Appendix

The Appendix provides additional technical details, derivations, implementation procedures, and experimental results that complement the main paper. It is organized as follows:

- Appendix A presents the full derivation of the latent anchor-based optimization framework introduced in the main paper, including the strong-form global objective, its closed-form solution, the reduced objective, and the relaxed per-timestep formulation adopted in practice.
- Appendix B provides additional implementation details for our framework, including dataset construction, training-free 3D editing, scalable dataset curation, and the experimental pipeline.
- Appendix C summarizes supplementary experimental settings, covering evaluation metrics, baseline implementations, and further comparisons.
- Appendix D reports additional qualitative and quantitative results that further validate the effectiveness and robustness of the proposed method.

A. Derivation of the Latent Anchor-based Optimization

In this section, we detail the derivation of the latent anchor-based optimization introduced in Sec. 3.2.

Definition of Latent Anchor-based Optimization. As defined in Sec. 3.2, we assume an ideal latent anchor $\boldsymbol{A}$. Let

$$\mathbf{s}_t \triangleq F_t(\boldsymbol{X}_t^{\mathrm{src}}, t, c_{\mathrm{src}}), \qquad \mathbf{g}_t \triangleq F_t(\boldsymbol{X}_t^{\mathrm{tar}}, t, c_{\mathrm{tar}}),$$

where $F_t(\cdot)$ maps a latent state at time t to a reference latent (*e.g.*, via a single-step inversion approximation). The strong form of the global anchor objective is

$$\mathcal{J}(\boldsymbol{A}) = \sum_{t=1}^{T} \Big(\|\mathbf{s}_t - \boldsymbol{A}\|_2^2 + \|\mathbf{g}_t - \boldsymbol{A}\|_2^2 \Big), \tag{12}$$

which enforces a single latent anchor $\boldsymbol{A}$ to explain both the source and target reconstructions across all timesteps.

Solving for the optimal latent anchor $\boldsymbol{A}^*$. The objective in (12) is quadratic and convex in $\boldsymbol{A}$. Setting the gradient to zero gives

$$\nabla_{\boldsymbol{A}} \mathcal{J}(\boldsymbol{A}) = 2 \sum_{t=1}^{T} \big[(\boldsymbol{A} - \mathbf{s}_t) + (\boldsymbol{A} - \mathbf{g}_t) \big] = 4T\,\boldsymbol{A} - 2 \sum_{t=1}^{T} (\mathbf{s}_t + \mathbf{g}_t) = \mathbf{0}.$$

Hence,

$$\boldsymbol{A}^* = \frac{1}{2T} \sum_{t=1}^{T} (\mathbf{s}_t + \mathbf{g}_t) \tag{13}$$

That is, the optimal latent anchor is the mean of all per-timestep midpoints $\mathbf{m}_t \triangleq \frac{1}{2}(\mathbf{s}_t + \mathbf{g}_t)$.

Substituting $\boldsymbol{A}^*$ Back. To obtain the reduced objective $\mathcal{J}(\boldsymbol{A}^*)$, we use the parallelogram identity:

$$\|\mathbf{x} - \mathbf{m}\|^2 + \|\mathbf{y} - \mathbf{m}\|^2 = \tfrac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2 + 2\big\|\mathbf{m} - \tfrac{\mathbf{x}+\mathbf{y}}{2}\big\|^2. \tag{14}$$

Applying (14) with $\mathbf{x}{=}\mathbf{s}_t$, $\mathbf{y}{=}\mathbf{g}_t$, and $\mathbf{m}{=}\boldsymbol{A}$ yields

$$\sum_{t=1}^{T} \big(\|\mathbf{s}_t - \boldsymbol{A}\|^2 + \|\mathbf{g}_t - \boldsymbol{A}\|^2 \big) = \frac{1}{2} \sum_{t=1}^{T} \|\mathbf{g}_t - \mathbf{s}_t\|^2 + 2 \sum_{t=1}^{T} \Big\| \boldsymbol{A} - \frac{\mathbf{s}_t + \mathbf{g}_t}{2} \Big\|^2.$$

Minimizing the right-hand side over $\boldsymbol{A}$ gives $\boldsymbol{A}^* = \bar{\mathbf{m}} \triangleq \frac{1}{T}\sum_t \mathbf{m}_t$, and the exact reduced objective is

$$\mathcal{J}(\boldsymbol{A}^*) = \frac{1}{2} \sum_{t=1}^{T} \|\mathbf{g}_t - \mathbf{s}_t\|^2 + 2 \sum_{t=1}^{T} \|\mathbf{m}_t - \bar{\mathbf{m}}\|^2, \qquad \mathbf{m}_t = \tfrac{1}{2}(\mathbf{s}_t + \mathbf{g}_t),\ \bar{\mathbf{m}} = \tfrac{1}{T} \sum_t \mathbf{m}_t. \tag{15}$$

The first term measures pairwise alignment between source and target reconstructions, while the second term enforces the midpoints to remain consistent across timesteps, corresponding to a single global anchor.

Relaxed Latent Anchor-based Optimization. For computational efficiency, we adopt a relaxed formulation. Since the second term in Eq. (15) is nonnegative, the objective satisfies

$$\mathcal{J}(\boldsymbol{A}^*) \geq \frac{1}{2}\sum_{t=1}^{T}\|\mathbf{g}_t - \mathbf{s}_t\|^2, \tag{16}$$

with equality if and only if $\mathbf{m}_t = \bar{\mathbf{m}}$ for all t. In practice, we adopt the simplified per-timestep form

$$\mathcal{L}_{\text{align}} = \tfrac{1}{2}\sum_{t=1}^{T}\|\mathbf{g}_t - \mathbf{s}_t\|^2 = \tfrac{1}{2}\sum_{t=1}^{T}\|F_t(\boldsymbol{X}_t^{\text{tar}}) - F_t(\boldsymbol{X}_t^{\text{src}})\|^2, \tag{17}$$

which corresponds to the natural relaxation (*i.e.*, a lower bound) of the strong-form objective (15), obtained by omitting the temporal midpoint-consistency term.

B. Additional Implementation Details

B.1. Details of Eval3DEdit

Eval3DEdit is a benchmark dataset specifically designed to evaluate 3D editing performance. It comprises 100 samples tailored for 3D editing tasks, categorized into five distinct types: action change, object addition, object removal, object replacement, and style change, with 20 samples allocated to each category. Each sample consists of an editing instruction, a source 3D shape, a source image, and a target image. The dataset is constructed through a multi-stage pipeline that collects high-quality 3D shapes, generates editing instructions, selects optimal source viewpoints, produces target images, and finally curates representative samples for comprehensive 3D editing evaluation.

- **Step 1: Source 3D Shape Collection.** We collect source models and their corresponding captions from Objaverse-XL [12]. To ensure data quality, we filtered for high-quality shapes using an aesthetics score threshold of 7.0.
- **Step 2: Editing Instruction Construction.** Given the caption associated with each 3D shape, we employed Gemini 2.5 Pro [9] to assign an appropriate editing category and to generate a corresponding editing instruction. The prompt used for this process is shown below.

Step2: Editing Instruction Construction

[Task Definition]: Generation of a Balanced Editing Instruction Dataset
[Objective]:
Given an input CSV file containing assets (*e.g.*, images or videos) and their corresponding textual captions, generate a dataset of editing instructions. This task involves assigning each asset to one of five predefined editing categories and subsequently formulating a precise instruction based on the caption and the assigned category.
[Editing Categories]:
The five (5) mandatory editing categories are:
Object Addition
Object Removal
Object Replacement
Style Change
Action Change
[Procedure and Requirements]:
Data Processing: For each asset in the input CSV, analyze its original caption.
Category Assignment: Assign one of the five predefined editing categories that is semantically appropriate based on the content described in the caption.
Instruction Generation: Based on the original caption and the assigned category, generate a specific, textual "Editing Instruction" detailing the desired modification.
Balanced Distribution: The final output dataset must maintain an approximately equal (balanced) distribution of samples across all five editing categories.
Constraint 1 (Contextual Relevance): All generated instructions (e.g., specifying additions, removals, replacements, or action modifications) must be contextually relevant and semantically coherent with the content of the original asset

caption.
Constraint 2 (Specificity): Instructions must be unambiguous and explicit. They must clearly define the specific outcome expected after the edit is applied. Avoid generalized or vague commands. The content to be edited should be specifically designated and clearly defined. For example, for a water cup, the content to be edited should not be a general 'accessory' but rather a specific 'cup handle'.
Constraint 3 (Content Preservation): The edit must not constitute a complete replacement of the original asset's core subject or scene, as this would be equivalent to new generation rather than modification. The core identity of the asset must be preserved.
Constraint 4 (Instructional Diversity): The set of generated instructions must exhibit high diversity. Repetitiveness should be minimized, particularly among instructions generated within the same editing category.
Constraint 5: Action change is only applicable to assets such as humans, animals, and others capable of producing actions. Careful selection of assets suitable for Action change is required when generating editing instructions.
[Output Format]:
The results should be delivered in a structured CSV file mapping each original asset (and its caption) to its assigned Editing Category and the corresponding generated Editing Instruction. Please generate this CSV file for me.

- **Step 3: Source Image Selection.** We rendered eight viewpoints for each source model using the rendering scripts provided by TRELLIS [64]. To select the most suitable source condition for editing, we used Gemini 2.5 Flash to identify the optimal viewpoint. The prompt used for this selection is provided below.

Step3: Source Image Selection

You are an expert visual assistant. You will be given an editing category, a specific instruction, and 8 images of the same object from different angles, labeled Image 0 through Image 7.
Your task is to determine which single image is the most suitable canvas for the requested edit. The best image should provide a clear and direct view of the object or area that needs to be modified.
- Editing Category: "{category}"
- Editing Instruction: "{instruction}"
Analyze the 8 images provided and identify the best one for this edit.
Your response MUST be a single integer number from 0 to 7, corresponding to the best image. Do not add any other text or explanation.

- **Step 4: Target Image Generation.** Next, we combined the selected source condition with the editing instruction and used Nano Banana [9] to generate the target condition. The prompt used for this generation is given below.

Step 4: Target Image Generation

As an expert image editor, your task is to edit the following image.
You must strictly and precisely follow the provided category and instruction. The edit should be realistic and seamlessly integrated into the image.
- Editing Category: "{editing_category}"
- Editing Instruction: "{editing_instruction}"

- **Step 5: Final Sample Selection.** To ensure high-quality editing outcomes, we prioritized samples where the source condition represented a frontal viewpoint. Ultimately, we selected 20 samples for each editing category to construct Eval3DEdit, a benchmark dataset covering representative rigid and non-rigid edits for comprehensive 3D editing evaluation.

B.2. Details of Training-Free 3D Editing

This section describes the implementation details of our training-free 3D editing pipeline. Given a source model and an editing instruction, we first construct the source and target conditions, and then apply the proposed AnchorFlow sampling procedure to perform 3D editing.

Condition Construction. We construct $(c_{\text{src}}, c_{\text{tar}})$ following Step 3 and Step 4 of Appendix B.1. We begin by rendering multiple candidate viewpoints of the source model and use an vision language model-based selector [9] to determine the

optimal source view as c_{src}. We then pair this selected view with the editing instruction to generate the target condition c_{tar} using a text-conditioned image editor [9]. This produces a condition pair that jointly encodes the source appearance and the desired editing semantics.

AnchorFlow Sampling. We adopt Hunyuan3D 2.1 [53] as the base shape model $\boldsymbol{v}_\theta$. Hunyuan3D 2.1 is a high-fidelity 3D generative framework combining a ShapeVAE with a flow-based DiT backbone, and operates on compact vector-set latent representations suitable for 3D geometry encoding. The source 3D shape is encoded into a latent code $\boldsymbol{X}_0^{\text{src}}$. Given $\boldsymbol{X}_0^{\text{src}}$, the condition pair $(c_{\text{src}}, c_{\text{tar}})$, and the base model $\boldsymbol{v}_\theta$, we perform AnchorFlow sampling following Algorithm 1 to generate an editing trajectory in latent space. The trajectory integrates over T time steps, evolving from $n_{\max}$ to $n_{\min}$. Source and target constraints are balanced by two guidance scales, s_{src} and s_{tar}, which regulate identity preservation and editing strength, respectively. We use the default configuration $T = 50, s_{\text{src}} = 3.5, s_{\text{tar}} = 7.5, n_{\min} = 1, n_{\max} = 41$. The final latent $\boldsymbol{X}_0^{\text{FE}}$ is decoded by Hunyuan3D 2.1 into a mesh, producing an edited shape that incorporates the desired modification while preserving the source identity. On an NVIDIA H100 (96GB) GPU, each editing instance takes approximately 26.71 s.

B.3. Scalable 3D Editing Dataset Curation

Building upon the construction pipeline in Appendix B.1 and our training-free 3D editing framework in Appendix B.2, we develop a scalable and fully automated procedure for curating large-scale 3D editing datasets. Requiring no training or manual mask annotation, the pipeline efficiently produces richly annotated editing pairs.

Condition Pair Construction. Our curation pipeline begins by collecting high-quality 3D shapes and their associated captions from 3D shape datasets. Given the captions, we employ an LLM to assign an editing category and generate a corresponding editing instruction. To establish the source condition, we render multiple viewpoints for each 3D shape and use an LLM-based selector to identify the most suitable view. The selected view and the editing instruction are then fed into a text-conditioned image editor to generate the corresponding target condition, forming a semantically aligned condition pair.

Training-Free Shape Editing. We then leverage our training-free 3D editing framework to synthesize the edited 3D shape. The source model is encoded into the latent space of flow-based 3D generative model, and AnchorFlow sampling produces an edited latent trajectory that faithfully applies the specified modification while preserving shape identity. The resulting latent is decoded into a mesh, yielding the edited 3D shape paired with its source shape and editing instruction.

Dataset Curation and Applications. Our pipeline enables scalable creation of high-quality 3D editing pairs, producing data in the form of (editing instruction, source shape, edited shape). It offers an efficient approach for constructing large-scale training data for instruction-following 3D editing, thereby supporting the development of more capable and generalizable 3D editing models.

C. Supplementary Experimental Settings

C.1. Details of Measurement Metrics

In the main paper, we employ two evaluation strategies to demonstrate the superiority of the proposed method. Here we supplement the details of the measurements.

C.1.1. Evaluation with CLIP.

Based on the CLIP (EVA02-E-14-plus) model [18, 45], we introduce two CLIP-based similarity metrics CLIP_{img} and CLIP_{txt} to assess the editing performance on identity preservation and semantic modification, respectively.

- **Quantitative Evaluation of Identity Preservation.** Since the image editor [9] shows identity-preserving edits, we treat the target image as the ground truth. Based on this assumption, we quantitatively evaluate identity preservation by measuring the alignment between the edited 3D shape and the target image. Specifically, for each edited shape, we render six views at a resolution of 512×512, with a 15° elevation and a camera radius of 2.4. We then compute the CLIP image similarity between each rendered view and the target image, and use the average similarity score across all views as the identity preservation score CLIP_{img}.
- **Quantitative Evaluation of Semantic Modification.** To evaluate semantic modification, we first construct a textual description of the edited shape based on the source shape caption and the editing instruction. Specifically, we use the prompt shown below. We then compute the CLIP image-text similarity between each rendered view and the generated target caption, and use the average similarity across all views as the semantic modification score CLIP_{txt}.

Target Image Caption

You are a rewrite assistant. Given the original object captions and an editing instruction, produce ONE English sentence that describes ONLY the final, edited state:
- Describe the final state only, in the format 'subject + change'. Do NOT mention the editing process or use verbs like add/remove/replace/change.
- Be specific and objective; avoid first/second person and irrelevant speculation.
- Output exactly one English sentence. No numbering, no explanations, no quotes.
- Keep it short (<= 10 words), lower-case, no quotes, minimal punctuation.
- Examples:
instruction: 'change the robot's pose to be kneeling on one knee.' -> 'robot kneeling on one knee'
instruction: 'add a hat' -> 'figure with hat'
instruction: 'remove the sword' -> 'knight without sword'
instruction: 'replace chair with stool' -> 'scene with stool'
Editing category: {category}
Editing instruction: {instruction}

C.1.2. Evaluation with Uni3D

For further validation of identity preservation and semantic modification, we introduce two metrics based on Uni3D [74]: Uni3D_{pc} and $\text{Uni3D}_{\text{txt}}$. Unlike CLIP, Uni3D processes 3D shapes in the form of point clouds. All evaluations are conducted using the Uni3D (eva_giant_patch14_560) model.

- **Quantitative Evaluation of Identity Preservation.** We measure identity preservation using the Uni3D similarity between the source shape and the edited shape. Specifically, we extract point clouds from both shapes and assign them a fixed color (*e.g.*, gray with RGB values (100, 100, 100)). Features are then extracted using Uni3D, and identity preservation is computed as the cosine similarity between the two point-cloud feature vectors.
- **Quantitative Evaluation of Semantic Modification.** Similar to CLIP_{txt}, we evaluate semantic modification $\text{Uni3D}_{\text{txt}}$ by computing the similarity between the Uni3D point-cloud feature of the edited shape and the target caption.

C.2. Details of Baselines

In this section, we give the implementation details of the baseline methods, including TextDeformer [15], MVEdit [6], EditP23 [1], Direct Editing [53], Editing-by-Inversion [22], and Inversion-free Editing [27].

- **Implementation Details of TextDeformer.** We reproduce TextDeformer [15] following the official implementation[1]. TextDeformer deforms a source mesh into a text-specified target shape using differentiable rendering and CLIP-based semantic alignment. Instead of directly optimizing vertex displacements, it optimizes per-face Jacobians and reconstructs the deformation via a Poisson solve, enabling smooth, globally coherent geometry updates and mitigating artifacts from noisy CLIP gradients. The method also employs a view-consistency loss to enforce multi-view coherence and a Jacobian regularization term to preserve source shape identity. TextDeformer requires a pair of captions: a source caption describing the input mesh and an edited caption specifying the desired target geometry. To obtain these, we first construct the target caption following Appendix C.1. We then generate a concise source caption from the original asset description using Gemini 2.5 Pro with the prompt below. For each editing shape, we select the output from the `mesh_best_clip` directory as the final edited result. Since TextDeformer occasionally fails to optimize certain shapes (as noted in the original paper, where optimization may diverge or produce degenerate geometry), we exclude failure cases when computing the final averaged quantitative metrics.

Concise Source Caption

Write EXACTLY ONE short English noun phrase (<= 5 words) that summarizes the primary subject(s) described below. Use lower-case, no quotes, no punctuation.
- examples:
captions: 'a humanoid robot ...' -> 'humanoid robot'
captions: 'a low-poly house ...' -> 'small house'

[1] https://github.com/threedle/TextDeformer

captions: captions_text or ”,
Now output the phrase:

- **Implementation Details of MVEdit.** MVEdit [6] is a training-free 3D editing framework that adapts pretrained 2D diffusion models into 3D-consistent multi-view editors using a 3D Adapter that jointly denoises multi-view images and reconstructs coherent geometry. For each editing task, the input consists of a source shape and an editing instruction, and the output is an edited 3D shape. We reproduce MVEdit using the official Gradio-based implementation[2]. Specifically, we use the `Instruct 3D-to-3D` mode, which performs instruction-guided 3D shape editing. We fix the random seed to 42 for all experiments, and keep all other hyperparameters identical to the “polnareff” example provided in the official interface. The edited shape produced by the pipeline is used as the final result.
- **Implementation Details of Meshup.** MeshUp [26] is a mesh deformation framework that edits a source mesh toward a target concept by optimizing per-face Jacobians under diffusion-based guidance. We reproduce MeshUp following the official implementation[3]. Due to computational constraints, we run 600 optimization iterations for each sample. For each editing task, the input consists of a source mesh and a target editing concept, and we use the final deformed mesh after optimization as the edited result. MeshUp performs less effectively on our benchmark. We attribute this to its Jacobian-based reconstruction, which is better suited to well-connected meshes.
- **Implementation Details of EditP23.** EditP23 [1] is a mask-free and training-free 3D editing framework that propagates a user-provided 2D edit to a full multi-view editing. We reproduce EditP23 following the official implementation[4]. For each sample, we first render six multi-view images of the source shape using the rendering setup described in EditP23. Each editing run takes as input the source view, the edited target view, and the rendered multi-view grid. EditP23 outputs an edited multi-view grid, which we then reconstruct into a 3D mesh following the reconstruction pipeline detailed in the official implementation. Following the configuration strategy in [1], we adopt different presets for different editing types. For style-change edits, we set the target guidance scale to 5.0 and use $n_{\max} = 31$. For all other editing categories, we set the target guidance scale to 21.0 and use $n_{\max} = 39$. The resulting reconstructed mesh is used as the final edited shape.
- **Implementation Details of Direct Editing.** Given the target image, we perform inference using the flow-based 3D generative model [53] to obtain the edited shape. All experiments follow the official implementation[5].
- **Implementation Details of Editing-by-Inversion.** Editing-by-Inversion extends Direct Editing [53] by inserting a flow-based inversion prior to inference. We follow the conditional inversion procedure of UniEdit-Flow [22]. UniEdit-Flow introduces Uni-Inv, a predictor-corrector inversion method designed to more accurately recover the flow trajectory by mitigating the error accumulation inherent to straight, non-crossing rectified-flow paths. In our setting, we perform conditional Uni-Inv using the source image as the conditioning input during inversion. In specific, we set the target guidance scale to 1.0 and run 50 inversion steps to obtain an latent aligned with the source geometry, following the official implementation[6]. After inversion, we perform standard flow-based inference using the editing image to synthesize the final edited shapes.
- **Implementation Details of Inversion-free Editing.** We reproduce the inversion-free editing [27] on top of the flow-based 3D generative model [53]. It constructs a direct ODE between the source and target conditioned distributions, avoiding the Gaussian-noise traversal required in inversion-based editing. Concretely, it introduces a stochastic forward process that linearly interpolates between the source latent and random noise, and uses the resulting paired source-target latent to compute the velocity difference field, which is then averaged across multiple noise samples to update the editing trajectory. For fair comparison, we use the same hyperparameter settings as those in our method.

D. Additional Experiments

D.1. More Quantitative Results.

User Study for Semantic Modification and Identity Preservation. Based on the Eval3DEdit dataset, we design an A/B test comparing FlowEdit and our method. We ask 20 domain experts to select their preferred results in terms of semantic modification and identity preservation. **97.2%** of participants prefer our method for semantic modification and **87%** for identity preservation, providing additional support for the effectiveness of our method.

Quantitative Evaluation with Uni3D-based Metrics. Beyond the CLIP-based evaluation in main paper, we further validate

[2] https://github.com/Lakonik/MVEdit
[3] https://github.com/threedle/MeshUp
[4] https://github.com/editp23/editp23
[5] https://github.com/Tencent-Hunyuan/Hunyuan3D-2.1
[6] https://github.com/DSL-Lab/UniEdit-Flow

Table 3. **Quantitative Comparison across Different 3D Editing Methods on the Eval3DEdit benchmark.** We report results using Uni3D_{pc} ↑ for identity preservation and $\text{Uni3D}_{\text{txt}}$ ↑ for semantic modification, where higher values indicate better performance. Metrics are evaluated across five representative editing categories, including action change, object addition, object removal, object replacement, and style change. The Overall denotes the average performance across all categories.

Method	Action Change		Object Addition		Object Removal		Object Replace.		Style Change		Overall	
	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$	Uni3D_{pc}	$\text{Uni3D}_{\text{txt}}$
Optimization-based 3D Editing Methods												
TextDeformer [15]	0.1321	0.1393	0.2170	0.1293	0.1549	0.1249	0.2579	0.1269	0.1444	0.1288	0.1818	0.1298
MeshUp [26]	0.1202	0.1348	0.1542	0.0962	0.0984	0.0956	0.1588	0.1187	0.1252	0.0987	0.1314	0.1088
LRM-based 3D Editing Methods												
MVEdit [6]	0.4891	0.1204	0.1366	0.0653	0.2224	0.0571	0.2741	0.0859	0.2162	0.0685	0.2677	0.0794
EditP23 [1]	0.1272	0.1527	0.0879	0.0667	0.0733	0.0774	0.1240	0.0862	0.0887	0.0743	0.1002	0.0915
LFM-based 3D Editing Methods												
Direct Editing [53]	0.2611	0.2227	0.2982	0.1878	0.2658	0.1184	0.4006	0.2250	0.2485	0.1274	0.2948	0.1762
Editing-by-Inversion [22]	0.4512	0.2432	0.4990	0.2445	0.5094	0.2320	0.5807	0.2767	0.5317	0.2609	0.5144	0.2515
Inversion-free Editing [27]	0.4701	0.2375	0.5242	0.2377	0.5447	0.2436	0.6052	0.2742	0.5407	0.2566	0.5370	0.2499
AnchorFlow (Ours)	0.3123	0.2756	0.4438	0.2504	0.4501	0.2314	0.5220	0.2788	0.4577	0.2494	0.4372	0.2571

these observations using Uni3D-based identity and semantic metrics (*cf*. Appendix C.1.2), as reported in Tab. 3. The results show a consistent trend in Tab. 1. LRM-based methods exhibit the weakest performance across both identity preservation and semantic modification due to multi-view inconsistency. LFM-based approaches achieve substantially higher scores, confirming the benefit of operating directly in a 3D latent space. Within this category, editing-by-inversion and inversion-free editing obtain strong identity preservation but either rely on expensive inversion or tend to produce under-edited or distorted geometry. In contrast, our AnchorFlow method achieves competitive identity scores while delivering the strongest semantic alignment among all LFM methods, showing that our approach provides a more balanced and reliable editing results. These Uni3D results corroborate the CLIP-based findings and further demonstrate the effectiveness and robustness of our training-free 3D editing pipeline.

D.2. More Qualitative Results.

Qualitative Analysis of Optimization Process. Fig. 9 shows a simple case of adding a blueberry onto an ice cream. We observe that FlowEdit is affected by inaccurate latent anchors, leading to an inconsistent editing trajectory (*e.g.*, the disappearing blueberry and missing ice cream highlighted by red bounding boxes). In contrast, AnchorFlow exhibits a smoother optimization process, enabling gradual editing while keeping the unedited regions largely unchanged.

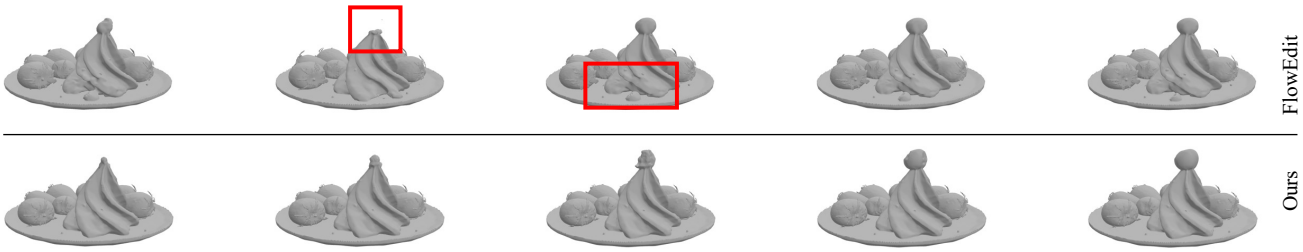

Figure 9. Optimization Process Visualization.

Extension to Image Editing. We further extend AnchorFlow to image editing using Stable Diffusion 3 [14]. Figure 10 presents several examples showing that AnchorFlow is effective for image editing as well. For instance, AnchorFlow generates ceramic lions with greater stylistic consistency.

Additional Qualitative Results on Eval3DEdit. Beyond the qualitative comparisons provided in the main paper, we include additional examples in Fig. 11 and Fig. 12 to further assess the effectiveness of our method. Note that in all comparative experiments, AnchorFlow and the baseline [27] share same hyperparameter settings to ensure a fair comparison. The results

Figure 10. Extension on Image Editing.

demonstrate that our method produces edits that are both semantically faithful to the instruction and geometrically coherent. Across non-rigid editing tasks such as action change (*e.g.*, rows 1–5 in Fig. 11), AnchorFlow generates smooth and globally consistent articulations. In contrast, inversion-free editing often exhibits insufficient deformation (*e.g.*, rows 2 and 4 in Fig. 11) or noticeable distortions in limb shape or body proportions (*e.g.*, rows 1 and 5 in Fig. 11). These additional examples confirm that our method maintains stable geometry even under large pose changes while accurately following fine-grained instructions such as "saluting" and "waving." For rigid editing tasks, such as object addition (*e.g.*, rows 1–3 in Fig. 12), AnchorFlow performs precise and localized modifications without affecting unrelated regions. This further supports that AnchorFlow enables mask-free editing. Overall, the extended qualitative results provide additional evidence that AnchorFlow achieves strong semantic modification while preserving identity across both rigid and non-rigid editing tasks.

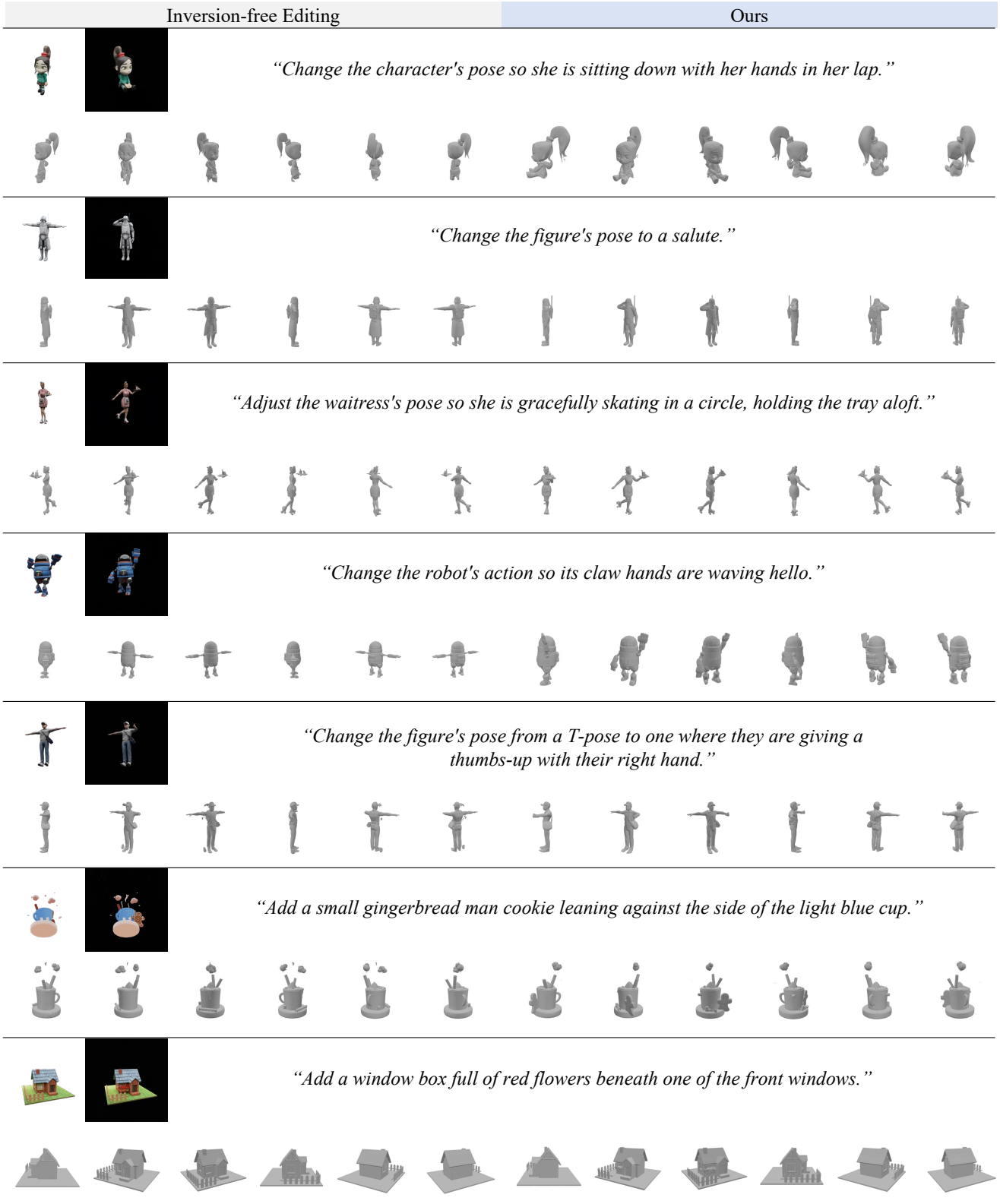

Figure 11. More qualitative results using AnchorFlow.

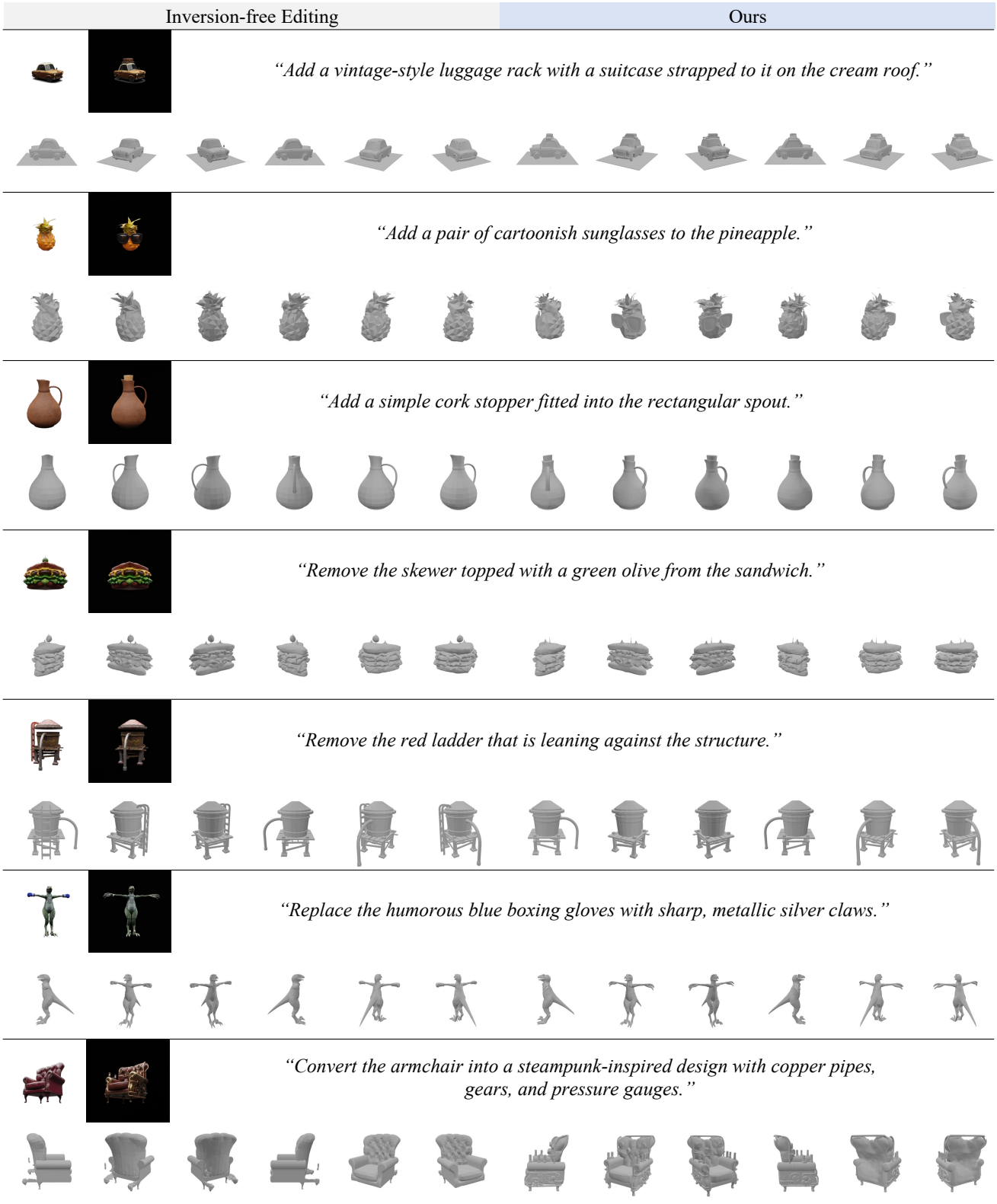

Figure 12. More qualitative results using AnchorFlow.