Edit360: 2D Image Edits to 3D Assets from Any Angle

Supplementary Material

This supplementary material provides in-depth details of the Edit360 system, expanding on the technical and analytical discussions from the main paper including:

- Section 7: A detailed analysis of Video 3D Diffusion Models (V3DM), including V3D [6] and SV3D [54], along with their strengths, limitations, and how Edit360 extends their applications to 3D editing.
- Section 8: Additional implementation details for Edit360, focusing on multi-view generation, LLM-based instruction parsing, and the Spatial Progressive Fusion (SPF).
- Section 9: Provides more ablation experiments.
- Section 10: Discusses failure cases and future works.

7. Additional Details and Analysis of V3DMs

This section provides an analysis of the Video 3D Diffusion Model (V3DM), detailing the methodologies of V3D [6] and SV3D [54], and highlighting their strengths and limitations. Based on these insights, we demonstrate how Edit360 integrates their advantages, overcomes their constraints, and evolves into a powerful, time-efficient tool for 3D editing.

7.1. Technical details of V3D and SV3D

The pipelines for both V3D [6] and SV3D [54] consist of two stages: dense-view image generation and high-quality 3D reconstruction.

Dense-view Video Generation. For dense-view image synthesis, both models fine-tune pre-trained Stable Video Diffusion (SVD) [1] architectures to generate orbital videos of synthetic 3D objects. V3D generates videos by rendering 18 evenly spaced camera positions in the horizontal plane, conditioned on a front-view image for semantic guidance. The frames cover a full 360° view of the object and each frame has a resolution of 512×512 pixels. SV3D generates videos with two types of camera positions: similar to V3D, $SV3D^u$ employs static orbits where all the cameras are positioned at evenly spaced azimuth angles at a fixed elevation; $SV3D^p$ allows for any specified rotation angles and elevations of each camera position at evenly spaced azimuth angles. Both produce videos consisting of 21 frames at a resolution of 576×576 pixels. We choose V3D and $SV3D^u$ for our experiments, as horizontal 360-degree editing effectively meets most editing requirements.

3D Reconstruction. The reconstruction pipelines for V3D and SV3D are designed to transform dense-view video into high-quality 3D assets. V3D not only uses space-carving-based initialization [23, 30] to efficiently position 3D Gaussian splats [20], but it can also extract the surface mesh using NeuS [56], enhancing its utility for real-world applications. SV3D utilizes a two-stage coarse-to-fine reconstruction process: the coarse stage employs a NeRF-based representation [34] to reconstruct the SV3D-generated dense views at a lower resolution, while the fine stage extracts a mesh using marching cubes [7], and refines it with a DMTet [45] representation, applying SDS-based [37] diffusion guidance from SV3D at full resolution. Due to the optimization with SDS, this process requires approximately 20 minutes. For our applications, we use the faster V3D reconstruction method (*i.e.*, without SDS loss), which completes the process in just two minutes. This effectiveness is demonstrated in Figure 10, showcasing examples of high-quality 3D assets reconstructed from dense views.

7.2. More Analysis for Limitations of V3DMs

V3DMs, represented by V3D and SV3D, leverage two advantages of video diffusion models: (1) temporal consistency between frames, and (2) the ability to generate dense sequences. These strengths translate into view consistency in 3D generation and dense multi-view images that are beneficial for subsequent reconstruction tasks.

However, despite these strengths, V3DMs face notable limitations due to their reliance on sparse input views. As discussed in Section 5.3 and illustrated in Figure 8, these limitations include significant texture and shape loss in areas not directly visible in the input view. For instance, as shown in Figure 11, when conditioned solely on a front view of a bunny doll with wings, the model generates an incomplete back side with missing wing details (second row). Conversely, when using a back view input, the model produces a bunny with no recognizable facial identity in the front view (third row). These observations highlight the challenge of achieving coherent 3D representations from single-view inputs.

Edit360 addresses these limitations through its novel Anchor-View Editing Propagation algorithm (detailed in Section 4). Edit360 extends V3DMs to multiple input views, enabling seamless integration of user-provided views with anchor views. Unlike traditional interpolation-based methods, Edit360 injects conditioning inputs across all frames during sampling, ensuring consistent propagation of edits throughout the sequence. As shown in the fourth row of Figure 11, Edit360 integrates edits like adding wings to a bunny doll, ensuring smooth transitions between edited and unedited regions. This approach preserves facial identity from the front view while consistently propagating the editing information from the anchor view to other 360-degree views of the 3D object.

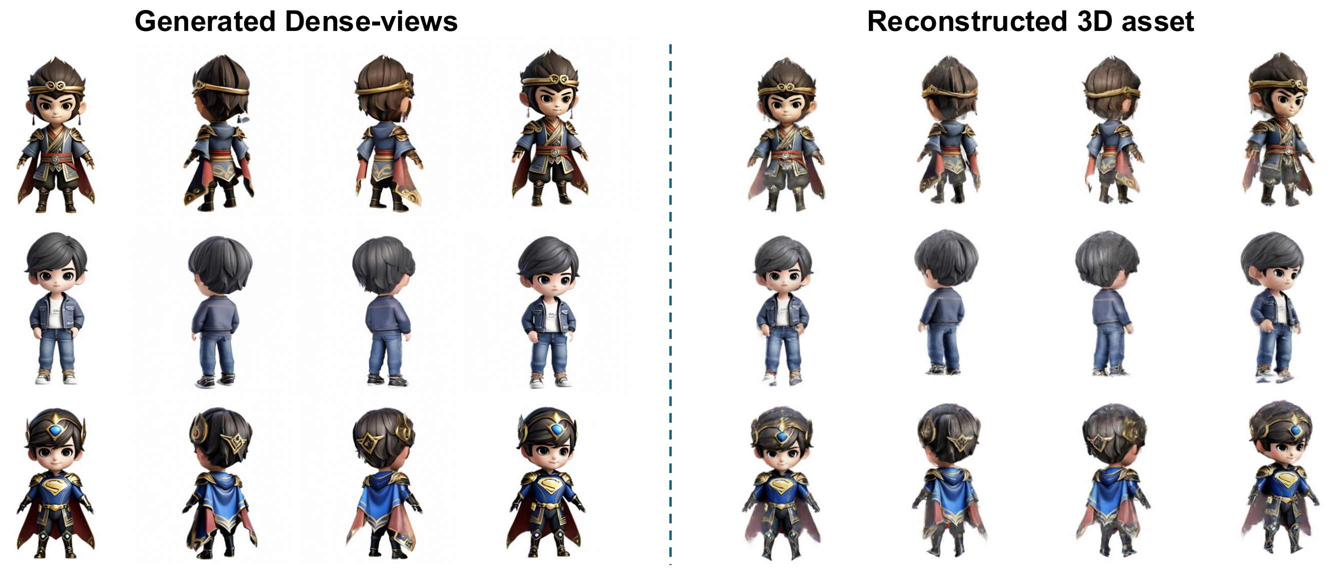

Figure 10. 3D asset reconstruction from dense views using 3DGS [20].

8. Edit360 Pipeline Implementation Details

The Edit360 pipeline is designed to streamline the process of 3D asset editing and reconstruction. The entire workflow, including video generation and 3D reconstruction, can be completed on a single GPU (*e.g.*, NVIDIA RTX 4090 with 24 GB of memory). Below, we detail three key stages of Edit360: multi-view generation (Section 8.1), edit instruction parsing with LLMs (Section 8.2), and the Spatial Progressive Fusion (SPF) (Section 8.3).

8.1. Multi-view Generation

This stage focuses on generating dense and consistent views of a 3D asset, accommodating various input types, including 3D models, single-view images (such as photographs of real-world objects), and text descriptions. We denote the number of generated views per asset as N.

8.1.1. Techniques for Different Input Types

Each input type is processed using tailored techniques to ensure high-quality multi-view generation.

For 3D models, Edit360 utilizes Blender to render N views of the object. The 3D model is imported into Blender, where N cameras are placed around the object in a horizontal plane, evenly spaced at intervals of $360/N$ degrees in a counterclockwise arrangement. Each camera captures a distinct perspective simultaneously, ensuring uniform and consistent coverage of the object across all views.

For single-view inputs, such as a front-view photograph or image, Edit360 utilizes a single-input V3DM (SV3D [54]). The V3DM takes the single input as a reference and generates the remaining $N-1$ views, ensuring consistency and coherence with the original input.

For text inputs, Edit360 first uses a text-to-image generation model, such as DALL·E [39], to generate an image based on the provided textual description. This generated image typically serves as the front view of the object, which is then processed in the same manner as single-view image inputs, using a single-input V3DM (*e.g.* SV3D [54]) to synthesize the remaining $N-1$ views.

8.2. Parsing Editing Instructions with LLMs

Edit360 employs Large Language Models (LLMs), such as GPT-4, to parse user-provided editing instructions and identify optimal perspectives, referred to as anchor views. This process involves analyzing the instruction, breaking them into actionable tasks, and selecting the viewing angles that best suit the editing objectives. The implementation relies on providing a carefully crafted prompt to the LLMs, which guides the model to effectively understand and execute the required tasks. An example of the prompt is as follows:

- **Instruction Analysis and Task Breakdown:**
  - Analyze the editing instructions to identify specific tasks, such as modifying an object's appearance or adding elements.
  - If multiple tasks are present, process each task individually. Further analyze the instruction to pinpoint the specific part of the object or scene requiring modification and evaluate task visibility across different views.
- **Optimal Anchor View Selection:**
  - For each task, determine the most suitable horizontal viewing angle (anchor view) to clearly display the target area for editing.

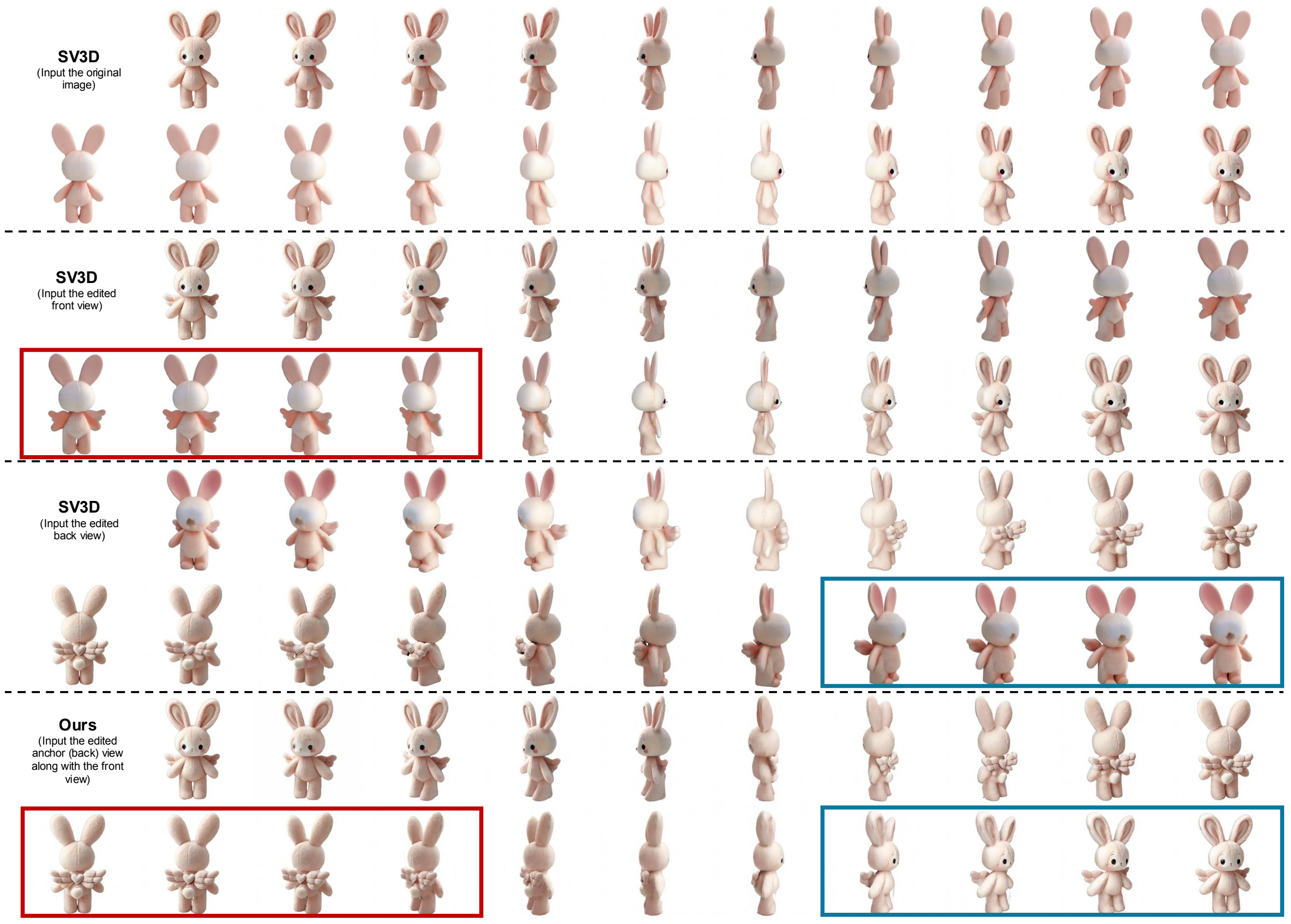
SV3D
(Input the original image)
SV3D
(Input the edited front view)
SV3D
(Input the edited back view)
Ours
(Input the edited anchor (back) view along with the front view)

Figure 11. Dense views of the first editing example in Figure 1, illustrating the clockwise rotation results of SV3D and Ours. With only the edited front view as input, SV3D generates an incomplete back view with missing wing details (red box). Conversely, using the edited back view as input results in missing facial features in the front view (blue box). Our method integrates both views to produce consistent and complete results across all angles.

  - Assess visibility to ensure critical modifications are visible from chosen angles (*e.g.*, a backpack visible from the back or wings requiring views from both front and back). If necessary, select multiple angles to guarantee consistency and coherence across the 3D model.
- **Standardized Output Format:**
  - Output the selected angles as discrete units of $360/N$ degrees, formatted as a comma-separated list (*e.g.*, “1, 9”).
  - Provide only numerical outputs, avoiding additional text or explanations for clarity and ease of integration.
- **Examples for Anchor View Selection:**
  - **Instruction 1:** “Add a backpack and a hat to a person.”
    Optimal Angles: “9, 1”
    (Angle “9” for the backpack visible from the back, and angle “1” for the hat visible from the front.)
  - **Instruction 2:** “Add wings to a person.”
    Optimal Angles: “1, 9”
    (Wings are visible from both front and back views, requiring both angles.)
  - **Instruction 3:** “Place a logo on the left sleeve of a shirt.”
    Optimal Angles: “5”
    (Angle “5” best captures the left side of the person.)

By leveraging this structured prompt, Edit360 ensures the LLM consistently identifies the most appropriate anchor view(s) for each modification task. This not only simplifies the editing process but also enables precise, user-aligned modifications while maintaining coherence across the entire 3D representation.

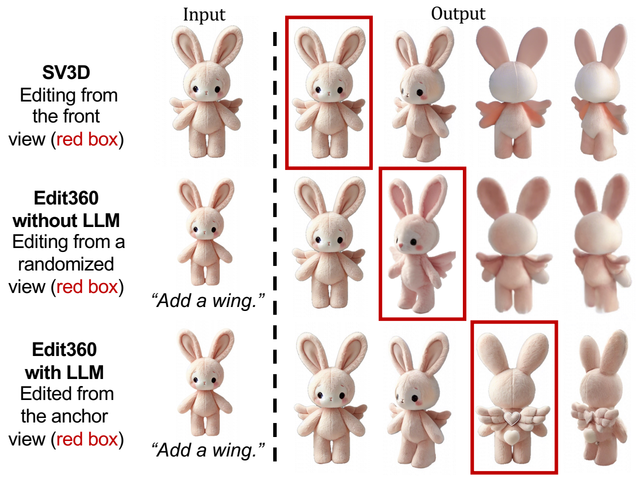
Input
Output
SV3D
Editing from the front view (red box)
Edit360 without LLM
Editing from a randomized view (red box)
"Add a wing."
Edit360 with LLM
Edited from the anchor view (red box)
"Add a wing."

Figure 12. Comparison of different view selection strategies for 3D editing. Top: SV3D editing from the front view fails to effectively show the added wing. Middle: Edit360 without LLM using a random view still produces suboptimal results. Bottom: Edit360 with LLM intelligently selects the back view as the anchor point, resulting in a more appropriate and visible wing placement. Red boxes indicate the views used for editing in each approach.

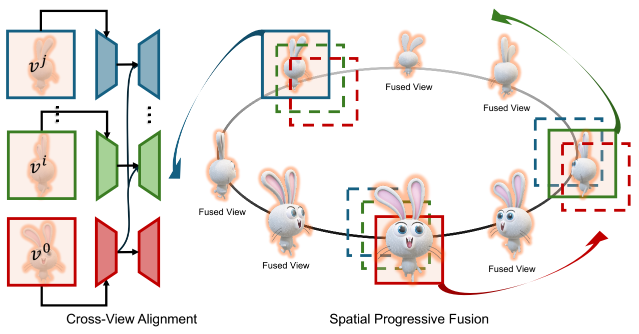

Figure 13. Illustration of CVA and SPF with Multiple Anchor Views (v^i and v^j). Edit360 can integrate multiple anchor views into a seamless 360-degree fused sequence.

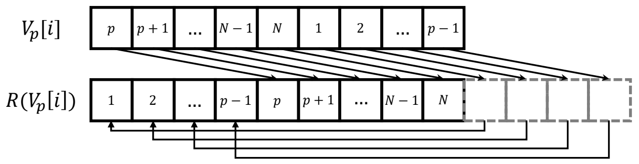

Figure 14. Illustration of cicular-shift (CS) in Equation 4, showing the rotation mapping used to align video sequence X_{anchor} with X_{front}.

8.2.1. Evaluation of LLM View Selection Strategy

As shown in Figure 12, editing with GPT-4-selected views rather than random view allows introducing editing-relevant details in suitable views (*e.g.*, wing details are more visible from the back view than the front/side). Moreover, we evaluate GPT-4's effectiveness in selecting anchor views using a set of 100 editing instructions. GPT-4 successfully identifies the appropriate editing views for all samples.

8.3. More details of Edit360's Key Components

The Spatial Progressive Fusion (SPF) and Cross-View Alignment algorithm (As shown in Figure 13) are designed to propagate user-defined edits across all views in the 360-degree representation, ensuring spatial and visual consistency. This section provides illustration of Spatial Alignment (As shown in Figure 14), detailed explanations of the dynamic weight adjustment strategies used during fusion, techniques for edge and texture extraction in latent space multi-scale fusion.

8.3.1. Detailed Configuration of Spatial Weights

Spatial Weighting (SW), as used in Equation 5, is a crucial component of the SPF algorithm, ensuring consistent propagation of user-defined edits across all frames in the 360-degree representation. This subsection provides a detailed explanation of how spatial weights are configured in the SPF algorithm, including the spatial dynamic adjustment to account for the cyclic nature of video sequences, the temporal dynamics for balancing contributions during sampling steps, and the normalization process to ensure balanced influence from multiple anchor frames. While the following details the dynamic adjustment of spatial weights, in practice, the weights can also be directly set or customized according to specific requirements or prior knowledge.

Spatial Dynamics of Weight Adjustment. In the case of a 360-degree looped video sequence, where the first and last frames are adjacent, so we define the distance to account for the cyclic nature of the sequence. The complete formula for the weight $\alpha_p(i)$ is given as:

$$\alpha_p(i) = \exp\left(-\frac{\min(|i-p|, N-|i-p|)^2}{2\sigma^2}\right), \quad (7)$$

where N is the total number of frames in the video sequence, and $\min(|i-p|, N-|i-p|)$ represents the shortest cyclic distance between frame i and the anchor frame p. This adjustment accurately accounts for the cyclic nature of the video sequence, ensuring smooth and consistent propagation of the anchor frame's influence across all views in the 360-degree representation.

Temporal Dynamics of Weight Adjustment. During the early stages of the sampling process, the goal is to distribute the influence of anchor frames broadly across the sequence, allowing edits to affect distant views. To achieve this, SPF starts with a larger variance (e.g., $\sigma = 5$) in Equation 7. As sampling progresses, since the editing information has already been injected, σ is linearly reduced to a smaller value (e.g., $\sigma = 2$), concentrating the influence of anchor frames on nearby views to refine local details. This progression enables a balanced approach, combining global consistency with precise local adjustments.

Normalization for Balanced Influence. To prevent any single frame sequence from dominating the fusion process, the weights are normalized to ensure that each frame maintains an equal total weight sum. Specifically, for each anchor frame sequence with weight α_p, the corresponding weight assigned to the baseline sequence V_0 is calculated as $1 - \alpha_p$. When multiple anchor sequences are present, the total weight for each frame is normalized by dividing the sum of weights by the number of anchor sequences, N_p. This process is formalized in Equation 5.

8.3.2. Multi-scale Fusion in Latent Space

To enhance visual fidelity and prevent over-smoothing during the progressive fusion process, multi-scale fusion integrates edge and texture features extracted from frame sequences. These features are incorporated to strengthen structural alignment and enrich textural details, ensuring the generated outputs are both consistent and visually detailed. Specifically, we employ Sobel edge detection [19] and a high-pass filter [12] to extract edge and texture details $(E_{t,f}, T_{t,f}, E_{t,a}, T_{t,a})$ from each frame in $X_{t,\text{front}}$ and $X_{t,\text{anchor}}$. The Sobel operator computes horizontal and vertical gradients via 3×3 kernels:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I, \tag{8}$$

where $*$ denotes convolution and I is the input. The edge strength is then:

$$E = \sqrt{G_x^2 + G_y^2}. \tag{9}$$

In addition, we extract high-frequency texture features with a high-pass filter:

$$T = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * I. \tag{10}$$

We then perform fusion across multiple scales following Equation 5 to produce E_t, T_t, and X_t. The final fused frame is computed as:

$$\widetilde{X}_t^i = w_1 \cdot X_t^i + w_2 \cdot E_t^i + w_3 \cdot T_t^i, \tag{11}$$

where $\widetilde{X}_t^i$ denotes the updated estimation of X_t^i. In the progressive fusion process, the weights w_1, w_2, and w_3 are dynamically adjusted to balance contributions from the original sequence, edge features extracted from the original image, and texture features, respectively.

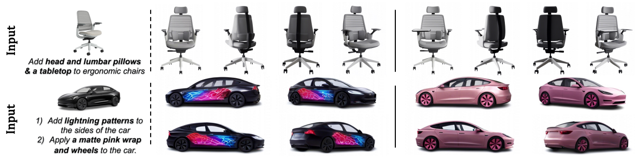
Input
Add **head and lumbar pillows & a tabletop** to ergonomic chairs
Input
1) Add **lightning patterns** to the sides of the car
2) Apply **a matte pink wrap and wheels** to the car.

Figure 15. Additional real-world editing examples using Edit360. Our method produces geometrically consistent and visually plausible results across diverse object categories and viewing angles.

SW	MF	CVA	LPIPS ↓	PSNR ↑	SSIM ↑	CLIP-S ↑	MSE ↓
			0.11	20.57	0.83	0.85	0.02
✓			0.10	21.13	0.85	0.85	0.02
✓	✓		0.08	20.90	0.87	0.85	0.02
✓	✓	✓	**0.07**	**22.17**	**0.90**	**0.88**	**0.02**

Table 4. Ablation study on the key components of the SPF module, Spatial Weighting (SW) and Multi-scale Fusion (MF), showing their individual contributions to performance.

Model	LPIPS ↓	PSNR ↑	SSIM ↑	CLIP-S ↑	MSE ↓
Edit-SV3D (v_0)	0.08	21.70	0.86	**0.89**	**0.02**
Edit-SV3D (v_0 & v_7)	0.08	21.78	0.86	**0.89**	**0.02**
Edit-SV3D (v_0 & v_{11})	**0.07**	22.34	**0.87**	0.88	**0.02**
Edit-SV3D (v_0 & v_7 & v_{14})	**0.07**	**22.37**	**0.87**	0.87	**0.02**

Table 5. Ablation study demonstrating the ability of Edit360 to effectively fuse different numbers of views from various angles.

9. More Ablation

We present more ablation studies based on the experimental settings outlined in Section 5.1.

The Key Components of SPF. As shown in Table 4, we evaluate the impact of two key components of SPF: Spatial Weighting (SW), Multi-scale Fusion (MF). The baseline model use simple linear interpolation with equal weights across all frames. The results highlight the importance of each component in the fusion process.

Impact of Fused Views. As shown in Table 5, using the front view v_0 as a baseline, we experiment with fusing other view inputs like viewpoint v_7, v_{11}, and a combination of v_7 and v_{14}. The results demonstrate robust performance of our Edit360 across all view combinations and angles.

10. Failure Cases and Future Work

Failure cases. While video diffusion models maintain good frame consistency for single-object views, they struggle with spatial relationships in multi-object scenes due to the lack of 3D priors, leading to object adhesion artifacts.

Scene-level applicability. Our method focuses on object-level editing due to the scope of current pre-trained models. With advances in video models for 3D scene generation, extending to scene-level editing is a promising direction.