TEST-TIME SCALING FOR RELIABLE IMAGE EDITING VIA CONTRACT-GUIDED VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling has substantially improved instruction-guided image editing by generating and comparing multiple candidate edits. However, ranking candidates by a unified evaluation score may cause highly ranked outputs to omit requested changes or alter content that should remain unchanged. In this work, we propose Contract-Guided Test-Time Scaling (CG-TTS), a training-free approach that uses explicit edit requirements to guide generation, verification, and repair with in place image-editing models. Specifically, for each instruction requirement, CG-TTS first constructs a contract specifying the editing target, requested changes, allowed regions, and preserved content, together with checks used to verify the given requirements. Then, we generate diverse candidate edits using a frozen editor equipped with a source-memory routing mechanism, which routes cached source-image features according to the contract. A staged verifier is further used to check spatial changes and operation-specific requirements, accepting candidates only when all active mandatory checks pass. If no candidates pass the verifier, CG-TTS uses verifier feedback to obtain diagnosis types that guide failure-conditioned repair, and the repaired candidates are verified against the same contract. Finally, CG-TTS selects the highest-ranked passing candidate as the output, while if none passes, any permitted fallback retains a non-pass status. Experimental results demonstrate that our method consistently improves performance across strong baseline models and outperforms state-of-the-art methods under the same editor-call budget. Code is available at https://anonymous.4open.science/r/s-E968.

1 INTRODUCTION

Instruction-guided image editing has advanced rapidly, yet obtaining a reliable editing result from a single model invocation remains challenging because success usually depends on several coupled conditions (Brooks et al., 2023), *i.e.*, a satisfactory edit should apply all requested changes to the intended targets while preserving non-target content and maintaining visual plausibility. For example, an instruction to turn only the left cup red is not desirable by recoloring both cups, even if the result is visually convincing. Existing approaches reveal quantitative and qualitative balance among instruction following, content preservation, and perceptual quality (Zhang et al., 2023; Ju et al., 2024b), including in complex image-to-image settings (Wang et al., 2026) and multi-turn editing (Zhou et al., 2025).

The naive test-time scaling can improve the image editing quality without retraining the base editor by allocating extra inference computation to Best-of-N sampling, verifier-guided search, or iterative refinement (Ma et al., 2025; Li et al., 2025). Figure 1(a) illustrates the commonly-used Best-of-N pipeline that generates multiple candidates with a frozen editor and selects the one with the highest scalar score, *e.g.*, a reward or preference score (Xu et al., 2023; Kirstain et al., 2023). Recent editing-specific methods further improve the use of test-time computation through preservation-aware candidate selection (Kim et al., 2025a) and verification-guided search (Qu et al., 2026). However, if the selection score permits trade-offs across multiple requirements within the given instruction, a candidate may rank highly despite omitting a requested change or modifying content that should remain unchanged (Kim et al., 2025b; Coste et al., 2024). Therefore, increasing the number of candidates may raise the best score but the selected output still cannot satisfy all requirements.

Fine-grained evaluators further assess individual semantic requirements, including object presence, attributes, and relations (Hu et al., 2023; Cho et al., 2024; Lin et al., 2026). These signals are used to

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

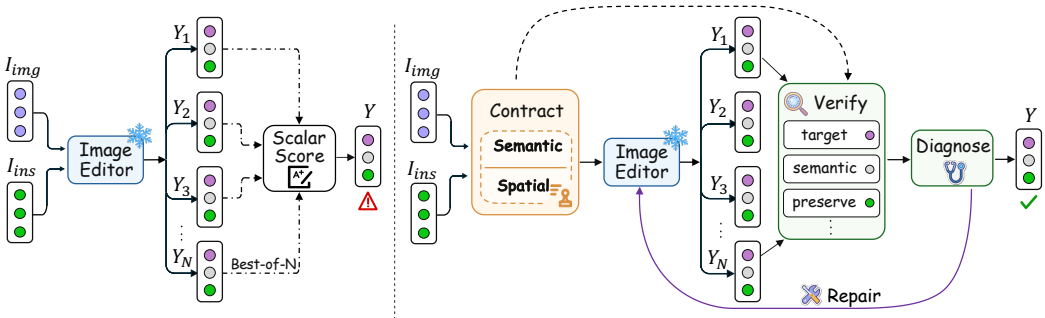

Figure 1: Comparisons of test-time paradigms for image editing. (a) Naive test-time scaling generates multiple candidates and selects the Best-of-N using a generic scalar score. (b) CG-TTS uses explicit edit requirements to guide generation, verification, and repair with existing image-editing models.

construct training rewards (Guo et al., 2026) and guide test-time search (Qu et al., 2026). However, fine-grained judgments alone cannot change the score-driven nature of candidate selection in existing test-time scaling methods (Yarom et al., 2023; Hosseini et al., 2025). Once collapsed into a single scalar, success on some sub-requirements can offset violations of others, leaving their joint satisfaction unverified (Xu et al., 2023; Kirstain et al., 2023; Qu et al., 2026). A central challenge is therefore to convert clause-level verification from a ranking signal into a non-compensatory acceptance criterion, where a candidate is admissible only when every mandatory editing and preservation requirement is satisfied. When no candidate meets this criterion, a further question is whether and how the remaining editor-call budget can be directed toward resolving the diagnosed failure. This distinction makes verifier reliability central, *i.e.*, an accepted edit is useful only when the acceptance decision agrees with an independent assessment of the underlying contract, while excessive rejection can improve apparent reliability at the cost of coverage.

In this work, we propose CG-TTS, a training-free framework that uses an edit contract to guide candidate generation, staged verification, and failure-conditioned repair. As illustrated in Figure 1(b), given a source image and an editing instruction, CG-TTS first constructs an edit contract with both semantic and spatial components. The semantic contract configures operation-specific generation settings and verification checks, while the spatial contract delimit editable and protected regions. The contract can provide a shared set of requirements for candidate generation and acceptance. Then, multiple candidates are generated with a frozen editor. Subsequently, a staged verifier is used to check spatial changes before evaluating operation-specific and semantic requirements, accepting a candidate only when all active mandatory checks pass. If no initial candidate passes, CG-TTS performs at most one failure-conditioned repair round within the remaining editor-call budget. Repaired candidates are verified against the unchanged contract.

Comprehensive evaluations are conducted on the challenging ReasonEdit (Huang et al., 2024) and HumanEdit-core (Bai et al., 2025) datasets, with comparisons against representative BrushNet (Ju et al., 2024a) and SDXL-Inpainting (Podell et al., 2024) baselines. With BrushNet, CG-TTS increases the GPT-5 IC score from 0.500 to 0.794 and background PSNR from 25.393 to 37.964 dB on ReasonEdit-U. On HumanEdit-core, it increases the GPT-5 IC score from 0.396 to 0.442.

2 Related Work

Instruction-Guided Image Editing. Instruction-guided image editing modifies a source image according to a language instruction while preserving visual content unrelated to the requested change. For example, InstructPix2Pix trains a diffusion model on image pairs and their corresponding editing instructions. MGIE (Fu et al., 2024) and SmartEdit (Huang et al., 2024) use multimodal large language models to interpret editing instructions in the context of the source image. OmniGen (Xiao et al., 2025) and FLUX.1 Kontext (Black Forest Labs et al., 2025) support both image generation and editing. Besides, complementary work improves spatial control to limit unintended changes to non-target content. BrushNet provides mask-conditioned inpainting (Ju et al., 2024a), while BrushEdit (Li et al., 2026) and SmartFreeEdit (Sun et al., 2025) infer edit regions from instructions to guide generation. Although these methods can improve instruction following and spatial control,

generation-time guidance alone does not establish whether the requested edit was completed while unrelated content was preserved. CG-TTS combines contract-guided verification with targeted repair, allowing incomplete edits and unintended changes to be corrected without retraining the editor.

Test-Time Scaling for Image Generation and Editing. Test-time scaling uses inference computation to generate, compare, and refine candidates (Ma et al., 2025; Li et al., 2025). For example, ImageReward (Xu et al., 2023) and PickScore (Kirstain et al., 2023) provide learned preference scores for ranking generated images. For image editing, ELECT (Kim et al., 2025a) selects noise seeds using early estimates of background inconsistency and extends to prompt selection when seed selection fails. ADE-CoT (Qu et al., 2026) combines difficulty-aware budget allocation, edit-specific pruning, and instance-specific verification for early stopping, and selects the final output using a combined evaluation score. ELECT uses preservation-related evidence to filter editing candidates, while ADE-CoT incorporates edit-specific checks into pruning and instance-specific verification into stopping (Kim et al., 2025a; Qu et al., 2026). CG-TTS differs in the role assigned to verification, *i.e.*, every instantiated mandatory check can veto final acceptance, and failed clauses guide the subsequent repair action. Scalar evidence ranks candidates only after they pass these checks. This makes false acceptance, false rejection, coverage, and inference cost relevant to evaluating the method.

Fine-Grained Evaluation and Verification. Fine-grained evaluation and verification assess specific aspects of generated or edited images, including semantic correctness, content preservation, and visual quality. TIFA (Hu et al., 2023) evaluates text-to-image faithfulness through generated question–answer pairs, and Davidsonian Scene Graph (Cho et al., 2024) organizes questions by their dependencies to improve coverage and answer consistency. For image editing, ADIEE (Chen et al., 2025) automatically constructs evaluation data and trains an editing-specific scoring model. Fine-grained verification has also been used to construct rewards for reinforcement learning in image editing (Guo et al., 2026; Miao et al., 2026). However, a fine-grained assessment can identify an unmet requirement without making that failure decisive in candidate acceptance. CG-TTS instead requires every active mandatory check to return `PASS` before accepting a candidate, so similarity or quality scores cannot override `FAIL` or `UNCERTAIN`. The same acceptance conditions apply to repaired candidates under the unchanged contract.

3 Reliable Image Editing via Contract-Guided Verification

3.1 Overview

As shown in Figure 2, CG-TTS mainly comprises three stages. **Stage 1: Semantic-Spatial Contract Construction** (in Sec. 3.2), where we construct an edit contract C specifying semantic editing requirements, spatial constraints, and mandatory verification checks. **Stage 2: Source-Memory Routing-Augmented Candidate Generation** (in Sec 3.3), where a frozen in place editor generates contract-conditioned candidates, and a Source-Memory Routing (SMR) mechanism reuses source representations to preserve relevant content during generation. **Stage 3: Contract-Guided Verification** (in Sec 3.4), where a staged verifier accepts candidates only when all active mandatory checks pass. If no candidate passes, failure diagnosis determines whether repair is triggered and guides failure-conditioned regeneration. Repaired candidates are verified against the same contract, and final selection prioritizes passing candidates.

3.2 Semantic-Spatial Contract Construction

Scalar evaluation scores may select candidates that omit requested changes or modify content that should remain unchanged. CG-TTS therefore constructs an explicit edit contract to specify semantic requirements, spatial constraints, and mandatory verification conditions. Given a source image I_{img} and an editing instruction I_{ins}, we construct an edit contract $C = (\mathcal{S}, \mathcal{M})$, where $\mathcal{S}$ denotes semantic contract fields and $\mathcal{M}$ denotes spatial contract masks. For benchmarks, *e.g.*, ReasonEdit (Huang et al., 2024) and HumanEdit-core (Bai et al., 2025), a frozen multimodal large language model (MLLM) produces a structured semantic description from the source image and instruction (OpenAI, 2024). Specifically, this description records the intended editing operation, the source entity and its relevant attributes, and the desired final state requested by the instruction. The resulting description is processed by deterministic rules to normalize the editing operation and standardize the source and requested-target descriptions.

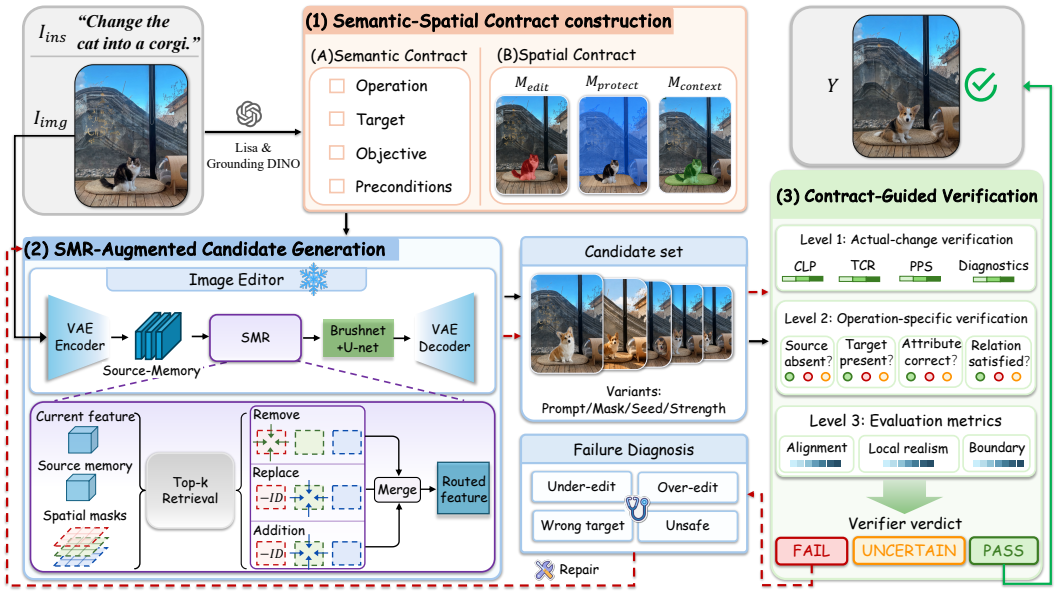

Figure 2: Overview of CG-TTS. A semantic–spatial edit contract guides SMR-augmented candidate generation and staged verification. The verifier accepts a candidate only when all mandatory checks pass. If no candidate is accepted, failure diagnosis guides a single repair round, and the repaired candidates are verified against the same contract.

Semantic Contract. The semantic contract specifies: what operation to perform, which object to change, and what the result should be. We represent the semantic contract fields as:

$$\mathcal{S} = \{o, \mathcal{E}, g, \mathcal{K}\}, \tag{1}$$

where o denotes the normalized editing operation, $\mathcal{E}$ contains descriptions of the source entity and requested target , together with the attributes or spatial information needed to identify the intended instance. The field g specifies the desired final state, while $\mathcal{K}$ contains operation-specific conditions that complement the desired final state. For example, the REPLACE operation uses o to specify the replacement action, $\mathcal{E}$ to identify the source entity and the requested target, and g to describe the desired final state after replacement. The set $\mathcal{K}$ records additional constraints, such as preserving required object attributes, counts, or spatial relations specified by the instruction. The contract specifies the semantic requirements, while the verifier determines whether the available evidence establishes each requirement (Section 3.4).

Spatial Contract. We use frozen LISA (Lai et al., 2024) for instruction-grounded target localization. Frozen Grounding DINO (Liu et al., 2024) for auxiliary proposals of non-target foreground instances, including the same-class distractors. Using these localization results and the editing operation, we construct the spatial contract mask as:

$$\mathcal{M} = \big(M_{\text{edit}}, M_{\text{protect}}, M_{\text{context}}\big), \tag{2}$$

where M_{edit} specifies the region to be edited for operation o, and M_{protect} specifies the region whose content should remain unchanged. The mask M_{context} specifies the local context region where limited changes are permitted to maintain visual continuity. These three masks define pairwise disjoint regions, which can be formulated as:

$$M_{\text{edit}} \cap M_{\text{context}} = \varnothing, M_{\text{protect}} \cap \big(M_{\text{edit}} \cup M_{\text{context}}\big) \quad = \varnothing. \tag{3}$$

We provide the detailed contract instantiation and mask construction are provided in the appendix A.2.

3.3 SOURCE-MEMORY ROUTING-AUGMENTED CANDIDATE GENERATION

Candidate generation with frozen editors must balance diverse edit hypotheses with preservation of source content. CG-TTS generates contract-conditioned candidates with a source-memory routing

mechanism that reuses cached source representations from the frozen editor without additional training. Based on the edit contract $C = (\mathcal{S}, \mathcal{M})$, we configure the i-th editor call in round r as:

$$a_i^{(r)} = \left(\rho_i^{(r)}, m_i^{(r)}, z_i^{(r)}, \theta_i^{(r)}\right), \tag{4}$$

where $\rho_i^{(r)}$ is the generation prompt derived from $\mathcal{S}$ by concatenating the normalized operation o and the desired final state g in a fixed order. $m_i^{(r)}$ denotes the spatial mask obtained from $\mathcal{M}$ by selecting the operation-relevant edit or context region and applying call-specific dilation. The configuration also includes a random seed $z_i^{(r)}$, and backend-specific inference settings $\theta_i^{(r)}$. The frozen editor G (Ju et al., 2024a) then generates $Y_i^{(r)} = G(I_{\text{img}}; a_i^{(r)}, C)$ for $i = 1, \ldots, n_r$, where n_r is the number of editor calls in round r. The initial round generates candidates without prior verification feedback, while an optional repair round updates the generation configurations according to detected failures. For each input, we define the realized number of editor calls as: $b = n_0 + n_1 \leq B$, where B denotes the maximum editor-call allowance and $n_1 = 0$ when repair is not triggered. In the appendix, we provide the inference configuration and accounting convention.

Within the frozen editor generation process, we introduce Source-Memory Routing (SMR), which reuses the frozen editor's cached source representations as a spatial memory without additional trainable parameters (Tumanyan et al., 2023; Cao et al., 2023). For source features $X = \{x_j\}_{j=1}^{N}$, each memory entry is defined as $e_j(X) = \left(k_j(X), v_j(X), p_j\right)$, where $k_j(X)$ and $v_j(X)$ denote the matching key and value associated with the cached source feature, and p_j denotes the normalized spatial position. The operation o and spatial constraints encoded in $\mathcal{M}$ determine the admissible memory entries $\mathcal{A} = \mathcal{A}(o, \mathcal{M})$. For a query feature q_i at position p_i, SMR computes the routing score, which can be formulated as:

$$s_{ij}(X) = \frac{\cos(q_i, k_j(X))}{\tau} - \lambda_{\text{geo}} \|p_i - p_j\|_2^2, \qquad j \in \mathcal{A}, \tag{5}$$

and aggregates the top-k retrieved source features as:

$$R_i^{\text{src}} = \sum_{j \in \mathcal{N}_k(i;\mathcal{A})} \frac{\exp(s_{ij})}{\sum_{h \in \mathcal{N}_k(i;\mathcal{A})} \exp(s_{ih})} v_j(X). \tag{6}$$

SMR selectively reuses contract-consistent source information while excluding protected regions from routing. In the appendix, we provide implementation details of SMR, including routing regions and update rules. The generated candidates are subsequently evaluated against the same edit contract C in Section 3.4.

3.4 Contract-Guided Verification

Candidate generation alone does not guarantee that all editing requirements are satisfied. CG-TTS therefore performs contract-guided verification, accepting candidates only when all active mandatory checks pass and using diagnosed failures for repair. The verifier evaluates each candidate $Y_i^{(r)}$ against the edit contract C through three levels (Qu et al., 2026). *First*, the actual-change verification checks whether the candidate introduces spatial changes consistent with the spatial contract $\mathcal{M}$ (Kim et al., 2025a). We compare $Y_i^{(r)}$ with the source image I_{img} using color and intensity differences to obtain a per-pixel change score. Based on this score, CLP measures change localization within the allowed region M_{allow}, TCR measures change coverage within M_{edit}, and PPS measures content preservation within M_{protect}. In the appendix, we provide the score definitions and verification thresholds. *Second*, operation-specific verification evaluates whether the edited result satisfies the semantic requirements encoded in C, including requirements related to the requested operation o and desired final state g (Liu et al., 2026). The operation-specific checks are detailed in the appendix A.2. *Third*, evaluation metrics provides auxiliary signals for comparing candidates that already satisfy mandatory requirements (Chen et al., 2025). Let $\mathcal{Q}_{\text{mand}}(C)$ contain the mandatory semantic and spatial checks instantiated for contract C. Each check $q \in \mathcal{Q}_{\text{mand}}(C)$ returns $v_q(Y, C) \in \{\texttt{PASS}, \texttt{FAIL}, \texttt{UNCERTAIN}\}$. A candidate is accepted only if every instantiated mandatory check returns `PASS`; a `FAIL` or `UNCERTAIN` decision cannot be offset by soft similarity or quality evidence. Such evidence is used only to rank accepted candidates. Verifier acceptance is an estimated decision, not an independent determination of editing correctness. We therefore distinguish

whether a candidate passes the instantiated checks from whether it satisfies the original instruction and preservation requirements under independent adjudication. If no candidate satisfies the contract, CG-TTS performs failure diagnosis, and for a repairable failure type ϕ, a fixed repair policy updates the generation configuration:

$$a_j^{(1)} = \mathcal{R}\left(a_j^{(0)}, \phi_j, C\right), \qquad j = 1, \ldots, n_1, \tag{7}$$

where ϕ_j denotes the diagnosed failure associated with the candidate. The repair operator updates only the candidate-generation configuration, while keeping the mandatory semantic requirements and protection constraints in C fixed. A diagnosed failure that would require changing these acceptance conditions is not treated as a repair under the same contract. Repaired candidates are verified using the unchanged contract, and the final output is selected using the same feasibility-first procedure. If no candidate passes verification, any fallback retains its non-pass status rather than being treated as an accepted edit. In the appendix, we provide implementation details of verification and repair.

4 Experiments

Benchmarks and Metrics. We evaluate CG-TTS on ReasonEdit (Huang et al., 2024) and HumanEdit-core (Bai et al., 2025). ReasonEdit contains Understanding and Reasoning subsets and evaluates editing preservation and instruction–crop alignment using PSNR, LPIPS, SSIM, and CLIPSim following its official protocol (Huang et al., 2024). HumanEdit-core evaluates high-resolution instruction-based editing using content preservation, image similarity, and text-alignment metrics (Bai et al., 2025). Instruction compliance (IC) is additionally assessed through post-hoc MLLM evaluation.

Baselines. We compare CG-TTS with its corresponding base editors and the benchmark-specific methods listed in Tables 1 and 3. ReasonEdit comparisons include InstructPix2Pix (Brooks et al., 2023), InstructDiffusion (Geng et al., 2024), MagicBrush (Zhang et al., 2023), BrushEdit (Li et al., 2026), SmartEdit-7B/13B (Huang et al., 2024), Gemini 2.0 Flash (Kampf & Brichtova, 2025), and SmartFreeEdit (Sun et al., 2025). HumanEdit-core comparisons include InstructPix2Pix (Brooks et al., 2023), MGIE (Fu et al., 2024), HIVE (Zhang et al., 2024), MagicBrush (Zhang et al., 2023), and SmartFreeEdit (Sun et al., 2025). The benchmark comparisons include published baseline results.

Implementation Details. We use a frozen GPT-4o (OpenAI, 2024) for contract construction on both ReasonEdit and HumanEdit-core. BrushNet (Ju et al., 2024a) and SDXL-Inpainting (Podell et al., 2024) serve as the base editors. For the main benchmarks, CG-TTS generates four initial candidates and, only when verification triggers repair, four additional candidates in a single repair round, giving a maximum editor-call allowance of $B = 8$ per input. All models remain frozen, and inference uses no benchmark-provided masks. Both benchmarks use MLLM-assisted semantic verification. We provide the details of the MLLM-based IC evaluation in the Appendix B.1.

4.1 Main Results

Results on ReasonEdit (Huang et al., 2024). Table 1 reports results on ReasonEdit-U and ReasonEdit-R. Across both subsets and backends, CG-TTS improves preservation metrics(PSNR, LPIPS (Zhang et al., 2018), and SSIM (Wang et al., 2004)) over the corresponding frozen editors. On ReasonEdit-U, CG-TTS with SDXL (Podell et al., 2024) achieves the strongest preservation results, while the BrushNet (Ju et al., 2024a) variant obtains the highest IC scores (GPT-5/Gemini 2.5 Pro: 0.794/0.694). Compared with scalar Best-of-8 selection using the same candidate pool, CG-TTS further improves instruction compliance, achieving higher GPT-5 IC than CLIP, VIEScore, and EditScore selection (0.794 vs. 0.744/0.600/0.723) on ReasonEdit-U with BrushNet. Similar trends are observed on ReasonEdit-R, where SDXL provides better reconstruction metrics and BrushNet achieves higher IC (0.869/0.781).

To further analyze why contract-guided verification improves over scalar candidate ranking, we perform a controlled comparison using the same candidate pool. Table 2 shows that scalar selection methods often choose candidates with comparable CLIP similarity but remaining mandatory constraint violations. In contrast, CG-TTS substantially reduces the proportion of selected candidates violating mandatory constraints from 48.1–59.5% to 21.4%, while maintaining a similar CLIPSim (24.04). This indicates that contract-based acceptance reduces compensatory failures that can be hidden by scalar aggregation.

Table 1: Experimental results on ReasonEdit (Huang et al., 2024). "U/R" denote Understanding/Reasoning, respectively. Boldface denotes the best results. Scalar Best-of-8 controls are included for reference and use the same candidate pool and the same eight editor calls. IC† is evaluated on U-80 for Understanding and R-60 for Reasoning.

	Understanding / Reasoning (U/R)					
Method	PSNR ↑	LPIPS ($\times 10^3$) ↓	SSIM ↑	CLIPSim ↑	GPT-5 IC† ↑	Gemini IC† ↑
Existing Editing Methods						
InstructPix2Pix (Brooks et al., 2023)	21.570/21.010	89.00/14.00	0.720/0.640	22.760/20.020	0.331/0.150	0.025/0.033
InstructDiffusion (Geng et al., 2024)	13.820/19.130	277.30/16.50	0.590/0.630	**25.030**/20.230	0.375/0.408	0.088/0.208
MagicBrush (Zhang et al., 2023)	16.850/19.850	185.20/16.21	0.660/0.650	23.440/19.620	0.144/0.300	0.019/0.167
BrushEdit (Li et al., 2026)	28.450/30.450	61.90/8.01	0.900/0.880	20.190/18.470	0.363/0.217	0.231/0.133
SmartEdit-7B (Huang et al., 2024)	22.050/25.260	87.00/55.67	0.730/0.740	23.610/20.950	0.405/0.488	0.518/0.516
SmartEdit-13B (Huang et al., 2024)	23.600/25.760	68.01/51.43	0.750/0.740	23.540/20.780	0.440/0.521	0.518/0.483
Gemini 2.0 Flash (Comanici et al., 2025)	20.060/22.470	138.76/105.02	0.730/0.730	22.980/20.800	0.587/0.519	0.383/0.307
SmartFreeEdit (Sun et al., 2025)	28.990/31.270	51.49/47.00	0.920/0.920	24.430/**21.460**	0.425/0.550	0.475/0.683
Frozen Editors and TTS Baselines						
BrushNet (Ju et al., 2024a)	25.393/25.846	46.58/55.99	0.797/0.749	21.755/18.391	0.500/0.217	0.425/0.175
SDXL Inpainting (Podell et al., 2024)	27.839/26.320	20.16/26.80	0.882/0.825	20.475/18.693	0.344/0.358	0.263/0.292
ADE-CoT (Qu et al., 2026)	14.080/21.764	254.01/139.97	0.642/0.793	22.357/18.175	0.604/0.717	0.479/0.642
ELECT (Kim et al., 2025a)	14.023/17.416	304.23/240.23	0.504/0.570	21.954/17.991	0.414/0.417	0.164/0.217
Scalar Best-of-8 Controls (BrushNet; Same Candidate Pool)						
CLIP Best-of-8	34.746/33.275	11.29/4.07	0.986/0.996	21.594/19.674	0.744/0.523	0.669/0.489
VIEScore Best-of-8	35.087/34.186	10.44/4.16	0.987/0.995	20.886/19.589	0.600/0.492	0.680/0.499
EditScore Best-of-8	35.551/34.372	9.64/4.17	0.988/**0.996**	20.119/19.421	0.723/0.523	0.667/0.477
Ours						
CG-TTS (BrushNet)	37.964/36.562	10.59/11.10	0.986/0.986	24.075/20.829	**0.794/0.869**	**0.694/0.781**
Gain vs. BrushNet	+12.571/+10.716	+35.99/+44.89	+0.189/+0.237	+2.320/+2.438	+0.294/+0.652	+0.269/+0.606
CG-TTS (SDXL Inpainting)	**40.882/42.376**	**8.42/4.06**	**0.989**/0.993	22.579/19.899	0.525/0.333	0.419/0.350
Gain vs. SDXL Inpainting	+13.043/+16.056	+11.74/+22.74	+0.107/+0.168	+2.104/+1.206	+0.181/ −0.025	+0.156/+0.058

Table 2: Verification analysis with the same candidate pool.

Metric	CLIP	VIEScore	EditScore	CG-TTS
Passing checks (%)	59.50	57.30	43.50	**90.80**
CLIPSim	24.59	23.89	23.12	24.04
Constraint violations (%)	49.60	48.10	59.50	**21.40**
Avg. failed clauses	0.86	0.72	0.84	**0.37**

Results on HumanEdit-core (Bai et al., 2025). As shown in Table 3, CG-TTS consistently improves preservation-related metrics, indicating that the benefits are not specific to a single editor. In particular, CG-TTS (SDXL) achieves the best L1, L2, CLIP-I, and DINO scores (Oquab et al., 2024) of 0.0395, 0.0134, 0.9412, and 0.9034, respectively, while increasing the GPT-5 and Gemini 2.5 Pro IC scores to 0.615 and 0.489, second only to ADE-CoT (Qu et al., 2026). Consistent improvements in L1, L2, and IC are also observed with BrushNet. CLIP-T exhibits a less consistent trend, decreasing slightly with BrushNet and improving only marginally with SDXL. This difference may arise because global text–image similarity is less sensitive to localized editing correctness than the instruction-compliance evaluators.

4.2 Ablations Analysis

Table 4 evaluates the components of CG-TTS on ReasonEdit-U with BrushNet. The ablation results indicate that the complementary roles of the components: the semantic contract specifies what to edit, the spatial contract constrains where the change may occur, and contract-guided verification determines whether the requested edit has succeeded and how a failure should be repaired. Removing the semantic contract reduces CLIPSim from 24.075 to 20.882 and PSNR from 37.964 to 31.217 dB. These degradation indicates that explicitly resolving the operation, source entity, and desired target helps reduce ambiguity that can affect both target realization and source preservation. Removing the spatial contract lowers PSNR to 34.870 dB and CLIPSim to 20.486, showing that explicit spatial constraints are important for localizing the requested change while limiting modifications to non-target

Table 3: Experimental results on HumanEdit-core (Bai et al., 2025). Scalar Best-of-8 controls use the same frozen BrushNet candidate pool and the same eight editor calls.

Method	L1 ↓	L2 ↓	CLIP-I ↑	DINO ↑	CLIP-T ↑	GPT-5 IC ↑	Gemini 2.5 Pro IC ↑
Existing Editing Methods							
InstructPix2Pix (Brooks et al., 2023)	0.1625	0.0570	0.7627	0.5349	0.2533	0.289	0.159
MGIE (Fu et al., 2024)	0.1294	0.0610	0.8670	0.7359	0.2589	0.116	0.071
HIVE (Zhang et al., 2024)	0.1162	0.0373	0.8441	0.7038	0.2563	0.237	0.129
MagicBrush (Zhang et al., 2023)	0.0760	0.0283	0.8946	0.8121	**0.2619**	0.293	0.224
SmartFreeEdit (Sun et al., 2025)	0.1794	0.0773	0.8829	0.8017	0.2556	0.301	0.149
Frozen Editors and TTS Baselines							
BrushNet (Ju et al., 2024a)	0.0732	0.0257	0.8890	0.7609	0.2231	0.396	0.214
SDXL Inpainting (Podell et al., 2024)	0.0483	0.0149	0.9186	0.8348	0.2119	0.299	0.329
ADE-CoT (Qu et al., 2026)	0.0465	0.0160	0.9400	0.8556	0.2114	**0.706**	**0.591**
ELECT (Kim et al., 2025a)	0.0855	0.0325	0.8679	0.7393	0.2106	0.433	0.284
Scalar Best-of-8 Controls (BrushNet; Same Candidate Pool)							
CLIP Best-of-8	0.0484	0.0181	0.8892	0.8067	0.2057	0.391	0.211
VIEScore Best-of-8	0.0461	0.0170	0.8915	0.8112	0.2142	0.423	0.236
EditScore Best-of-8	0.0452	0.0166	0.8940	0.8135	0.2128	0.438	0.247
Ours							
CG-TTS (BrushNet)	0.0467	0.0169	0.8904	0.8105	0.2159	0.442	0.233
Gain vs. BrushNet	+0.0265	+0.0088	+0.0014	+0.0496	−0.0072	+0.046	+0.019
CG-TTS (SDXL Inpainting)	**0.0395**	**0.0134**	**0.9412**	**0.9034**	0.2157	0.615	0.489
Gain vs. SDXL Inpainting	+0.0088	+0.0015	+0.0226	+0.0686	+0.0038	+0.316	+0.160

Table 4: Ablation study on ReasonEdit-U (Huang et al., 2024).

Variant	PSNR ↑	LPIPS ↓	SSIM ↑	CLIPSim ↑	GPT-5 IC ↑	Gemini IC ↑
CG-TTS (BrushNet)	**37.964**	**10.59**	**0.986**	**24.075**	**0.794**	**0.694**
BrushNet (Ju et al., 2024a)	25.393	46.58	0.797	21.755	0.500	0.425
w/o Semantic Contract	31.217	18.05	0.973	20.882	0.363	0.263
w/o Spatial Contract	34.870	12.74	0.980	20.486	0.369	0.244
w/o Verification	17.920	4.65	0.976	23.543	0.744	0.669
w/o SMR	35.004	11.69	0.983	22.118	0.763	0.661

content. Disabling contract-guided verification reduce PSNR to 17.920 dB and SSIM to 0.976, while CLIPSim decreases to 23.543. This result highlights the role of output-level verification: semantic and spatial contracts guide candidate generation, but do not by themselves determine whether an individual candidate has successfully realized the requested edit while preserving protected content. Although this variant obtains a lower LPIPS value (4.65 versus 10.59), the improvement is not accompanied by corresponding gains in PSNR, SSIM, or CLIPSim.

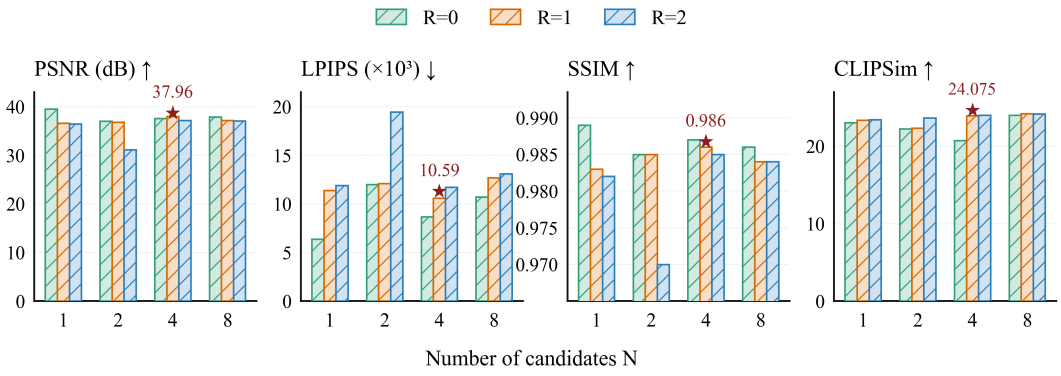

Figure 3: Comparison of candidate and repair configurations on ReasonEdit-U (Huang et al., 2024).

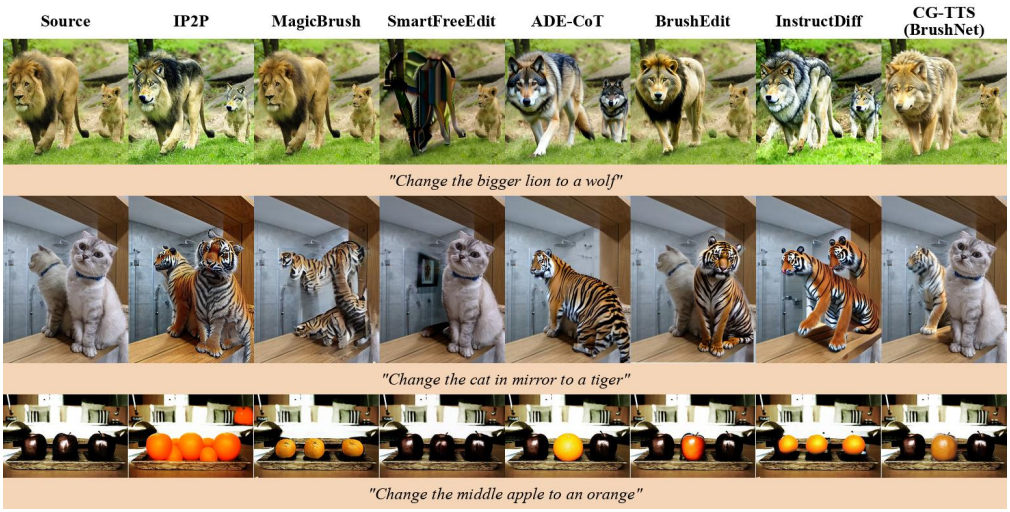

Figure 4: Qualitative comparisons on representative ReasonEdit (Huang et al., 2024) samples.

4.3 Hyperparameter Analysis

For sensitivity analysis, we further extend the default one-repair policy to configurations with up to two repair rounds. Figure 3 studies candidate scaling and repair allocation under different editor-call budgets. A configuration with N candidates per round and at most R repair rounds uses $B = N(1+R)$ maximum editor calls. A single failure-conditioned repair round improves instruction–crop alignment, whereas increasing candidate numbers or additional repair rounds provides limited benefit for preservation. For the default $N = 4, R = 1$ setting, CLIPSim improves from 20.737 to 24.075 and PSNR from 37.587 to 37.964 dB compared with no repair, while maintaining similar background preservation. Additional scaling to $N = 8$ or $R = 2$ yields diminishing returns. Since repair is triggered only for failed initial candidates, it is activated for 32/131 ReasonEdit-U examples, resulting in 4.98 average editor calls per example, which is 37.8% fewer than uniform Best-of-8.

4.4 Qualitative Results

Figure 4 compares CG-TTS with representative baselines on spatial selection, reflection, and multi-object editing cases. In the first example, CG-TTS changes only the specified animal while preserving the other instance and surrounding scene, whereas several baselines modify both objects or leave the requested change incomplete. In the mirror-reflection example, CG-TTS edits the reflected object without altering the corresponding foreground object, demonstrating its ability to distinguish target and non-target instances. In the multi-object replacement example, CG-TTS replaces only the requested object while preserving neighboring objects and scene layout. These examples illustrate that contract-guided verification helps CG-TTS combine instruction-specific target selection with preservation of non-target content.

5 Conclusion

In this work, we propose CG-TTS, a training-free framework for test-time scaling in image editing. CG-TTS constructs an edit contract that specifies semantic requirements, spatial constraints, and mandatory verification checks, and uses the same contract for candidate generation and contract-guided verification. Candidates are accepted only when all active mandatory checks pass, and failure diagnosis guides a single repair round when no candidate satisfies the contract. CG-TTS nevertheless depends on accurate contract construction and target localization, and errors in either can propagate through generation, verification, and repair. Moreover, passing the instantiated checks does not guarantee semantic correctness or artifact-free rendering, and the framework remains limited by the edits its frozen editor can produce within the available editor-call budget. Future work will investigate better-calibrated clause-level verification and adaptive allocation to uncertain or failed requirements.

AI USE STATEMENT

Generative AI tools were used to improve the clarity and readability of the manuscript. We also used a multimodal large language model for the semantic-spatial contract construction. All AI-assisted generated contents were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with AI assistance.

REFERENCES

Jinbin Bai, Wei Chow, Ling Yang, Xiangtai Li, Juncheng Li, Hanwang Zhang, and Shuicheng Yan. Humanedit: A high-quality human-rewarded dataset for instruction-based image editing, 2025. 2, 3, 6, 7, 8, 16

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space, 2025. 2

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 1, 6, 7, 8

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. MasaCtrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *ICCV*, 2023. 5

Sherry X. Chen, Yi Wei, Luowei Zhou, and Suren Kumar. ADIEE: Automatic dataset creation and scorer for instruction-guided image editing evaluation. In *ICCV*, 2025. 3, 5

Jaemin Cho, Yushi Hu, Roopal Garg, Peter Anderson, Ranjay Krishna, Jason Baldridge, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *ICLR*, 2024. 1, 3

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. 7

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *ICLR*, 2024. 1

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 *ICLR*, 2024. 2, 6, 8

Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Houqiang Li, Han Hu, et al. Instructdiffusion: A generalist modeling interface for vision tasks. In *CVPR*, 2024. 6, 7

Hanzhong Guo, Jie Wu, Jie Liu, Yu Gao, Zilyu Ye, Linxiao Yuan, Xionghui Wang, Yizhou Yu, and Weilin Huang. Leveraging verifier-based reinforcement learning in image editing. In *CVPR*, 2026. 2, 3

Seyed Mohammad Hadi Hosseini, Amir Mohammad Izadi, Ali Abdollahi, Armin Saghafian, and Mahdieh Soleymani Baghshah. T2I-FineEval: Fine-grained compositional metric for text-to-image evaluation, 2025. 2

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. TIFA: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *ICCV*, 2023. 1, 3

Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, et al. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *CVPR*, 2024. 2, 3, 6, 7, 8, 9, 16, 17, 18

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 *ECCV*, 2024a. 2, 5, 6, 7, 8, 13

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Pnp inversion: Boosting diffusion-based editing with 3 lines of code. In *ICLR*, 2024b. 1

Kat Kampf and Nicole Brichtova. Experiment with Gemini 2.0 Flash native image generation. Google Developers Blog, mar 2025. 6

Joowon Kim, Ziseok Lee, Donghyeon Cho, Sanghyun Jo, Yeonsung Jung, Kyungsu Kim, and Eunho Yang. Early timestep zero-shot candidate selection for instruction-guided image editing, 2025a. 1, 3, 5, 7, 8

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. In *ICLR*, 2025b. 1

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *NeurIPS*, 2023. 1, 2, 3

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. LISA: Reasoning segmentation via large language model. In *CVPR*, 2024. 4, 13, 14

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Arsh Koneru, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Reflect-DiT: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. In *ICCV*, 2025. 1, 3

Yaowei Li, Yuxuan Bian, Xuan Ju, Zhaoyang Zhang, Junhao Zhuang, Ying Shan, Yuexian Zou, and Qiang Xu. Brushedit: All-in-one image inpainting and editing. *IEEE TPAMI*, 2026. 2, 6, 7

Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, et al. Jarvisevo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. In *CVPR*, 2026. 1

Runzhou Liu, Hailey Weingord, Sejal Mittal, Prakhar Dungarwal, Anusha Nandula, Bo Ni, Samyadeep Basu, Hongjie Chen, Nesreen K. Ahmed, Li Li, et al. Human-aligned MLLM judges for fine-grained image editing evaluation: A benchmark, framework, and analysis, 2026. 5

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *ECCV*, 2024. 4

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025. 1, 3

Yingmao Miao, Pengfei Zhang, Xiaochen Lv, Meng Yu, Lei Sun, Xiangxiang Chu, Chao Shen, and Chenhao Lin. Evaluation-verification reward for consistent multi-reference image editing, 2026. 3

OpenAI. GPT-4o system card, 2024. 3, 6

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, et al. DINOv2: Learning robust visual features without supervision, 2024. 7

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 2, 6, 7, 8, 13

Xiangyan Qu, Zhenlong Yuan, Jing Tang, Rui Chen, Datao Tang, Meng Yu, Lei Sun, Yancheng Bai, Xiangxiang Chu, Gaopeng Gou, Gang Xiong, and Yujun Cai. From scale to speed: Adaptive test-time scaling for image editing, 2026. 1, 2, 3, 5, 7, 8

Qianqian Sun, Jixiang Luo, Dell Zhang, and Xuelong Li. Smartfreeedit: Mask-free spatial-aware image editing with complex instruction understanding. In *ACM MM*, 2025. 2, 6, 7, 8

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023. 5

Juntong Wang, Jiarui Wang, Huiyu Duan, Jiaxiang Kang, Guangtao Zhai, and Xiongkuo Min. I2I-Bench: A comprehensive benchmark suite for image-to-image editing models. In *CVPR*, 2026. 1

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6

Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. OmniGen: Unified image generation. In *CVPR*, 2025. 2

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *NeurIPS*, 2023. 1, 2, 3

Michal Yarom, Yonatan Bitton, Soravit Changpinyo, Roee Aharoni, Jonathan Herzig, Oran Lang, Eran Ofek, and Idan Szpektor. What you see is what you read? improving text-image alignment evaluation. In *NeurIPS*, 2023. 2

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023. 1, 6, 7, 8

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, Caiming Xiong, and Ran Xu. HIVE: Harnessing human feedback for instructional visual editing. In *CVPR*, 2024. 6, 8

Zijun Zhou, Yingying Deng, Xiangyu He, Weiming Dong, and Fan Tang. Multi-turn consistent image editing, 2025. 1

A Implementation Details

A.1 Frozen Components and Inference Settings

Frozen backbones and default settings CG-TTS is evaluated with frozen BrushNet (Ju et al., 2024a) and SDXL-Inpainting (Podell et al., 2024) backbones. No model parameters are updated during inference. For SDXL-Inpainting, the inpainting strength is set to 0.9999. We use frozen LISA (Lai et al., 2024) for instruction-grounded target localization.

Candidates are generated at the original image resolution. Metric preprocessing follows the official benchmark evaluation protocols, with ReasonEdit metric inputs resized to 256×256. Both ReasonEdit and HumanEdit-core use MLLM-assisted contract construction. The MLLM outputs structured semantic information, which is converted by deterministic rules into the semantic–spatial contract used by CG-TTS. Benchmark annotations are not provided during inference. Table 5 summarizes the default inference configuration.

SMR Implementation Details This section specifies the feature extraction, routing regions, and residual updates used to implement SMR in Section 3.3. SMR modifies conditioning-image features within the frozen editor, with memory access and update regions determined by the operation and spatial masks in Eq. equation 2.

The source features $X = \{x_j\}_{j=1}^{N}$ are extracted from the output of the frozen VAE encoder bottleneck (`encoder.mid_block`), before output normalization and latent projection. For an input of size $H \times W$, the feature grid has resolution $H/8 \times W/8$, with 512 channels in the BrushNet VAE. The memory image is I_{img}, or the pre-filled conditioning image when pre-filling is enabled. Current query features are obtained at the same encoder location. Both BrushNet and SDXL-Inpainting apply SMR along their conditioning-image encoding paths, once before denoising; the resulting conditioning latent is reused throughout the denoising schedule.

In Eqs. equation 5 and equation 6, current features provide queries and memory features directly provide keys and values, without learned projections. Queries and keys are normalized for cosine similarity, while aggregation uses the original memory values. We use $k = 16$, $\tau = 0.10$, and $\lambda_{\text{geo}} = 0.20/0.12 = 5/3$, with coordinates normalized to $[0, 1]$ along each axis.

Table 6 summarizes the admissible memory sources, update regions, and coefficients for each operation. For a feature q_i before routing and a retrieved feature R_i, all positive routes use

$$q_i^{\text{out}} = q_i + \lambda(R_i - q_i), \tag{8}$$

where λ is the route coefficient. Edit-positive routes apply 3×3 average pooling to memory features before retrieval. For removal, an optional background-support prior in the retrieval score favors suitable context entries. Protected-region features remain available for retrieval but receive no routing updates. Target-dependent routes are active only when eligible memory is available, and their coefficients are scaled by its confidence $c_{\text{tgt}} \in [0, 1]$. For replacement, source-negative routing additionally suppresses the source component not shared with the target memory. All retrievals and scaling factors use features before routing.

Source-negative routing requires source-exclusive attributes and is disabled when distinct canonicalized source and target classes belong to the same predefined semantic group. When disabled, only the target-positive term remains within M_{edit}. Context-positive routing acts separately on the disjoint region M_{context}.

SMR operates within each invocation of G and does not introduce an additional editor invocation. Contract construction, localization, verification, and SMR routing are auxiliary inference operations and are not counted as editor calls. We therefore report editor-call budgets separately to characterize candidate-generation effort rather than total end-to-end computation or latency.

A.2 Contract Construction and Verification Details

Section 3.2 defines the semantic-spatial contract $C = (\mathcal{S}, \mathcal{M})$ conceptually. Here, we describe how the contract is instantiated and how its requirements are evaluated by the verifier in Section 3.4. For each example, only requirements instantiated by the resulting contract are evaluated. Each

Table 5: Default inference settings.

Setting	Value
Candidates per round	4
Maximum repair rounds	1
Maximum editor calls	8
Diffusion steps	50
SDXL inpainting strength	0.9999

Table 6: Operation-specific SMR routing configurations. The table summarizes the memory sources and update regions used for different editing operations.

Operation	Route	Memory source	Update region	
Remove	Edit-positive	Smoothed source context	M_{edit}	0.65
Replace	Context-positive	Source context and protected regions	M_{context}	0.10
	Target-positive	Target-edit memory	M_{edit}	0.18
	Source-negative	Source edit region, relative to target memory	M_{edit}	0.16
Add	Context-positive	Source context and protected regions	M_{context}	0.08
	Target-positive	Target-edit memory	M_{edit}	0.18
Background	Context-positive	Source context and protected regions	M_{context}	0.10
	Edit-positive	Smoothed source context and protected regions	M_{edit}	0.12

mandatory check returns `PASS`, `FAIL`, or `UNCERTAIN`, and a candidate is accepted only when all active mandatory checks return `PASS`.

Contract construction. The same contract construction pipeline is applied to both benchmarks. The contract is constructed in two steps. First, a frozen MLLM receives the source image I_{img} and the original editing instruction and produces a symbolic description of the requested transformation. Deterministic post-processing then normalizes the edit operation and canonicalizes the target description, including source and requested target information, disambiguating cues, and phrases required by subsequent localization and generation. Accordingly, $\mathcal{S}$ is implemented as an extensible structured field set whose instantiated fields depend on the requested edit, rather than as a fixed tuple.

Second, the semantic description is grounded into the spatial component $\mathcal{M}$. Frozen LISA (Lai et al., 2024) provides the primary instruction-grounded localization, while instruction-derived spatial and appearance cues are used to disambiguate candidate instances when needed. Let S denote the resulting source region, E_0 the operation-specific initial must-change region, A_0 the initial allowed-change region, and P_0 the explicitly protected content. With $P_{\text{out}} = P_0 \setminus S$, the implementation-level regions are constructed as

$$E = (E_0 \setminus P_{\text{out}}) \cup S,\ A \quad = (A_0 \cup S \cup E) \setminus P_{\text{out}},\ P = (\overline{A} \cup P_0) \setminus S, \tag{9}$$

where the union with S in the first line is omitted for addition edits, which do not modify an existing source object. These regions instantiate the role-specific masks defined in Section 3.2:

$$M_{\text{edit}} = E, \qquad M_{\text{context}} = A \setminus E, \qquad M_{\text{allow}} = A, \qquad M_{\text{protect}} = P. \tag{10}$$

Operation-specific construction may additionally instantiate spatial regions required by the instruction, such as an anchor or destination region.

The MLLM output is therefore used as semantic evidence for contract construction rather than as an executable verification policy. Malformed outputs or recoverable missing fields are handled by

deterministic normalization or fallback construction, while a required edit region that cannot be localized is not allowed to proceed as a valid spatial contract. The active mandatory checks are instantiated deterministically from the normalized operation, instruction, and resulting contract; MLLM-proposed verification fields do not directly activate verifier checks. The resulting contract configures candidate generation in Section 3.3 and verification in Section 3.4.

Actual-change verification. The actual-change verification provides spatial evidence for whether a candidate modifies the intended region while preserving protected content. For each candidate Y_i, we compare it with the source image I_{img}. For a pixel u, the normalized color and intensity differences are

$$d_{\text{Lab}}(u) = \frac{\|\text{Lab}(Y_i)_u - \text{Lab}(I_{\text{img}})_u\|_2}{255\sqrt{3}}, \qquad d_{\text{gray}}(u) = \frac{|\text{gray}(Y_i)_u - \text{gray}(I_{\text{img}})_u|}{255}. \tag{11}$$

The continuous change score is

$$d(u) = \text{clip}_{[0,1]}\left(0.75\, d_{\text{Lab}}(u) + 0.25\, d_{\text{gray}}(u)\right). \tag{12}$$

We convert the continuous score into an initial binary change map using the adaptive threshold

$$\tau_D = \text{clip}_{[0.10,\,0.30]}\left(Q_{0.975}(d)\right), \qquad D_0 = \{u : d(u) \geq \tau_D\}, \tag{13}$$

where $Q_{0.975}(d)$ denotes the 97.5th percentile of the change scores over all image pixels. We then apply a 3×3 morphological opening followed by a 5×5 closing and remove 8-connected components smaller than $\max(8, \lfloor 0.00015HW \rfloor)$ pixels, yielding the cleaned change region $\widehat{D}$.

Based on $\widehat{D}$ and the continuous score d, the verifier computes

$$\text{CLP} = \begin{cases} \dfrac{|\widehat{D} \cap M_{\text{allow}}|}{|\widehat{D}|}, & |\widehat{D}| > 0, \\ 1, & |\widehat{D}| = 0, \end{cases} \tag{14}$$

$$\text{TCR} = \frac{|\widehat{D} \cap M_{\text{edit}}|}{\max\{1, |M_{\text{edit}}|\}}, \tag{15}$$

$$\text{PPS} = 1 - \frac{\sum_{u \in M_{\text{protect}}} d(u)}{\max\{1, |M_{\text{protect}}|\}}. \tag{16}$$

CLP measures the localization of detected changes within the allowed region, TCR measures change coverage within the edit region, and PPS measures preservation within the protected region. All three scores lie in $[0, 1]$, with higher values indicating better localization, coverage, and preservation. An empty mandatory edit region is rejected before generation, and PPS is set to 1 when M_{protect} is empty.

For local edits, the shared spatial requirements are

$$\text{CLP} \geq 0.65, \qquad \text{TCR} \geq \theta_{\text{TCR}}(o, a), \qquad \text{PPS} \geq 0.88, \tag{17}$$

where o denotes the normalized verification operation and

$$a = \frac{|M_{\text{edit}}|}{HW} \tag{18}$$

is the relative edit-region area. The operation-dependent TCR thresholds are reported together with the corresponding mandatory checks in Table 7.

To prevent degenerate spatial passes, the total changed fraction must additionally satisfy

$$\frac{|\widehat{D}|}{HW} \geq \begin{cases} 0.0003, & o = \text{remove and } a < 0.01, \\ 0.0005, & \text{otherwise}, \end{cases} \tag{19}$$

so that an unchanged candidate cannot pass through the empty-change convention used by CLP. For local edits, detectable change within the protected region must also satisfy

$$\frac{|\widehat{D} \cap M_{\text{protect}}|}{\max\{1, |M_{\text{protect}}|\}} \leq 0.035. \tag{20}$$

These spatial definitions and base thresholds are shared across generation rounds. When the strict verifier is enabled for SDXL-Inpainting, the backend additionally applies the hard preservation requirement $\text{PPS} \geq 0.97$. This backend-specific gate supplements rather than replaces the common PPS requirement above; BrushNet does not use the additional 0.97 threshold.

Operation-specific verification. The spatial requirements in Eq. equation 17 provide the first-level mandatory constraints, while operation-specific checks instantiate additional semantic requirements from the contract. A candidate is accepted only when both spatial and semantic requirements are satisfied. Spatial change alone does not establish that the requested semantic result has been achieved. The verifier therefore instantiates a deterministic set of operation-specific mandatory checks from the normalized operation, instruction, and contract. Table 7 summarizes these checks together with the operation-dependent TCR thresholds rules used for local edits.

For removal and replacement, the TCR threshold varies with the relative edit-region area a defined in Eq. equation 18; the corresponding area ranges and threshold values are given directly in Table 7. The remaining local operations use the fixed TCR thresholds reported in the same table. Scene editing does not use a TCR threshold because the required change is evaluated over the background region rather than a localized target mask.

Each instantiated requirement corresponds to one mandatory check $q \in \mathcal{Q}_{\text{mand}}(C)$. Evidence associated with that requirement is aggregated before assigning $v_q(Y_i, C) \in \{\texttt{PASS}, \texttt{FAIL}, \texttt{UNCERTAIN}\}$. Multiple evidence sources may contribute to the same check, but they are combined before the final status is assigned rather than being used to override a previously failed mandatory requirement. A conclusive violation yields `FAIL`, whereas insufficient evidence for a required judgment yields `UNCERTAIN`.

Following the conjunctive acceptance rule in Section 3.4, Y_i is accepted iff every mandatory check $q \in \mathcal{Q}_{\text{mand}}(C)$ returns `PASS`. A remaining `FAIL` or `UNCERTAIN` therefore makes the candidate inadmissible and cannot be compensated for by stronger evidence from other checks. Soft similarity, preservation, and visual-quality signals are used only to rank candidates that satisfy all mandatory requirements.

Failure-conditioned repair. If no initial candidate passes verification, CG-TTS normalizes the detected failures and applies a fixed repair policy to generate recovery candidates. The repair policy updates the generation configuration while keeping the semantic–spatial contract and mandatory verification criteria unchanged. Recovery candidates are evaluated using the same verifier, and the final output follows the same acceptance procedure. Repair changes the generation configuration in Eq. equation 7, but does not modify the semantic–spatial contract or the mandatory verification criteria. Repaired candidates are therefore evaluated by the same checks and thresholds as the initial candidates. Ambiguous or conflicting contracts do not trigger repair.

A.3 Dataset Composition and Metrics

ReasonEdit (Huang et al., 2024) contains 191 examples, including 131 examples in the Understanding (U) split and 60 examples in the Reasoning (R) split. The Understanding split includes spatial relations, relative size, mirror reflection, color editing, and multi-object instructions. HumanEdit-core (Bai et al., 2025) contains 400 high-resolution examples covering action, addition, counting, relation, removal, and replacement edits. The category distributions are summarized in Table 8. Experimental analysis of repair behavior is provided in Appendix B.2.

For ReasonEdit, we follow the official benchmark evaluation protocol. PSNR, LPIPS, and SSIM evaluate preservation of source content outside the edited region using the benchmark evaluation masks, while CLIPSim evaluates instruction–crop alignment within the edited region. Metric inputs are resized to 256×256 following the official evaluation protocol. In controlled analyses, U-PSNR

Table 7: Operation-specific mandatory checks and TCR thresholds used by the verifier.

Operation	Mandatory checks	θ_{TCR}
Remove	Image changed; sufficient source-region change; requested target absent.	0.12/0.18/0.20
Replace	Image changed; sufficient source-region change; original source absent; requested replacement present.	0.15/0.20/0.25
Add	Image changed; change in the allowed new-object/anchor region; requested object present. Count, attribute, and spatial-relation checks are additionally activated when specified.	0.08
Layout	Image changed; source region sufficiently cleared; destination region occupied as required.	0.10
Size	Image changed; target identity retained; boundary region changed. The requested grow/shrink direction is additionally checked.	0.10
Color	Image changed; sufficient source-region change; requested color realized; source identity retained.	0.10
Attribute	Image changed; sufficient source-region change; requested attribute realized; target identity retained.	0.10
Text	Image changed; sufficient text-region change; normalized OCR output matches the requested text.	0.10
Scene	Image changed; sufficient background-region change; requested background semantics satisfied.	—

Table 8: Dataset composition. HumanEdit-core category names follow the original benchmark.

Dataset	Category	Examples
ReasonEdit-U	Left/right	28
	Relative size	22
	Mirror	30
	Color	27
	Multiple objects	24
	Total	131
HumanEdit-core	Action	37
	Addition	30
	Counting	20
	Relation	28
	Removal	188
	Replacement	97
	Total	400

and U-CLIP refer specifically to these metrics computed on the ReasonEdit-U split. All image-metric evaluations are conducted on the full 400 HumanEdit-core examples for both BrushNet and SDXL-Inpainting. The Ins-align metric is omitted because the required human-authored questions and reference answers are unavailable.

Instruction compliance (IC) is evaluated separately through post-hoc MLLM judgments and is not used as a benchmark image metric. The supplementary MLLM-judge audit in Appendix B.1 adopts a separate five-criterion evaluation protocol and is reported independently.

B Additional Experiments

B.1 Supplementary MLLM-Judge Audits

Instruction compliance (IC) in the main experiments is assessed by MLLM-based evaluation. To examine the consistency of this evaluation, we conduct a blind multi-judge audit on ReasonEdit (Huang et al., 2024). Specifically, we evaluate the generated outputs of CG-TTS and representative baselines using three independent MLLM judges: Gemini 2.5 Pro, GPT-5, and Qwen3-VL-Plus. The judges receive the same source image, editing instruction, and candidate outputs without access to

Table 9: Blind MLLM-judge evaluation on ReasonEdit (Huang et al., 2024).

Judge	Method	TC ↑	WTC ↓	NTP ↑	AS ↓	IC ↑
Understanding (U), $N = 80$						
Gemini 2.5 Pro	CG-TTS (BrushNet)	0.6125	0.2500	0.7190	0.5815	0.6940
	CG-TTS (SDXL)	0.4440	0.2625	0.6875	0.7250	0.4190
	SmartFreeEdit	0.2625	0.2625	0.7065	0.7250	0.2375
GPT-5	CG-TTS (BrushNet)	0.8065	0.0690	0.9500	0.4625	0.7940
	CG-TTS (SDXL)	0.5250	0.1500	0.8500	0.7500	0.5250
	SmartFreeEdit	0.2125	0.4375	0.6815	0.7815	0.2125
Qwen3 VL-Plus	CG-TTS (BrushNet)	0.8500	0.0565	0.9440	0.5440	0.8125
	CG-TTS (SDXL)	0.5065	0.2690	0.7690	0.7940	0.4875
	SmartFreeEdit	0.1815	0.6065	0.4625	0.8940	0.1625
Reasoning (R), $N = 60$						
Gemini 2.5 Pro	CG-TTS (BrushNet)	0.3835	0.2375	0.7500	0.7000	0.3410
	CG-TTS (SDXL)	0.3915	0.2585	0.7165	0.7250	0.3500
	SmartFreeEdit	0.3835	0.3415	0.6500	0.6250	0.3415
GPT-5	CG-TTS (BrushNet)	0.4710	0.1710	0.8500	0.7750	0.4585
	CG-TTS (SDXL)	0.3415	0.2585	0.8665	0.5915	0.3335
	SmartFreeEdit	0.2750	0.3415	0.8415	0.5835	0.2750

Table 10: Breakdown of verifier-triggered repair cases.

Failure evidence	Cases	Contract pass after repair
Target-presence / realization failure	26	6
Insufficient TCR only	1	0
Over-change	0	–
Weak-pass recovery trigger	5	5
Total	32	11

method identities. The evaluation criteria follow the main IC assessment and consider target-change consistency, non-target preservation, and artifact-related quality.

Table 9summarizes the results across different judges and subsets. CG-TTS shows consistent relative comparisons across judges, while achieving stronger target-change consistency and non-target preservation in most settings. The additional Qwen3-VL-Plus evaluation shows similar relative comparisons, indicating that the observed improvement is not tied to a single MLLM judge.

B.2 Failure-conditioned Repair Analysis

We further analyze the behavior of failure-conditioned repair on initially rejected candidates. Unlike generic retry, CG-TTS uses verifier-derived evidence to trigger a deterministic repair policy, while keeping the semantic-spatial contract and mandatory verification criteria unchanged.

Beyond final acceptance, repair modifies the verification state: verifier gate PASS increases from 3/32 to 7/32, TCR PASS increases from 9/32 to 12/32, and target-presence PASS increases from 2/32 to 7/32, while PPS remains unchanged at 32/32. These results indicate that repair primarily addresses unmet target realization requirements rather than trading preservation for stronger edits.

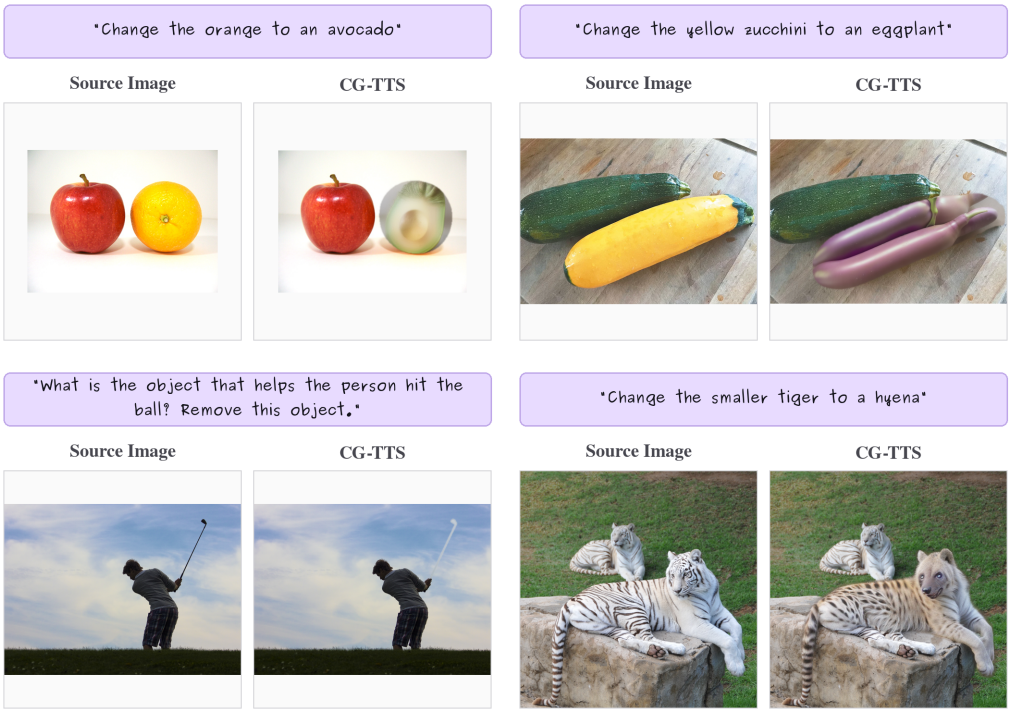

Figure 5: Representative failure cases of CG-TTS on ReasonEdit.

B.3 Failure Cases

Figure 5 presents representative failure cases where candidates pass the instantiated contract checks but still exhibit mismatches with the intended edit semantics. These cases reveal limitations in the coverage and granularity of the current verification design. For example, the verifier may confirm the presence of the requested target or sufficient regional changes while missing fine-grained attribute inconsistencies, incomplete object removal, or imperfect target realization. Such cases indicate that contract acceptance provides explicit guarantees over the encoded requirements, but its reliability is bounded by the completeness of the contract specification and verification criteria. Improving fine-grained semantic verification remains an important direction for future work.

B.4 Additional Qualitative Comparisons

Figure 6 presents additional qualitative comparisons on ReasonEdit, including spatial relations, mirror reflections, multi-object selection, and relative-size edits. These examples highlight the challenge of identifying the intended target while preserving unrelated content. CG-TTS more consistently localizes modifications to the requested instance, although imperfect appearance matching and residual source attributes remain possible due to limitations of the frozen editor.

C AI use statement

In this work, we used ChatGPT to provide feedback on research methodology and experimental comparisons, assist with translation, and improve the clarity, organization, and conciseness of the manuscript. We reviewed the AI-assisted text and analyses, checked reported numerical comparisons against the experimental results, and assessed the resulting claims against the available evidence. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.



Figure 6: Additional qualitative comparisons on ReasonEdit.