Towards Robust Sequential Decomposition for Complex Image Editing

Supplementary Material

A. Synthetic Data Generation

A.1. Scene Initialization

We implement our synthetic data generation pipeline on top of BlenderProc [11], a Blender-based pipeline for photorealistic image rendering. To create a photorealistic 3D scene, we first construct an empty rectangular room of size 8x8x3, in which we create fundamental structural planes, including a floor, a ceiling, and four walls. These structural planes are placed in a coordinate system with the floor at $z = 0$ and walls aligned to the room boundaries. Additionally, a light plane (embedded in the ceiling) is added to ensure basic illumination. Then we randomly sample textures from CC0 Textures for the floor and walls.

After the base empty room is constructed, we start to populate it with movable objects selected from an asset catalog, which is compiled beforehand to list available objects with their metadata, including semantic labels, categories, and estimated sizes. To ensure every scene has at least one object that can serve as a supporter to other objects, we first sample an object from predefined supporter categories (*e.g.*, tables, desks), and place the object at the center of the room. Before we sample other room objects, we first compute a safe object size range based on the room dimensions, object count, and camera field of view to prevent choosing objects that are too large to fit comfortably in the scene, with which we can filter objects according to their estimated sizes from the catalog. We then sample $N = 9$ unique objects in a single batch from the catalog, and place them sequentially into a constrained area where the camera can easily obtain good visibility on all objects. During this process, each placement finds a collision-free position for the object on the floor using collision checkers from BlenderProc. If no collision-free positions can be found for the objects, we skip the placement of the current object and move on to the next one. When the whole batch of selected objects are processed, we run physics simulation for all objects simultaneously to improve scene realism. Once the scene are populated with objects, we sample a camera pose that provides a good coverage of scene content, ensuring the spawned objects are adequately visible and well-framed in the rendered image. By default, we render every scene into a 512×512 image with 256 samples for each pixel.

Furthermore, we record the metadata (*e.g.*, object labels, categories, poses, sizes, etc.), and the spatial relationships between objects into a registry for all loaded objects in the scene, providing a clearly documented initial state where the subsequent editing process can operate on.

A.2. Editing Operation Design

To enable complex edits in our pipeline, we predefined 10 editing operations in total, each associated with a set of parameters. Each operation either performs an edit on a single object in the scene, changes the camera position, or changes the scene background. We outline the predefined editing operations in detail below:

Object Addition introduces a new object of a novel category under a specified size constraint into the scene to encourage diversity. We support two modes for regular object addition: adding an object on top of a supporter object or near another object on the floor. Additionally, to enable position reference dependency, we allow the operation to directly take in a 3D coordinate of the initial location of an object that has been moved. When adding the object, we also run collision and visibility checking to maintain the visual quality of the rendered image.

Object Removal deletes an object from the scene. However, we do not allow the structural planes (*e.g.*, floors and walls) or objects that are supporting other objects to be removed.

Object Translation repositions a non-supporter object while maintaining collision-free placement and camera visibility. We support three regular translation modes, allowing the target object to move by direction (left/right/forward/backward), near a reference object with a distance constraint, or onto another object. Similarly, we allow the operation to take in 3D coordinates directly for position reference.

Object Rotation rotates the object around its vertical axis, with a degree sampled from predefined candidates (60°, 90°, 120°, 180°) and a direction sampled from "clockwise" or "counter-clockwise".

Object Replacement replaces an existing object with a different one sampled from the catalog. The newly placed object inherits the xy location of the original objects. We also run collision checks for the new object and perform slight adjustments if collisions occur.

Color Change replace the color of an existing object with new one selected from a predefined palette of 12 colors, which includes "red", "blue", "green", "yellow", "orange", "purple", "pink", "white", "black", "gray", "brown", and "beige".

Material Change replace the material of an existing object with a new one chosen from a predefined set of 5 materials, which includes "wood", "metal", "plastic", "fabric", and "glass".

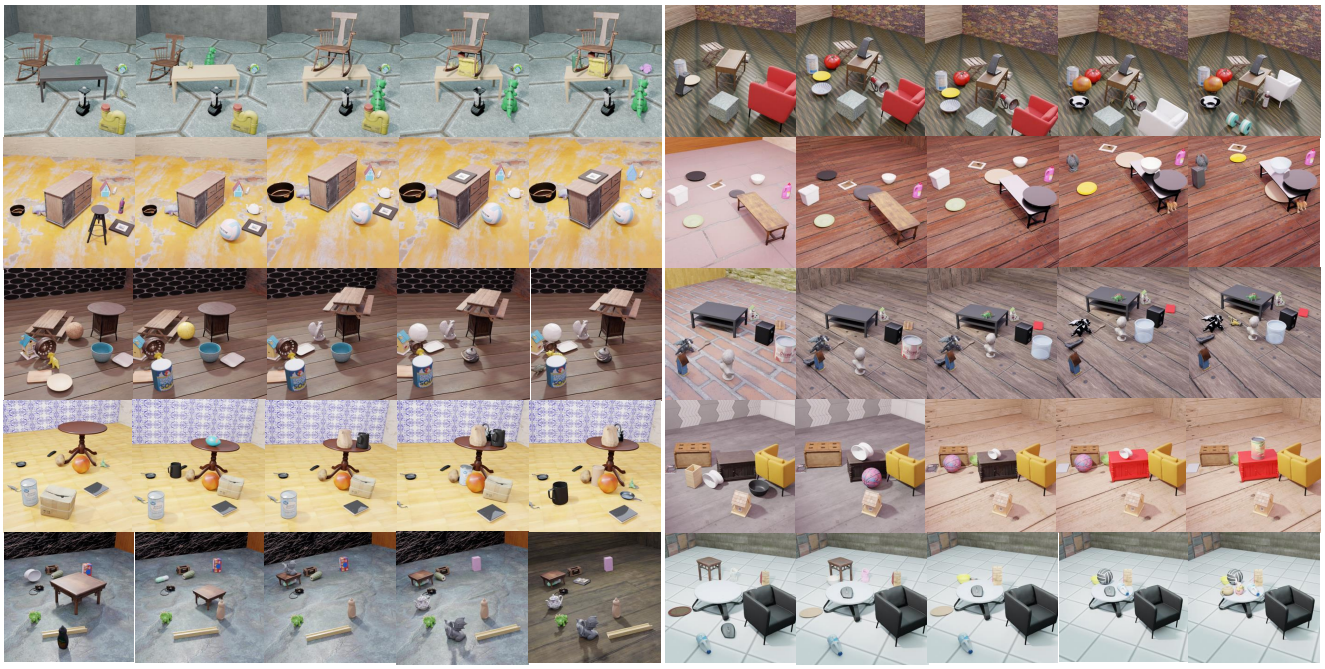


Figure A1. **Visualizations of constructed editing sequence samples**

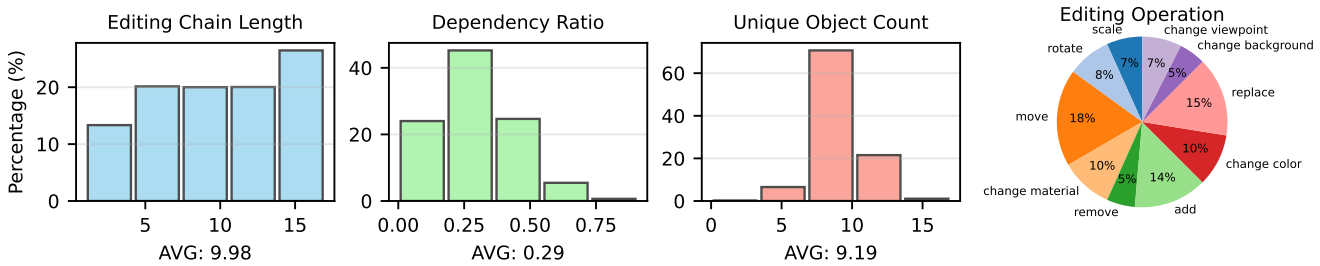


Figure A2. **Statistical analysis of synthetic editing sequences**

Size Change applies uniform size transformations to an existing object with a scale sampled from predefined candidates (0.6, 0.8, 1.2, 1.5, 1.8) with collision checks.

Viewpoint Change includes four types of camera pose transformation: (1) translation (0.4m forward or backward); (2) panning (0.4m left or right); (3) tilting (5° up or down); (4) yawing (5° left or right). Meanwhile, the visibility of all objects should remain the same after the camera pose transformation.

Background Change modifies the visual appearances of structural surfaces (*e.g.*, floors and walls) by replacing their textures with a new material sampled from CC0 Textures. Every background change operation can either change the texture of the floor or both the floor and the walls.

Dependency Injection. We maintain a detailed log of all executed editing operations when constructing editing chains. This log records each operation's type, sampled parameters, and the resulting changes to object states (*e.g.*, updated 3D coordinates). Our dependency injector builds directly on this editing history. For every newly sampled operation (excluding viewpoint and background changes), we decide whether to inject one of two inter-step dependencies, operation reference or position reference, with probability $p_d = 0.5$. Concretely, the task construction begins with only operation-reference injection enabled: for every operation after the first, there is a 50% chance that its instruction will be augmented with reference-based object descriptions. Once an object has been removed or repositioned, position-reference injection becomes available, and both dependency types are then sampled with equal probability (25% each).

A.3. Instruction Generation

We use a library of hand-crafted templates to generate natural language instructions for editing operations. Each template contains placeholders that get filled with operation-specific parameters like object names, colors, directions, and action verbs. Action verbs are randomized to prevent repetitive language. For example, object addition can use

Table A1. **GPT-5 prompts for two-step evaluation**

System Prompt	You are an expert evaluator for image editing. You will first be provided with a complex editing instruction and the source image to be edited, please analyze the complex task by rigorously tracking the editing transitions, and list all modifications that need to be reflected in the final edited image. Then you will be given an edited image for evaluation. You will need to examine carefully if the edited image accurately reflects all the required changes according to your reasoning. A successful edit should not have any extra changes that are not required by the instruction. The edited image should have minimal changes to reflect the modifications from the instruction. You need to provide a final rating for the editing result from 0 to 10, with 10 being the perfect edit that precisely reflects all changes while preserving everything else. Conclude your evaluation with the score wrapped with <s>...</s>.
Step-1 Prompt	Please analyze the modifications that should be reflected in the final image by strictly reasoning over the editing trace, given the source image and the complex instruction. Modifications include but are not limited to object placement, object removal, object modification, background change, and camera movement.
Step-1 Input	Source Image I_{src}, Complex Instruction T_c
Step-2 Prompt	Please evaluate whether the edited image accurately reflects the modifications specified in the complex editing instruction based on your previous reasoning. Be critical of the changes made during the editing process. Conclude your assessment with a final score wrapped with <s>...<\s>.
Step-2 Input	Generated editing result I_{gen}
Output Example	...<s>5.0</s>

"Add", "Place", or "Put" as the opening verb, object translation alternates between "Move" and "Relocate", and color changes use "Recolor" or "Paint". This randomization enables natural variations that improve instruction diversity. For inter-step dependency, (1) operation reference connects the current edit to a recently manipulated object using temporal descriptions like "the object that was moved" or "the object that was recolored", which anonymizes the object identity, using only "object" or "the object" to force models to track which object was affected by the referenced operation. When multiple objects share the same history operations (*e.g.*, more than one objects were moved), we enrich the referenced action verb with attribute details to ensure the reference unambiguously identifies the intended target. For instance, if both a chair and a book were moved by prior edits, referring to "the object that was moved" would be ambiguous, so the instruction will be further enriched with "the object that was moved onto the desk" to distinguish between them. (2) Position reference encodes spatial memory by referring to locations where objects were originally placed. These instructions use temporal markers to clarify the reference point: "Move the lamp to where the vase originally was" or "Add a book where the cup initially was".

A.4. Heuristic Filters for Data Generation

To avoid untraceable transitions that trivialize the task in single-turn settings, we heuristically filter out operation sequences whose later steps depend on transient intermediate states that can not be inferred without step-by-step visualization. For instance, a sequence that adds an object and subsequently rotates it is excluded, since the rotation relies on the object's intermediate orientation introduced by the addition operation, which is not observable in the single-turn setting. Plus, we only allow position reference to point to the original position from scene initialization rather than intermediate positions during editing. This design ensures position references have clear, unambiguous targets.

Meanwhile, we do not explicitly exclude trajectories in which a later editing operation overwrites an earlier one, as such interactions naturally arise in complex editing tasks and contribute to instruction complexity from different angles. Moreover, correctly resolving these dependencies is itself a non-trivial challenge for achieving high-fidelity edits. We also avoid any explicit prompt rewriting that could dilute or simplify the constructed instruction complexity in our experiments. This choice allows us to directly stress-test existing models' editing capabilities using the full, unaltered complex instructions.

Table A2. **Classifier-Free Guidance Implementation Comparisons**

	CFG_IMG	CFG_TEXT	ST_INPUT	CTX_INPUT	Guidance Computation
INPUT: $\langle I_0, \hat{T}_c \rangle$					
Single-Turn Editing	$\hat{T}_c$	I_0			$v_{\text{text}} = v_\theta(\text{cfg\_text}) + \gamma_{text}\big(v_\theta(\text{input}) - v_\theta(\text{cfg\_text})\big)$ $v = v_\theta(\text{cfg\_img}) + \gamma_{img}\big(v_{\text{text}} - v_\theta(\text{cfg\_img})\big)$
INPUT: $\langle I_0, T_1, I_1, \ldots, T_i \rangle$					
Default	$\{T_j\}_{j=1}^{i}$	$I_0, \{(T_j, I_j)\}_{j=1}^{i-1}$			$v_{\text{text}} = v_\theta(\text{cfg\_text}) + \gamma_{text}\big(v_\theta(\text{input}) - v_\theta(\text{cfg\_text})\big)$ $v = v_\theta(\text{cfg\_img}) + \gamma_{img}\big(v_{\text{text}} - v_\theta(\text{cfg\_img})\big)$
Separate	$\{T_j\}_{j=1}^{i}$	$\{I_j\}_{j=0}^{i-1}$			$v_{\text{text}} = v_\theta(\text{cfg\_text}) + \gamma_{text}\big(v_\theta(\text{input}) - v_\theta(\text{cfg\_text})\big)$ $v = v_\theta(\text{cfg\_img}) + \gamma_{img}\big(v_{\text{text}} - v_\theta(\text{cfg\_img})\big)$
Full-Context	$I_0, \{(T_j, I_j)\}_{k=1}^{i-2}, \{T_k\}_{k=i-1}^{i}$	$I_0, \{(T_j, I_j)\}_{k=1}^{i-1}$			$v_{\text{text}} = v_\theta(\text{cfg\_text}) + \gamma_{text}\big(v_\theta(\text{input}) - v_\theta(\text{cfg\_text})\big)$ $v = v_\theta(\text{cfg\_img}) + \gamma_{img}\big(v_{\text{text}} - v_\theta(\text{cfg\_img})\big)$
Context-Guided	$\{T_j\}_{j=1}^{i}$	I_0	$I_0, \{T_j\}_{j=1}^{i}$	$I_0, \{T_j\}_{j=1}^{i}$, I_{i-1} or other context composition	$v_{\text{text}} = v_\theta(\text{cfg\_text}) + \gamma_{text}\big(v_\theta(\text{st\_input}) - v_\theta(\text{cfg\_text})\big)$ $v_{\text{d}} = v_\theta(\text{cfg\_img}) + \gamma_{img}\big(v_{\text{text}} - v_\theta(\text{cfg\_img})\big)$ $v = v_{\text{d}} + \gamma_{ctx}\big(v_\theta(\text{ctx\_input}) - v_\theta(\text{st\_input})\big)$

A.5. Dataset Analysis

We construct a synthetic dataset including 60K independent chains and 34K dependent chains through our pipeline. Samples of constructed editing sequences are illustrated in Figure A1. In Figure A2, we provide a statistical analysis over the dataset from the following four aspects: (1) Editing Chain Length: The constructed editing chains have an average length of 9.98, skewed slightly towards longer sequences; (2) Dependency Ratio: Among all the editing chains with inter-step dependencies, we have dependency ratios concentrated in the range of (0, 0.55], with an average ratio of 0.29; (3) Unique Object Count: Each scene have 9.19 unique objects on average, which are drawn from 2,000+ 3D assets, offering sufficient scene complexity for editing chain construction; (4) Distribution of Editing Operations: 10 different operations are distributed in balance, mildly favoring object translation and replacement.

B. Experimental Details

Baselines. In synthetic experiments, we compare our approach against four state-of-the-art editing models as baselines: Qwen-Image [37], GPT-4o [22], Gemini 2.5 Flash Image [33] and BAGEL (Zero-Shot) [10]. All four baselines are evaluated in a zero-shot manner. To evaluate their sequential editing performance, we implement Gemini 2.5 Flash Image and BAGEL (Zero-Shot) as in-context editing, where all subsequent edits attend to all previous contextual inputs, and GPT-4o as canonical multi-turn editing, in which every edit is performed on the result from the last editing session without access to the historical context.

Implementation details. Our framework is built on top of BAGEL [10], which naively supports the training of interleaved generation. Specifically, we adapted the implementation from [19] for instruction decomposition and in-context editing training tasks. We set max_num_tokens to 55000 to cover more complex tasks with longer interleaved sequences, and increased the dropout rate for different tokens to {text: 0.15, VAE: 0.4, ViT: 0.4}. For other hyperparameters, we reuse the configurations provided by the authors and finetune the model for 12,000 steps in synthetic experiments and 1,000 steps in sim-to-real experiments, with a learning rate of 2e-5. All experiments are performed on 8 H100 GPUs. In Complex-Edit evaluation, we employ direct concatenation, rather than using GPT-4o, to compound 8 atomic instructions in each task.

Table A3. Evaluations on different sequential editing paradigms

Sequential Editing	Independent	Dependent
Default (K=3)	4.06	4.07
Default (K=5)	**4.18**	**4.13**
Separate (K=3)	4.11	4.12
Separate (K=5)	**4.19**	4.05
Full-Context (K=3)	4.14	4.06
Full-Context (K=5)	4.16	**4.13**
Context-Guided (K=3)	4.15	4.04
Context-Guided (K=5)	**4.20**	**4.14**

Metrics. For DINOv3-based similarity metrics, we compute (1) DINO-I: the cosine similarity between the DINO features of the generated image and its reference target image as $cos(\boldsymbol{C}(I_{gen}), \boldsymbol{C}(I_{tgt}))$; (2) DINO-D: the cosine similarity over editing directions as $cos(\boldsymbol{C}(I_{gen}) - \boldsymbol{C}(I_{src}), \boldsymbol{C}(I_{tgt}) - \boldsymbol{C}(I_{src}))$, where $\boldsymbol{C}$ denotes DINOv3 encoder, and $I_{src}, I_{tgt}, I_{gen}$ denote the source image, the target image and the generated image respectively. For GPT-5–based evaluation, we adopt a two-step procedure: (a) GPT-5 first infers the desired modifications by jointly analyzing the source image and the complex instruction, (b) we then provide GPT-5 with the generated editing result to assess how well it satisfies those inferred requirements. The prompts we applied are shown in Table A1.

C. More Experiments and Ablation Studies

Classifier-Free Guidance. In our framework, different ways of computing classifier-free guidance encompass versatile editing paradigms. We instantiate our framework

with BAGEL [10], in which the authors provide one way to compute classifier-free guidance that is slightly different from what we used in full-context sequential editing and context-guided sequential editing. In the following, we term the editing paradigm induced by the default CFG computation from BAGEL as "default sequential editing". Specifically, the sequential editing performances of BAGEL (Zero-Shot) in Table 1 and Table 2, BAGEL-Zero-Shot (2-Step Sequential Editing), as well as BAGEL-R (2-Step Sequential Editing) in Table 5, are evaluated with default sequential editing paradigm. We provide a detailed comparison between default, full-context, context-guided paradigms, along with another editing paradigm which separately compute the guidance from text-only and image-only sequences (termed "separate") in Table A2, following the notations from BAGEL's implementation. The default, separate and full-context sequential editing paradigms utilize all contextual inputs to generate the next editing result with different context focuses, whereas context-guided editing selectively use previous editing results as an additional signal to guide single-turn editing. Furthermore, we report the evaluation results on both independent and dependent chains for all four sequential editing paradigms with our finetuned BAGEL in Table A3. We discover that all sequential editing paradigms achieves better performance than single-turn editing after finetuning, highlighting the effectiveness of sequential decomposition and robustness of our framework. Also, the default and full-context editing paradigms generally have similar performances on the challenging dependent setting, where the separate editing paradigm encounters a decline when the number of decomposition steps increases. Context-guided editing overall can be considered as a robust paradigm for sequential decomposition.

Effect of γ_{ctx}. By default, we use $\gamma_{ctx} = 2.5$ for synthetic editing tasks and $\gamma_{ctx} = 0.5$ for real-world editing tasks. We also test context-guided sequential editing ($K = 3$) with different values of γ_{ctx} on 100 unseen independent and dependent chains, and provide the results in Table A4. We found $\gamma_{ctx} = 2.5$ achieves the best overall performance among the evaluated settings.

Table A4. **Ablation on γ_{ctx} for Context-Guided Editing (K=3)**

γ_{ctx} ($K = 3$)	0.5	1.5	2.5	3.5
Independent Chains (100)	**4.41**	4.24	**4.35**	4.30
Dependent Chains (100)	3.96	4.10	**4.23**	3.87

Effect of context composition. By default, we use the last editing result alone as the guidance for guidance computation in context-guided editing. We also compare different context compositions on 100 independent and dependent chains, and provide the results in Table A5. We observe that using all previous editing results can lead to a performance drop. We hypothesize that the errors in all previous editing results can cumulatively impact the current edits, leading to compounding errors and performance degradation.

Table A5. **Ablation on context composition for Step i**

Context Composition (K=3, $\gamma_{ctx} = 2.5$)	All Previous Editing Results $\{I_j\}_{j=0}^{i}$	Last Editing Result I_{j-1}
Independent Chains (100)	4.18	**4.35**
Dependent Chains (100)	4.13	**4.23**

Ablation on decomposition steps. We study how the number of decomposition steps impacts the performance of context-guided sequential editing and provide the results in Table A6. We observe that more decomposition steps yield performance gains. Specifically, $K = 4$ performs the best on independent chains, while $K = 5$ achieves the highest performance on editing tasks with inter-step dependencies.

Table A6. **Ablation on decomposition steps (K) for Context-Guided Sequential Editing with $\gamma_{ctx} = 2.5$**

K	Single-Turn (1)	2	3	4	5
Independent Chains	4.0	4.06	4.15	**4.27**	4.20
Dependent Chains	3.91	4.11	4.04	4.11	**4.14**

Editing sequences versus pairwise transitions as training data. We also investigate how the training data formulation affects the sequential editing performance. To enable sequential editing, we can either train our model on editing sequences with objectives described in 4, or break each editing sequence down into individual editing pairs and optimize the model to perform single-turn editing on these pairwise transition data. We train BAGEL on these two types of data with their corresponding objectives respectively, and compare their sequential editing performance on independent chains with three decomposition steps. Table A7 shows that even though the model is trained on data that teaches the model how to perform the visual transitions individually, it remains difficult to perform sequential editing robustly due to the train-test gap. On the other hand, training on editing sequences naturally encourages the model to perform sequential editing by utilizing the rich contextual inputs, leading to better performance.

Table A7. **Comparison between sequential and pairwise training data for sequential editing inference**

# Editing Operations in Tasks	3	>3
Pairwise Transitions	2.94	2.30
Editing Sequences	5.03	4.04

Ablation on single-turn editing training objective. We investigate whether incorporating a single-turn editing objective during training affects the performance of context-guided editing. In Table A8, we compare context-guided

editing results using three decomposition steps (*i.e.* $K = 3$) on independent chains, with and without the single-turn editing objective enabled. We find that the instruction decomposition and in-context editing objectives alone already yield robust improvements through context-guided sequential editing, while adding the single-turn editing objective provides a further performance boost.

Table A8. **Impact of single-turn editing training objective on context-guided sequential editing**

Training Objective	w/o Single-Turn Editing	w/ Single-Turn Editing
Context-Guided Editing	4.12	4.15

Additional Baselines on Complex-Edit. In Table A9, we re-run AnyEdit [41] and UltraEdit [43] with instructions composed by concatenating 8 atomic editing operations.

Table A9. **Additional baselines on Complex-Edit**

Complex-Edit	IF ↑	IP ↑	PQ ↑
AnyEdit (Single-Turn)	1.68	8.49	7.37
AnyEdit (2-Step Sequential Editing)	2.61	7.43	6.63
UltraEdit (Single-Turn)	6.00	5.49	7.19
UltraEdit (2-Step Sequential Editing)	5.78	5.01	7.38

Inference Cost. In Table A10, we analyze inference cost on a single A100 GPU with K=$\{1,3,5\}$ and 50 denoising steps per image. We find that fully generating K>1 images with CGSE takes approximately $0.85K$ times the cost of single-turn editing (K=1)—a 15% speedup enabled by KV cache. However, this cost can be further reduced on more advanced GPUs (*e.g.* H200) and with faster sampling after distillation.

Table A10. **Inference cost analysis**

Decomposed Steps (K)	1	3	5
Inference Time Per Task (s)	43.2 ± 1.5	112.1 ± 1.6	182.0 ± 1.4

D. More Related Work

Instruction-based Image Editing. As opposed to prior works [4, 9, 16, 18, 20, 35] that rely on detailed textual descriptions of the desired image to guide the editing process, "instructions" specify how the image should be changed, providing a user-friendly interface for image editing. Fueled by advancements in model architectures and multimodal datasets for image synthesis, instruction-based image editing has achieved rapid progress over recent years. InstructPix2Pix [3] pioneered the use of instructions for image editing by finetuning Stable Diffusion [27] on synthetic data [16], and inspired a line of following works in which editing models are further improved to perform diversified types of edits on both real and synthetic images [5, 14, 17, 30, 42, 43].

E. More Qualitative Results

- We provide a full five-step illustration of the context-guided sequential editing on a dependent chain with 13 editing operations in Figure A3. Context-guided sequential editing modifies the scene following the per-step instruction, leading to high-accuracy edits with merely one incomplete modification (*i.e.*, rotated the tissue box for 180° but did not continue the rotation for 90° counter-clockwise), which is labeled by the yellow box.
- We compare context-guided sequential editing with full-context sequential editing on an independent chain (Figure A4) and a dependent chain (Figure A5) when decomposing the complex instruction into 5 steps. We observe that context-guided sequential editing robustly produces correct edits, whereas full-context sequential editing often leads to undesired modifications.
- In Figure A6 and Figure A7, we compare our methods against baselines on two independent chains with 6 and 5 editing operations, respectively. Particularly, we observe that in Figure A6, even though context-guided sequential editing fails to replace the trivet with a cocoa bean in the first edit, it can still correct its behavior in the subsequent edits and achieves a perfect edit after three steps. Overall, context-guided sequential editing effectively improves over single-turn editing and outperforms other sequential editing baselines.
- In Figure A8, we compare our methods against baselines on two dependent chains with 10 editing operations. Context-guided sequential editing produces a high-quality result with only one incomplete edit (*i.e.*, replaced the vase with a shoe but did not continue to update its material to fabric), whereas other proprietary baselines generate results with more incorrect edits.
- We provide additional qualitative results (Figures A9-A23) on Complex-Edit [40] for sim-to-real generalization. In most of the real-world editing examples, we find that two-step context-guided sequential editing consistently demonstrates better instruction-following and identity preservation capabilities compared to other baselines. However, we also observe that sequential editing can sometimes fail to execute fine-grained edits (*e.g.*, A22) or overly emphasize certain effects (*e.g.*, Figures A21, A23), resulting in compromised fidelity.

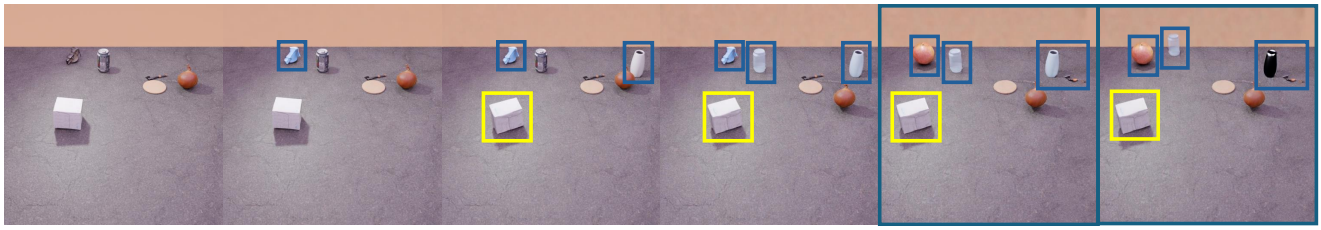


Figure A3. Full illustration of 5-step decomposition for Context-Guided Editing. Decomposed Instruction over 13 editing operations: (1) Fully cover the bench vice with metal material, fully cover the bench vice with plastic material. (2) Add a vase on the floor near the yellow onion, rotate the tissue box 180 degrees, rotate the tissue box 90 degrees counter-clockwise. (3) Move the yellow onion forward, fully cover the can with glass material, fully cover the vase with glass material. (4) Replace the bench vice with a grapefruit, move the hatchet to the right, pan the camera right. (5) Fully cover the vase with metal material, move the can backward. Numbered list denotes the decomposition steps. Blue boxes indicate the correct edits, and yellow boxes indicate edits that are not fully executed.

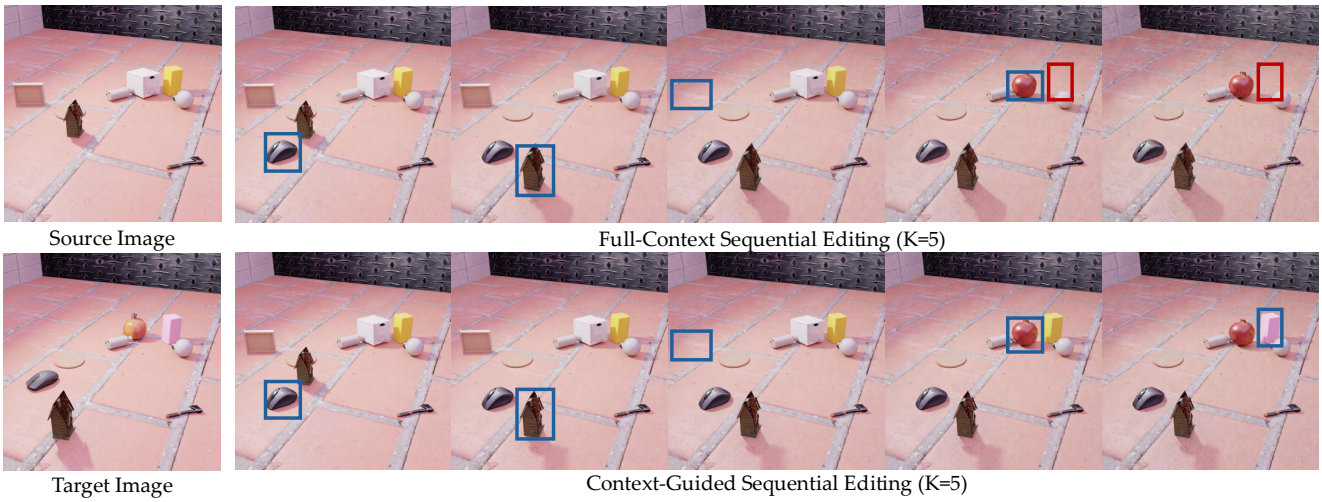


Figure A4. Full illustration of 5-step decomposition for tasks with 5 editing operations. Instruction: (1) Add a mouse on the floor near the trivet, (2) move the bird house forward, (3) remove the frame, (4) replace the tissue box with a pomegranate, and (5) paint the cuboid wood block pink. Blue boxes indicate correct edits, whereas red boxes indicate undesired ones. Numbered list denotes the decomposition steps.

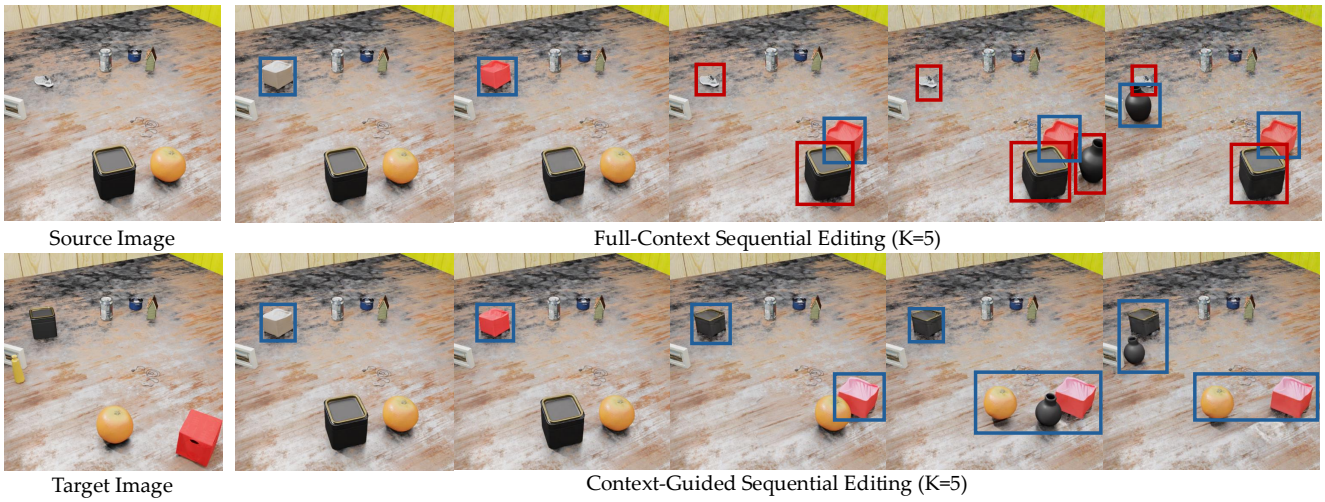


Figure A5. Full illustration of 5-step decomposition for tasks with 7 editing operations and inter-step dependencies. Instruction: (1) Replace the shoe with a tissue box, (2) paint the object that was recently placed in exchange for the shoe to red, (3) move the object that was recolored near the tangerine, move the coffee canister where the shoe initially was, (4) relocate the tangerine back to where the coffee canister initially was, add a vase where the tangerine initially was, and (5) relocate the object that was added near the frame. Blue boxes indicate correct edits, whereas red boxes indicate undesired ones. Numbered list denotes the decomposition steps.

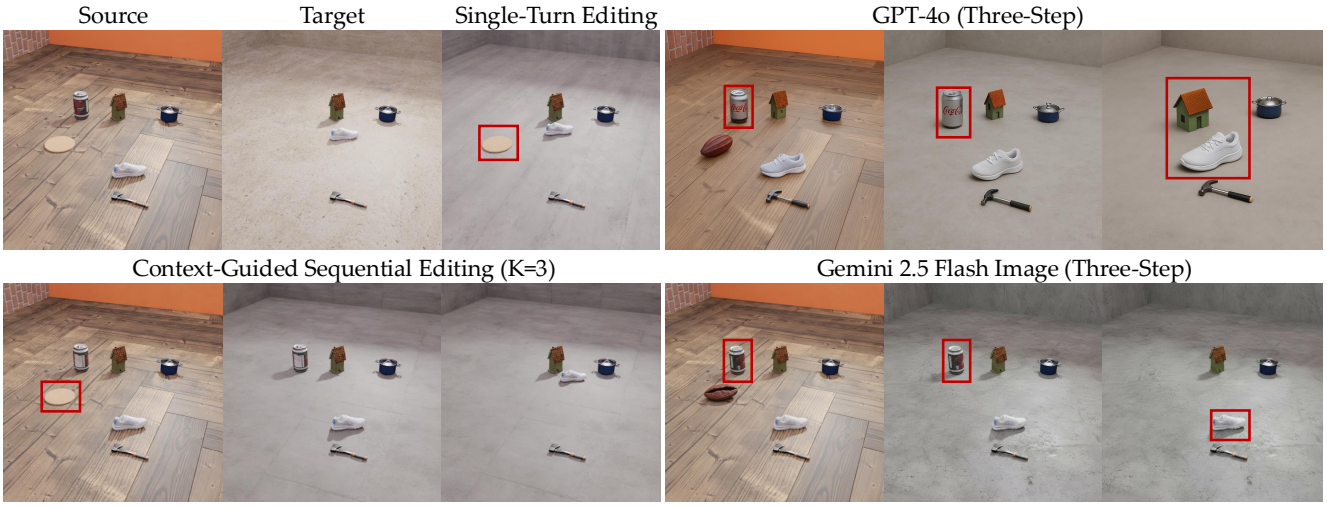


Figure A6. Comparison for tasks with 6 editing operations. Instruction: (1) Replace the trivet with a cocoa bean, rotate the can 90 degrees counter-clockwise. (2) Remove the cocoa bean, change the floor and walls to concrete. (3) Remove the can, move the shoe near the bird house. "Single-Turn Editing" refers to the single-turn editing result from finetuned BAGEL. Red boxes indicate wrong edits. Numbered list denotes the decomposition steps.

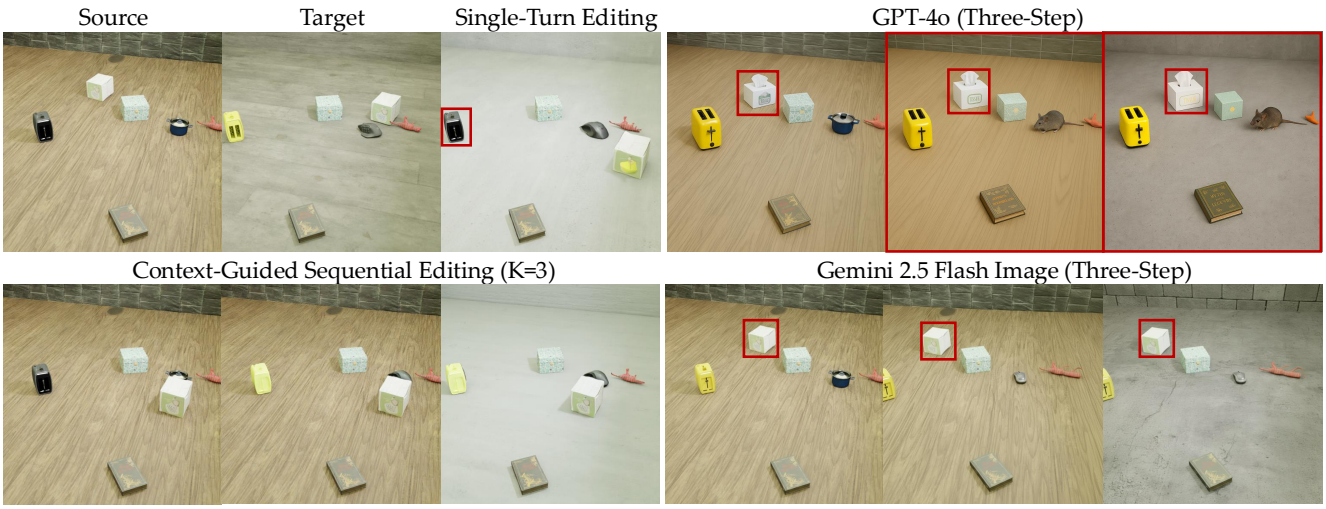


Figure A7. Comparison for tasks with 5 editing operations. Instruction: Move the tissue box near the pot, paint the toaster yellow, replace the pot with a mouse, pan the camera right, and change the floor and walls to concrete. "Single-Turn Editing" refers to the single-turn editing result from finetuned BAGEL. Red boxes indicate wrong edits.

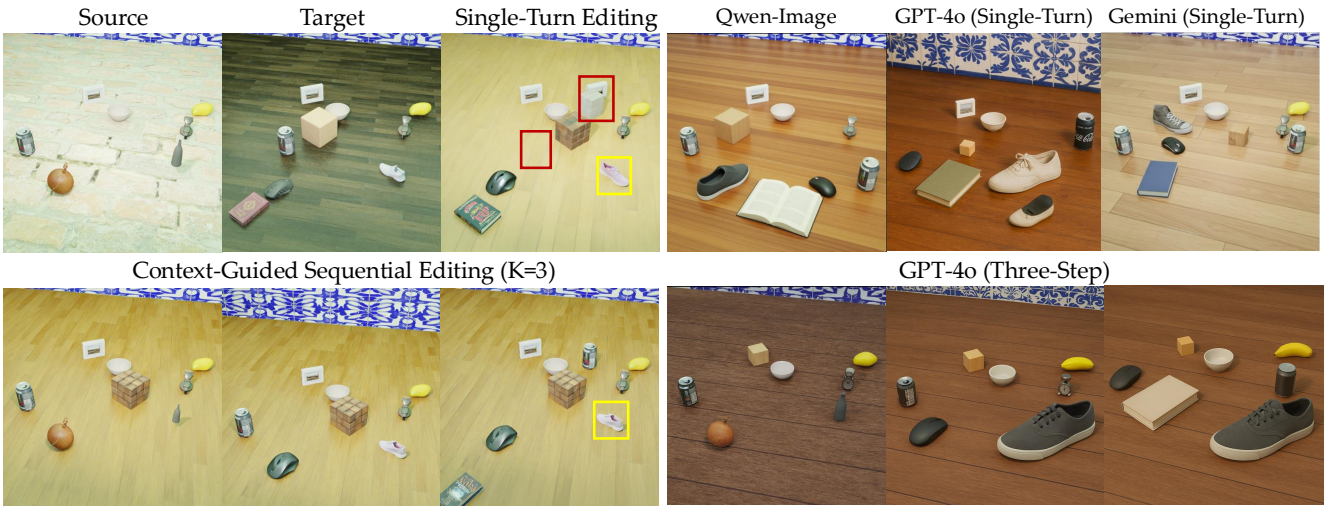


Figure A8. Comparison for tasks with 10 editing operations and inter-step dependencies. Instruction: Make the vase smaller, add a cube wood block on the floor near the mixing bowl, change the floor to wood floor, replace the vase with a shoe, tilt the camera up, change the material of the object that was placed in exchange for the vase to fabric material, replace the yellow onion with a mouse, add a book on the floor near the mouse, tilt the camera down, and move the can to the right. "Single-Turn Editing" refers to the single-turn editing result from finetuned BAGEL. On the left, red boxes indicate wrong edits, whereas yellow boxes indicate edits that are not fully executed. Results from the proprietary baselines generally all fail to edit the image accurately.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

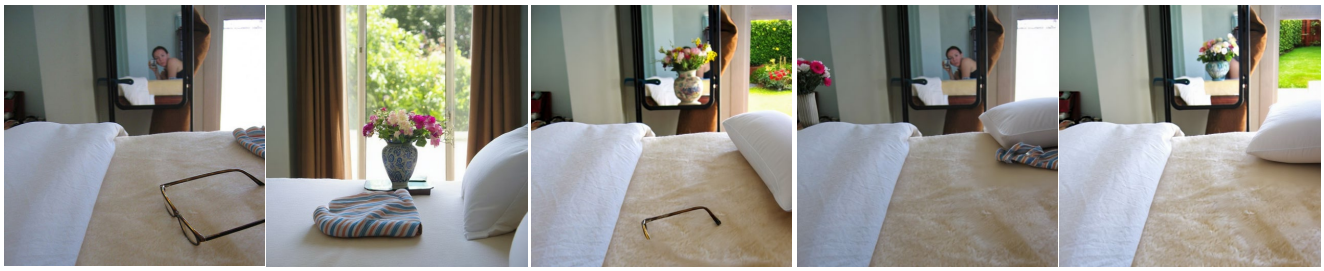
Figure A9. Comparison for real-world editing with instruction: Change the color of the car from red to metallic blue, remove the surfboard on the car, add a luggage rack with vintage suitcases on top of the car, change the small american flag to a small french flag mounted on the same spot, replace the apartment building background with a european cobblestone street, apply a warmer lighting effect, add a soft motion blur to the surrounding environment, and crop the image to emphasize the car.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A10. Comparison for real-world editing with instruction: Remove the laptop from the setup, insert a bouquet of fresh flowers in place of the laptop, replace the train seat with a wooden table and an outdoor garden background, relocate the plate with grilled meat to the center, duplicate the bottle of wine and place the copy on the opposite side of the table, add warm sunlight to simulate a late afternoon ambiance, refocus composition on the plates and enhance surrounding elements, and apply a soft sepia filter.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A11. Comparison for real-world editing with instruction: Replace the view through the door window with a scenic garden view, place a modern laptop on the table in front of the foreground dining chair, remove the small decorative plate on the edge of the dining table, add an afternoon sunlight effect through the curtains on the right, transform the wooden tabletop to a granite texture, turn the red stripes on the dining chairs' upholstery to blue, insert a note pad with scribbled letters next to the laptop, and zoom out slightly to provide a wider view of the room.



Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A12. Comparison for real-world editing with instruction: Place a decorative vase with flowers on the table in the background, remove the glasses from the bed, reposition the striped fabric so it is neatly folded next to the location of the removed glasses, insert an extra pillow onto the bed right side, brighten the lighting slightly, add a soft glow effect, replace the visible outside portion in the mirror to show a garden view instead, and center the image frame on the bed and balance added elements.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A13. Comparison for real-world editing with instruction: Add a motion blur effect to the skateboard, replace the background with a mountainous environment, change the rider's helmet color to bright red, add a trail of dust particles behind the skateboard, add another rider, similarly equipped, in the image, slightly behind the main subject, adjust the duplicated rider's pose to a crouching position, apply soft lighting from the upper-left corner, and apply a warm golden evening filter.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A14. Comparison for real-world editing with instruction: Replace the wallpaper with a modern solid gray tile pattern, change the current wooden cabinet to a sleek, glass-door shelf, change the shower curtain to a turquoise color, add a decorative plant in a white pot on the countertop, replace the bathroom sink tap with a modern chrome tap, increase the lighting brightness, place a small anti-slip rug on the floor in front of the sink, and apply a photo filter to enhance details and colors.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

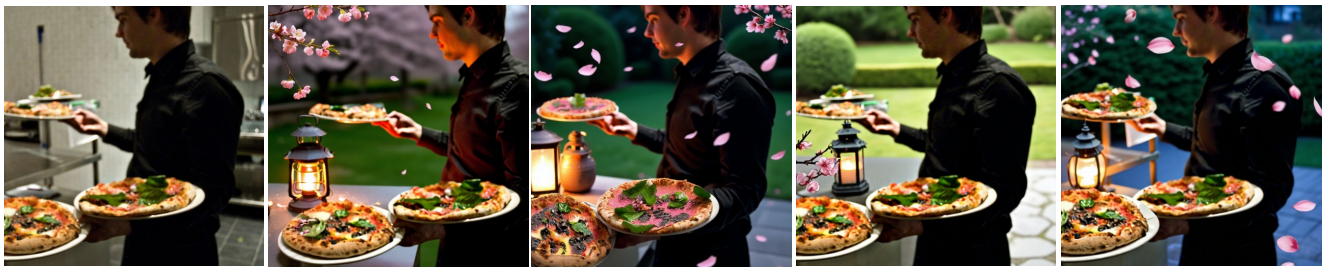
Figure A15. Comparison for real-world editing with instruction: Replace the background with an aesthetic library study environment, change the screen content to a natural scenery image, place a coffee cup near the keyboard, add a warm ambient light effect, reposition the cat to be seated beside the display, duplicate the cat and reposition it in the composition, modify the cats' fur to include patterned orange highlights, and crop the image to focus on the computer setup and the cat.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A16. Comparison for real-world editing with instruction: Change the color of the refrigerator to pastel pink, add sparkles to the refrigerator's surface, place a vase with fresh flowers on top of the refrigerator, modify the wall behind the refrigerator to a floral-patterned wallpaper, remove the appliance to the right of the refrigerator, make the refrigerator appear taller by about 20% in height, apply a warm color tone filter, and add the word 'vintage chic' in a cursive font on the refrigerator door.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A17. Comparison for real-world editing with instruction: Insert a colorful beach umbrella to the left side of the image, place a seagull perched on the foremost bench, enhance lighting to resemble a sunny day, replace the overcast background with a clear sky, add subtle sparkling effects to the water, add a sandy pathway leading to the benches, scatter light sand particles near the foreground of the image, and adjust the angle so the benches align symmetrically in the middle third of the view.

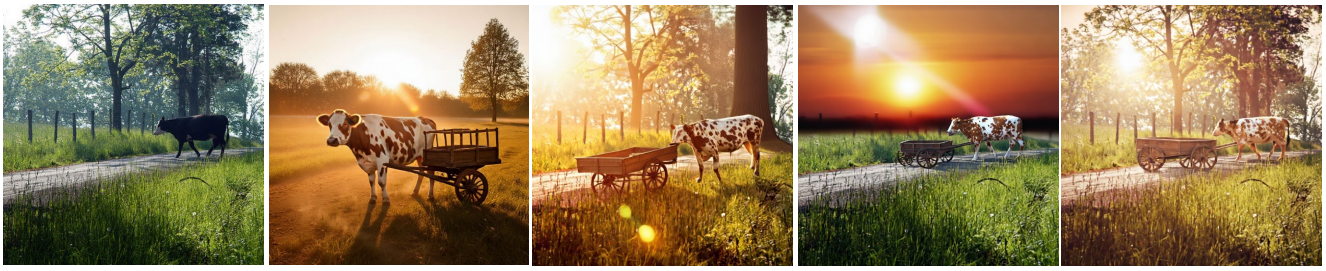


Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A18. Comparison for real-world editing with instruction: Place a small cherry blossom branch next to the dishes on the counter, replace the kitchen background with an outdoor garden setting, enhance the vibrancy of the pizza's topping colors, introduce a candlelit lantern next to the person carrying pizzas, change the lighting to mimic a late evening ambiance, add a slight glow effect, resize the pizzas to make them slightly larger, and add softly falling petals.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A19. Comparison for real-world editing with instruction: Replace the surfer with a dolphin breaching out of the water, change the background houses and coastline to a tropical beach setting, add a setting sun low on the horizon in the background, apply a golden hour lighting filter over the image, add a flock of seagulls to the sky at a distance, change the water color to sparkling turquoise hues, add water sprays around the dolphin, and adjust shadows and highlights.



Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A20. Comparison for real-world editing with instruction: Replace the background with a sunset scene, add a lens flare effect near the sunset, change the color of the cow to spotted brown and white, add a vintage wooden cart being pulled by the cow, add soft shadows and warm lighting around the cow and cart, add small dust particles to the background, make the tree on the right taller, and apply a golden-hue filter.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A21. Comparison for real-world editing with instruction: Insert a large tree to the front left of the image, remove the bus on the left side of the image, replace the white car in the foreground with a vintage electric sedan, switch the background to depict a sunset scene, modify the color of the house to light blue, introduce floating leaves around the tree, dim the overall lighting, and add a light glow effect near the sun.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A22. Comparison for real-world editing with instruction: Change the bed linens color from blue to neutral gray, place an open book on the side of the bed, add a houseplant in a ceramic pot on top of the cabinet, increase the brightness of the light from the wall sconce, alter the bed's headboard fabric to textured velvet, shift the side lamp to the left of the bed, apply a warmer tone filter to the image, and introduce sunlight rays coming through the window.

Source Image | BAGEL-R (Single-Turn) | BAGEL-SR (Single-Turn) | BAGEL-SR (Context-Guided, K=2)

Figure A23. Comparison for real-world editing with instruction: Replace the dog with a fox, replace the grass background with a snowy forest scene, change the sheep's texture to resemble a rocky shape, modify the lighting to simulate a cold, wintry evening, introduce a snowfall effect, add a faint magical glow around the fox, alter the fox's color to a striking silver shade, and reframe the composition to focus on the fox and the modified object.