement Training Rewards

Stage 3 is the primary multi-turn training stage, where the agent interacts with the frozen mask-conditioned editor in a multi-turn loop. The rewards in this stage are dynamic and trajectory-based, focusing on the convergence of the entire editing session. In this multi-turn setting, the reward mechanism is bipartite, consisting of an initial planning reward and subsequent reflection rewards guided by a stronger VLM.

Planning Reward. During the initial turn of the editing trajectory, the agent generates the structured edit agenda. To maintain the foundational planning capabilities established earlier, the planning reward R_{plan} remains strictly consistent with the formulation in Stage 1. This ensures that the generated agenda maintains rigorous structural, sequential, and spatial alignment with the ground truth before execution begins.

VLM-Guided Reflection Rewards. For subsequent turns, we employ a stronger, more capable VLM as a Reward Model (RM) to evaluate the intermediate edited images. The RM independently assesses the current editing status after each editor execution. The reflection reward is then dynamically calculated based on the consistency between the agent's self-diagnosis and the RM's objective evaluation.

Specifically, the status classification reward R_{status} prioritizes reliable convergence over greedy completion through the following criteria:

- **Success Reward (High Positive):** Awarded only when both the agent and the RM agree that all tasks are completed correctly.
- **Continue/Progress Reward (Cumulative Positive):** To encourage safe, incremental edits, the agent receives positive reinforcement for correctly identifying a `continue` state that aligns with the RM's assessment. This allows the system to build on partial successes rather than failing a complex request in one go.
- **Rollback Penalty (Heavy Negative):** If the agent triggers a `rollback`, it incurs a "rollback tax"—a penalty for inefficient execution. However, a "correct rollback" (where the agent identifies a fatal error corroborated by the RM and chooses to restart) is penalized far less severely than a "false success," thereby prioritizing product integrity.
- **Early Termination Penalty (Extreme Negative):** We strictly penalize the agent if it erroneously predicts `success` when the RM detects missed operations or visual corruption. This mechanism is crucial to prevent the model from exploiting a shortcut by indiscriminately predicting success to terminate the loop early.

Beyond these status-based rewards, the reflection phase strictly retains the format reward (R_{format}) and the next-task identification reward (R_{next}) introduced in Stage 1. This ensures that the structured diagnosis remains parsable and logically consistent for directing the subsequent turns.

Trajectory-Level Advantage Allocation. Unlike standard RL, the multi-turn reward is assigned to specific token spans across different turns. For a trajectory with T rounds, the total reward is a combination of the initial planning quality (Turn 1) and the subsequent VLM-guided reflection accuracy (Turns 2 to T). We utilize a step-level advantage calculation to ensure that the reward from Round t specifically reinforces the policy parameters responsible for that exact turn's output, preventing negative reflection rewards from degrading competent planning behaviors.

B Details of Dataset Pipeline

To construct a high-quality, fine-grained dataset for e-commerce image editing, our editable element mining pipeline leverages a suite of state-of-the-art vision and language models. This multi-stage perception and generation pipeline is designed to accurately decompose complex e-commerce posters into independent, structured semantic assets. The specific models and their corresponding roles are detailed below:

Product and Graphic Object Localization: We employ the YOLOv10 [43] object detector to identify and localize primary and salient visual elements within the source images. This model is responsible for generating precise bounding boxes for product instances, brand logos, decorative stickers, and other foreground graphic objects, serving as the foundational spatial prior for subsequent extraction.

Layer Segmentation and Matting: To ensure the extracted visual assets are of high fidelity and seamlessly reusable, we utilize a segmenter based on SAM2 [44]. Conditioned on the bounding boxes provided by the localization stage, SAM2 performs pixel-level semantic segmentation and high-quality alpha matting. This process isolates the objects from complex backgrounds, yielding sharp, transparent masks that are crucial for downstream operations like *move* and *exchange*.

Text and Layout Detection: Recognizing the critical importance of text in e-commerce images, we rely on PP-OCRv5 [45] for comprehensive text analysis. This robust OCR model extracts both the textual content and the geometric layout of all text blocks. Capturing this text-specific metadata is essential for enabling localized, attribute-level modifications.

Background Recovery: Once the foreground elements (products, graphics, and text) are extracted, the remaining image contains missing regions. To create a clean, element-free background canvas, we deploy Flux2-klein-9B [33], a powerful diffusion-based inpainting model. This model intelligently fills the voids by referencing the surrounding context, resulting in a natural, semantically consistent background that serves as the foundation for generative recomposition.

Structural Synthesis: Finally, we utilize Qwen3.5 [46], a highly capable Multimodal Large Language Model, as the central integration engine. Acting as the cognitive orchestrator, Qwen3.5 synthesizes the parsed visual attributes, extracted textual content, and layout coordinates from the preceding modules. It translates these discrete, multi-modal components into a structured, machine-interpretable format (e.g., HTML). This hierarchical representation captures the underlying semantic logic and spatial constraints of the original poster, effectively forming the grounded "edit program" used to guide the training of both the VLM agent and the mask-conditioned editor.

C Details of Benchmark

C.1 Benchmark Construction

Following recent benchmark designs that report both task-wise and difficulty-wise results [25, 19], we construct EComEditBench as a held-out benchmark for grounded multi-operation e-commerce image editing. The benchmark contains 1,000 examples covering five atomic operations: `add`, `remove`, `move`, `exchange`, and `replace`. It includes 500 single-operation cases, with 100 examples for each atomic operation, and 500 multi-operation cases, including 200 two-operation, 200 three-operation, and 100 four-operation requests. This split allows us to separately analyze primitive operation execution and compositional generalization.

Each benchmark example contains a source product image, a target image for reference-enabled diagnostics, an optional patch image for insertion tasks, an operation-type string, and structured region annotations. For fuzzy evaluation, we additionally provide a natural-language user instruction

Table 4: Composition of EComEditBench. Single-operation cases are balanced across five atomic operations, while multi-operation cases evaluate compositional editing under multiple local constraints.

Group	Subgroup	Number of samples
Single-operation	`add` / `remove` / `move` / `exchange` / `replace`	$5 \times 100 = 500$
Multi-operation	Two operations	200
	Three operations	200
	Four operations	100
Total	–	1,000

Table 5: Structured annotation fields used by the precise evaluation protocol.

Operation	Required region information	Editing requirement
`add`	Target box and inserted-object label; optional patch image	Insert the specified patch or object at the target region.
`remove`	Source box and object label	Remove the specified object while preserving surrounding content.
`move`	Source box, target box, and object label	Move the object from the source region to the target region.
`exchange`	Source/target boxes for two labeled objects	Swap the two specified objects or regions.
`replace`	Source/target box, label, and replacement constraints	Replace local content, attributes, text, color, or layout as specified.

for each sample. The benchmark is disjoint from the training data at the product and edit-program level.

C.2 Operation Annotation Format

The precise protocol uses a structured edit program converted from region annotations. Each operation is represented by its type, semantic label(s), and the required source and/or target bounding boxes in `xywh` format. The five operation types are defined as follows:

For compositional examples, the edit program contains multiple operation entries. This makes it possible to evaluate whether a model misses sub-operations, edits the wrong region, confuses source and target roles, or introduces spillover outside the intended local regions.

C.3 Precise and Fuzzy Evaluation Protocols

We evaluate all models under two complementary protocols.

Fuzzy protocol. The model receives the source image and a natural-language request written in a user-like form. The instruction may contain approximate spatial descriptions, referring expressions, or implicit operation bindings. For `add` cases, an insertion patch can be provided when the task requires copying a logo, text, graphic, or product element. This protocol evaluates the full editing pipeline, including instruction understanding, operation decomposition, target localization, and local editing.

Precise protocol. The model receives the source image and a structured region-level edit description derived from the annotations in Table 5. For our method, this program is further compiled into operation-aware masks and optional patch references. This protocol removes most grounding ambiguity and isolates the editor's ability to execute local operations faithfully.

C.4 VLM-as-Judge Evaluation

Inspired by recent automatic evaluation protocols for generation and editing benchmarks [25, 19], we use a fixed VLM-as-Judge protocol for all methods. The judge receives images in a fixed order: the source image, the edited result, an optional ground-truth target image in the reference-enabled setting, and an optional patch image for `add` tasks. Unless otherwise specified, the reported results use the same Gemini-3 Pro judge as the main paper; the implementation keeps the judge temperature at 0 and uses the same prompt and parsing rules for every model.

The judge outputs three integer scores in $[1, 10]$: **Instruction Accuracy (IA)**, **Edit Fidelity (EF)**, and **Background Preservation (BP)**. IA measures whether all requested operations are applied to the

Table 6: Score bands used by the VLM evaluator.

Metric	10	7–9	4–6	1–3
IA	All operations exactly executed	Minor omission or placement error	Partial execution or some wrong operations	Instruction largely not executed
EF	Correct and natural edited content	Minor visual/style/text issues	Recognizable but noticeably imperfect	Wrong, distorted, or missing edit content
BP	Pixel-level background preservation	Imperceptible drift	Visible unintended change but structure kept	Major unintended background alteration

correct objects and locations. EF measures the semantic correctness and visual quality of the edited regions, including patch faithfulness for insertion tasks. BP measures whether non-target regions remain unchanged.

C.5 Evaluation Prompt Template

We use one unified prompt template for both protocols. The placeholders are filled according to the instruction mode, whether a ground-truth target is used, and whether the sample contains an insertion patch.

> **Role.** You are an expert image editing evaluator. You will be shown multiple images in this exact order: source image, edited image, optional ground-truth target image, and optional patch image.
>
> **Task.** Evaluate the quality of the edited image against the benchmark instruction, the source image, and optional references.
>
> **Benchmark Instruction.** `{benchmark_instruction}`
>
> **Evaluation Dimensions.**
> 1. `instruction_accuracy`: Did the model execute the specified edit operations?
> 2. `edit_fidelity`: How well does the edited result satisfy the requested edit inside the target regions? For insertion tasks, also judge whether the inserted content matches the patch image.
> 3. `background_preservation`: Outside the requested edit regions, is the edited image unchanged compared with the source image?
>
> **Instructions.** Read the benchmark instruction carefully. In the precise protocol, examine each bounding box and reference region indices in the reasoning. In the fuzzy protocol, identify the intended edit from the natural-language request and visible evidence. Be objective and do not reward changes that were not requested.
>
> **Response Format.** Return exactly three integer scores in $[1, 10]$ followed by a short explanation:

```
<scores>
instruction_accuracy: [int 1--10]
edit_fidelity: [int 1--10]
background_preservation: [int 1--10]
</scores>
<reasoning>
2--4 sentences referencing the instruction and visible evidence.
</reasoning>
```

For the precise protocol, `{benchmark_instruction}` is a formatted list of operation entries and bounding boxes. For the fuzzy protocol, it is the natural-language user request. The same response parser is used in both settings, and a sample is treated as invalid if any of the three scores cannot be parsed as an integer in $[1, 10]$.

C.6 Aggregation and Reporting

After scoring, we average IA, EF, and BP over all valid samples. We also report fine-grained results by single-operation type and by the number of operations in compositional cases. Specifically, single-operation samples are grouped by their atomic operation type, while multi-operation samples are grouped into two-, three-, and four-operation cases. This aggregation exposes complementary failure modes: low IA indicates grounding or operation-binding failures, low EF indicates poor local rendering or semantic mismatch, and low BP indicates unintended changes to non-target regions.

D Experiments Details

Our image editing backbone is initialized from Flux2-Klein-9B [33], while the VLM editing agent is based on Qwen3.5-9B [46]. The VLM handles instruction parsing, region-level planning, and visual reflection, whereas the editor receives the source image along with operation-aware masks and optional patch references. We optimize both models using AdamW. In Stage 1, the VLM undergoes supervised fine-tuning (SFT) with LoRA of rank 32 and a learning rate of 1×10^{-4} for 20K steps with a batch size of 8. The Stage 1 GRPO training uses a group size of 8, learning rate 5×10^{-6}, batch size 16, and runs for 1,200 steps. The image editor is fine-tuned separately with LoRA of rank 128, learning rate 1×10^{-4}, for 25K steps. Stage 3 employs multi-turn GRPO with a sampling group of 8, LoRA rank 16, batch size 8, learning rate 1×10^{-5}, and 500 training steps. Qwen3-VL-32B [47] serves as the reward model, and the maximum number of execution rounds during sampling is set to 4. Unless otherwise specified, all experiments are conducted on the same constructed training dataset and evaluated on the held-out EComEditBench split. All experiments are conducted on 16 NVIDIA B200 GPUs.

E More Experiment Results

E.1 Multi-turn Editing

To evaluate our multi-turn joint RL and reflection-driven mechanism, we conduct experiments on a test set containing 7–10 task editing scenarios per image, designed to validate the model's ability to handle multi-operation, region-specific instructions. Table 3 compares four inference configurations:

- **Stage1-one-turn:** The baseline agent (after Stage 1 SFT and rule-based RL) executing the entire edit agenda in a single forward pass.
- **Stage3-one-turn:** The Stage 3 agent restricted to a single turn, isolating the intrinsic performance gains of joint RL training without using multi-round rollouts.
- **Stage3-fixed-turn:** The Stage 3 agent executing a fixed maximum number of rounds (e.g., 3), dividing tasks blindly without adaptive state reflection.
- **Stage3-reflection (Ours):** Our full framework. The agent autonomously evaluates intermediate states to dynamically choose to `success` (terminate), `continue` (execute remaining), or `rollback` (discard and retry).

Quantitative Analysis. As shown in Table 3, **Stage3-one-turn** outperforms **Stage1-one-turn** (IA: 7.04 vs. 6.89; EF: 6.23 vs. 5.98), demonstrating that joint RL inherently improves spatial planning even for single-pass generation. Extending to multiple rounds, **Stage3-fixed-turn** further elevates IA and EF (7.21 and 6.37) by successfully recovering missed operations. However, forcing fixed iterative edits drops Background Preservation (BP) from 8.14 to 8.08 due to noise accumulation from unnecessary forward passes. Our **Stage3-reflection** policy optimally resolves this trade-off. By adaptively halting completed tasks to prevent over-editing and rolling back corrupted steps, it achieves the highest scores across all metrics (7.29 IA, 6.47 EF, 8.42 BP), ensuring strict product preservation.

Qualitative Analysis. As illustrated in Fig. 8, 9, 10, and 11, single-turn editors often suffer from *instruction interference* on complex requests, leading to attribute bleeding, or omitted edits. Our multi-turn paradigm mitigates this by chunking requests into digestible sub-programs, highlighting two key behaviors: 1) Incremental Safety: Isolating distinct operations across turns prevents color or text styles from bleeding between adjacent regions. 2) Autonomous Recovery: If an intermediate edit distorts text or misplaces patches, the agent identifies the anomaly, triggers a `rollback`, and attempts an alternative sub-plan. These results validate our framework's capacity for reliable sequential planning in real-world e-commerce applications.

E.2 Additional Qualitative Results

To further demonstrate the effectiveness of GMO-E^{2}DIT, we provide additional visual comparisons against strong closed-source models. Fig. 7 presents results across multiple multi-operation editing tasks, highlighting our framework's ability to accurately perform localized edits and preserve unmodified content.

F Limitations

While our proposed GMO-E^{2}DIT achieves strong performance in multi-task e-commerce image editing, several limitations remain to be addressed in future work.

Dependence on VLM Capabilities. First, the framework's effectiveness relies heavily on the cognitive reasoning and reflection abilities of the VLM agent. Errors in understanding complex instructions during the planning phase or misjudgments of intermediate states during reflection can compromise the entire multi-operation workflow. In practical applications, user prompts may be extremely vague, ambiguous, or linguistically unusual, leading to incorrect operation bindings, hallucinated edits, or omitted sub-tasks. These challenges highlight the need for more robust intent-resolution mechanisms.

Dataset Limitations and Generalization. Second, the model's generalization is inherently constrained by the constructed dataset. Although the training pipeline and EComEditBench benchmark cover a comprehensive set of operations (e.g., `add`, `remove`, `move`, `exchange`, `replace`), they are primarily derived from synthetic or semi-synthetic e-commerce posters. Real-world imagery often presents more complex layouts, diverse typography, and severe visual clutter.

Computational Cost and Efficiency. Finally, the multi-turn editing mechanism introduces a trade-off between reliability and computational efficiency. The reflection-driven iterative loop, which employs `continue` and `rollback` actions, substantially improves task completion and reduces error propagation, but it requires multiple sequential forward passes through both the VLM agent and the diffusion-based image editor. Consequently, the iterative process increases inference latency and computational overhead. Deploying the framework for real-time interactive editing or large-scale batch processing may therefore face practical efficiency limitations. Future work will explore model distillation and more efficient sampling strategies to mitigate these constraints.



Figure 7: More visual comparison with closed-source methods across multiple tasks.

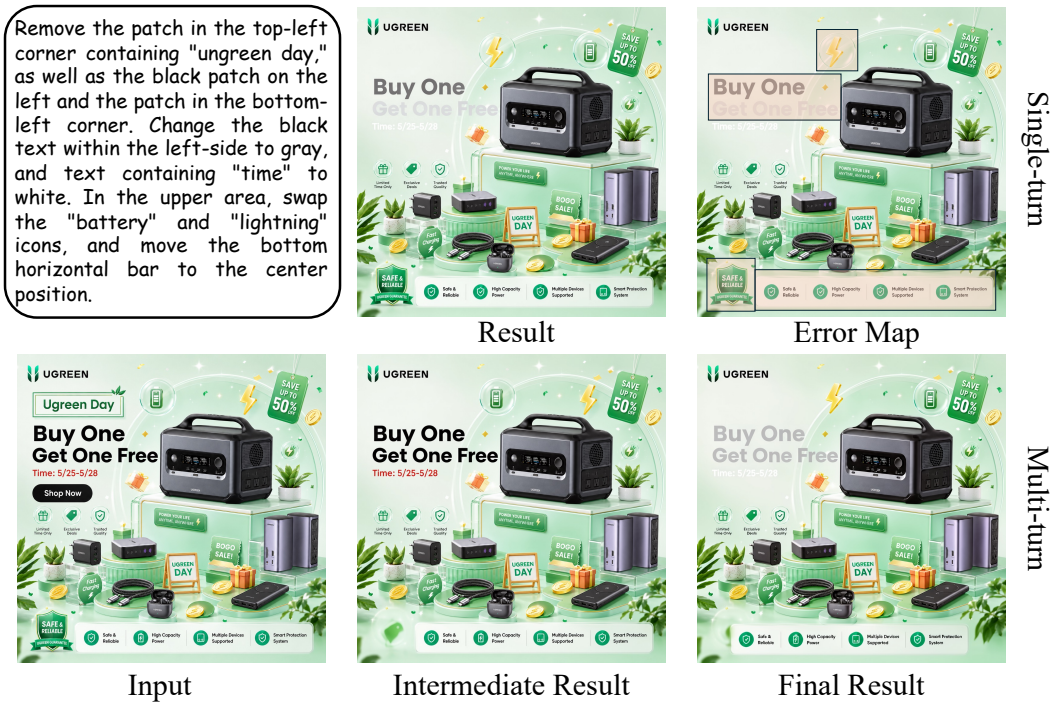

Figure 8: Comparison of single-turn vs. multi-turn instruction execution. (Example 1)

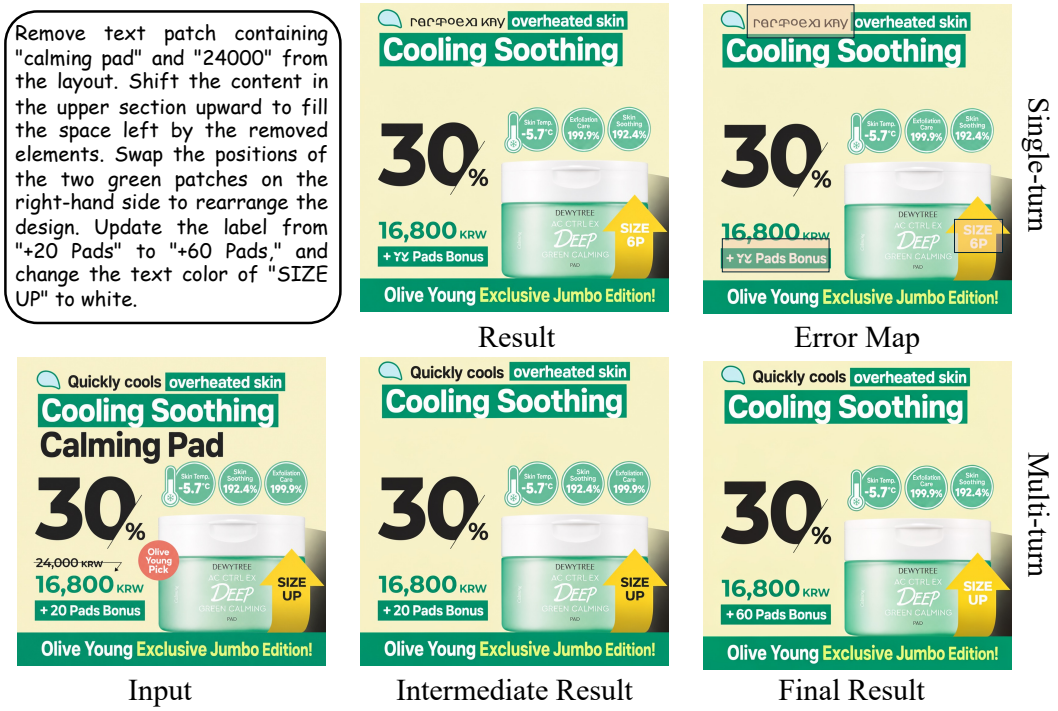

Figure 9: Comparison of single-turn vs. multi-turn instruction execution. (Example 2)



Figure 10: Comparison of single-turn vs. multi-turn instruction execution. (Example 3)



Figure 11: Comparison of single-turn vs. multi-turn instruction execution. (Example 4)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction outline our theoretical and technical contributions, the proposed methodology, and a comprehensive experimental evaluation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: At the end of the appendix, a limitation discussion is provided.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: A complete proof is given in the method section.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, the baseline model, technical details, hyperparameters, and configuration are detailed in the submission for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets this paper uses are publicly available, and the source code is promised to be public once published.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The implementation details are given at the beginning of the experimental section and the beginning of the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following prior works in this field, the evaluation protocols on the corresponding datasets do NOT require a report of the error bar.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The implementation details are given at the beginning of the experimental section and the beginning of the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper focuses on a fundamental task of machine learning and conducts experiments on publicly available datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The societal impact of this work has been discussed at the end of the conclusion section. We do not envision negative societal impact could be brought by this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not envision such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the assets have been properly cited, with a license to use for academia and no commercial purpose.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We leverage Large Language Models (LLMs) for both data curation and evaluation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.