CVPR #5998

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 Dense-view Diffusion Models (DDM), including V3D [3] and SV3D [14], 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 Frame Fusion (SFF).
- We provide additional examples of outfit customization and style transfer in Fig. 12, demonstrating the effectiveness of our approach.

7. Additional Details and Analysis of DDM

This section provides an analysis of the Dense-view Diffusion Model (DDM), detailing the methodologies of V3D [3] and SV3D [14], 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 [3] and SV3D [14] 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 v 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 [8, 9] to efficiently position 3D Gaussian splats [7], but it can also extract the surface mesh using NeuS [15], 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 [10] to reconstruct the SV3D-generated dense views at a lower resolution, while the fine stage extracts a mesh using marching cubes [4], and refines it with a DMTet [13] representation, applying SDS-based [11] 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 Fig. 7, showcasing examples of high-quality 3D assets reconstructed from dense views.

7.2. More Visualization for Limitations of DDM

DDM, represented by V3D and SV3D, leverages 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, DDMs face notable limitations due to their reliance on sparse input views. As discussed in Sec. 5.3 and illustrated in Fig. 6, these limitations include significant texture and shape loss in areas not directly visible in the input view. For instance, as shown in Fig. 9, 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 Spatial Frame Fusion (SFF) algorithm (detailed in Sec.4.3). SFF extends DDMs to multiple input views, enabling seamless integration of user-provided views with anchor views. Unlike traditional interpolation-based methods, SFF injects conditioning inputs across all frames during sampling, ensuring consistent propagation of edits throughout the sequence. As shown in the fourth row of Fig. 9, 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.

CVPR
#5998

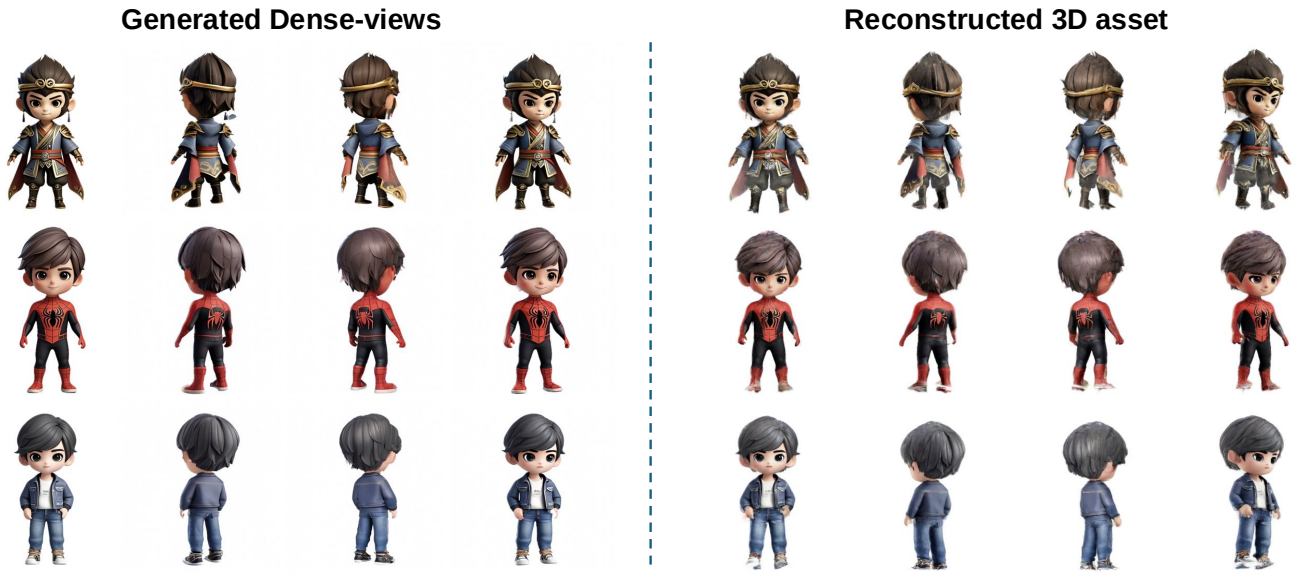

Figure 7. 3D asset reconstruction from dense views using 3DGS [7].

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) within five minutes. Below, we detail three key stages of Edit360: multi-view generation (Sec. 8.1), edit instruction parsing with LLM (Sec. 8.2), and spatial frame fusion (Sec. 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 $360/N$ degree intervals 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 DDM (SV3D [14]). The DDM takes the single input as a reference and generates the remaining $N-1$ views, ensuring consistency and coherence with the original input.

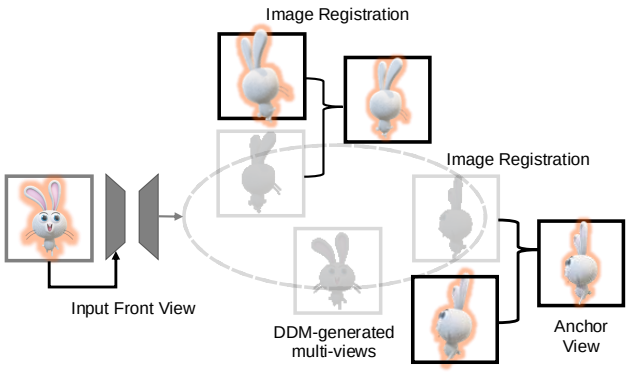

Figure 8. Illustration of Spatial Registration.

For text inputs, Edit360 first uses a text-to-image generation model, such as DALL·E [12], 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 DDM (*e.g.* SV3D [14]) to synthesize the remaining $N-1$ views.

8.1.2. Spatial Registration

Real-world photographs or rendered multi-view inputs often suffer from inconsistencies between views due to variations in camera poses (*e.g.*, the distance between the camera and the object). These inconsistencies can lead to misaligned object edges when anchor views are rotated to align with the front view, disrupting spatial coherence across perspectives.

To address this, we propose spatial registration, a method that leverages the spatially consistent dense-view sequences generated by the Dense-view Diffusion Model (DDM) and incorporates image registration [2, 16] techniques for align-

CVPR
#5998

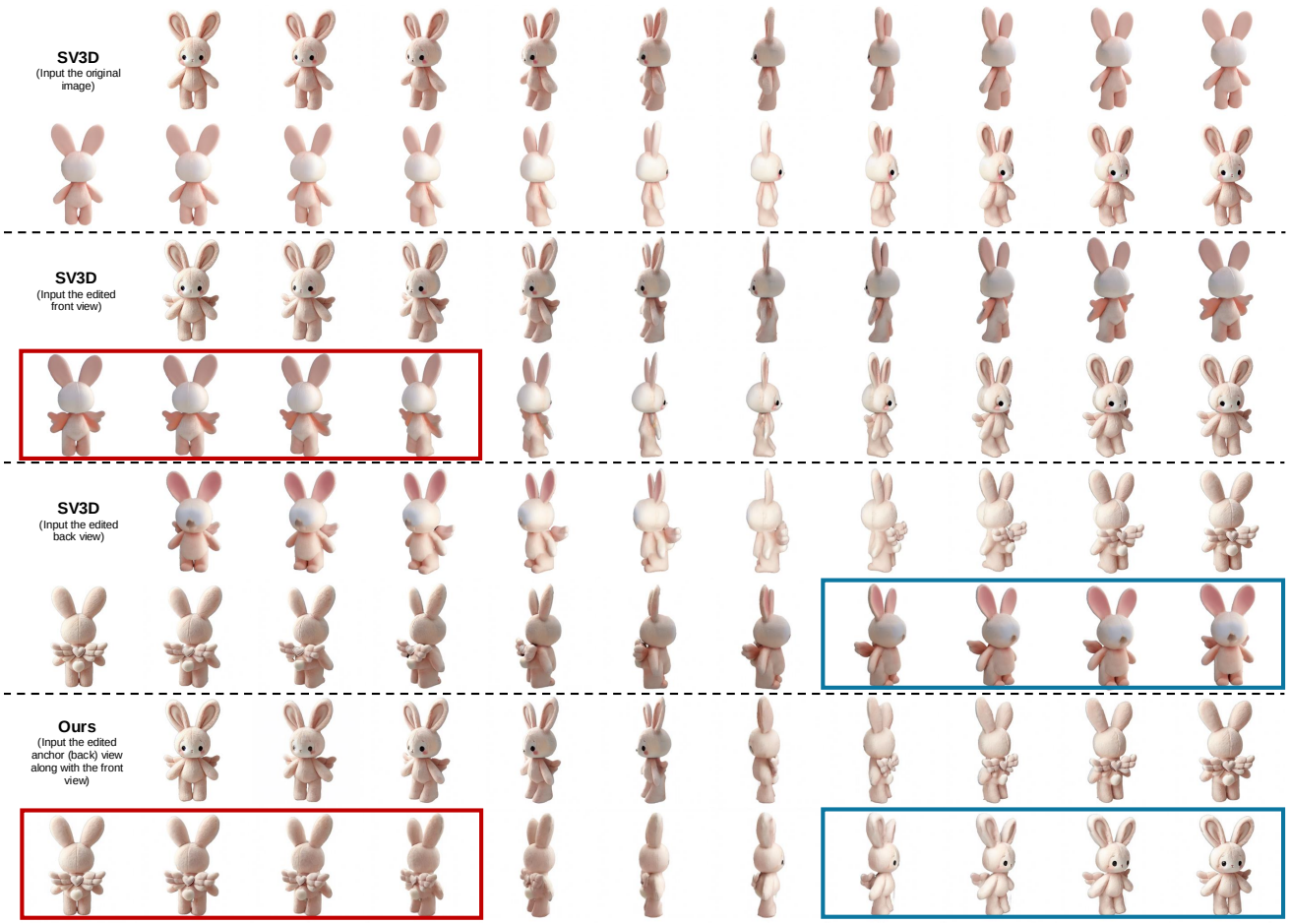

Figure 9. Dense views of the first editing example in Fig. 1, illustrating the clockwise rotation results of SV3D and Ours. With only the edited front view as input, SV3D generates an incomplete back view with missing wing details (red box). Conversely, using the edited back view as input results in missing facial features in the front view (blue box). Our method integrates both views to produce consistent and complete results across all angles.

ment. Spatial registration ensures that anchor views and the front view are properly aligned in 3D space, creating a unified input for subsequent processing.

As shown in Fig. 8, the process begins with the front view input, which is passed through the DDM to generate a dense-view sequence covering all angles. This sequence serves as a spatially consistent reference. For each anchor view, Edit360 selects the corresponding viewpoint from the generated sequence, ensuring both share the same perspective. Since their viewpoints are now consistent, image registration techniques [2, 16] are applied to align the anchor view with the corresponding generated view.

This approach achieves precise spatial registration between anchor views and the front view, seamlessly integrating all views into the dense multi-view sequence. As a result, spatial coherence across perspectives is maintained, ensuring that subsequent fusion sampling operates on fully aligned inputs and produces consistent outputs.

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 it 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. Below is an example of the prompt used:

- **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.
  - 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.

8.3. Spatial Frame Fusion (SFF)

The Spatial Frame Fusion (SFF) algorithm (As shown in Fig. 10) is 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 (Fig. 11), detailed explanations of the dynamic weight adjustment strategies used during fusion, techniques for edge and texture extraction in latent space multi-scale fusion, and the implementation details of Deep-Guided Fusion (DGF).

8.3.1. Detailed Configuration of Spatial Weights

Spatial Weighting (SW), as described in Eq. (3), is a crucial component of the SFF 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

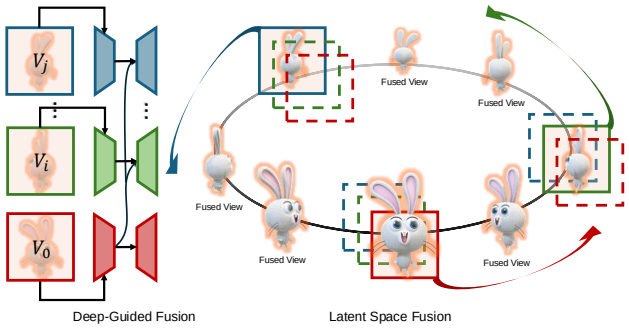

Figure 10. Illustration of SFF with Multiple Anchor Views (V_i and V_j). Edit360 can integrate multiple anchor views into a seamless 360-degree fused sequence.

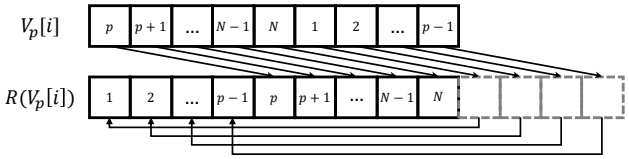

Figure 11. Illustration of Spatial Alignment in Eq. (2), showing the rotation mapping used to align video sequence V_p with V_0.

SFF 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.

Spatial Dynamics of Weight Adjustment. In Eq. (3), for simplicity, we initially defined $|i-p|$ as the direct distance between frame i and the anchor frame p. However, in the case of a 360-degree looped video sequence, where the first and last frames are adjacent, this definition does not accurately capture the spatial relationships. To address this, we redefine the distance to account for the cyclic nature of the sequence. The complete formula for the weight $\alpha_p(i)$ is given as:

$$\alpha_p(i) = \exp\left(-\frac{\min(|i-p|, N-|i-p|)^2}{2\sigma^2}\right), \quad (6)$$

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, SFF starts with a larger variance $\sigma = 5$ in Eq. (6).

CVPR #5998

As sampling progresses, since the editing information has already been injected, σ is linearly reduced to a smaller value, $\sigma = 2$, 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 Eq. (4).

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, the method employs Sobel edge detection to capture key structural edges and Gabor filters to extract fine texture patterns.

Sobel Edge Detection. Sobel edge detection [6] identifies high-frequency structural features by calculating image gradients at each pixel, determining the direction of maximum intensity change and the rate of change in that direction. It employs two 3×3 convolution kernels: one for the horizontal gradient G_x and another for the vertical gradient G_y:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I, \tag{7}$$

where $*$ denotes the convolution operation and I is the input image. The gradient magnitude E is then computed for each pixel:

$$E = \sqrt{G_x^2 + G_y^2}. \tag{8}$$

This magnitude represents the edge strength at each pixel, ensuring that critical boundaries are preserved during fusion.

Gabor Filters. We use Gabor filters [5] for texture analysis and extraction, as they are designed to be sensitive to specific spatial frequencies and orientations. The standard 2D Gabor filter is expressed as:

$$g(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right), \tag{9}$$

where:

$$x' = x\cos\theta + y\sin\theta,$$
$$y' = -x\sin\theta + y\cos\theta.$$

In this formulation, λ controls the wavelength (spatial frequency), θ specifies the orientation, ψ is the phase offset, σ determines the scale via the Gaussian envelope, and γ defines the elliptical aspect ratio. These filters emphasize texture patterns in specific orientations, capturing fine details in the frames.

Weight Adjustment During Fusion. In the progressive fusion process, the weights w_1, w_2, and w_3 are dynamically adjusted during the last 30 of the total 50 sampling steps to balance contributions from the original sequence, edge features extracted from the original image, and texture features, respectively. Throughout these 30 steps, multi-scale fusion is applied, with weights transitioning linearly from $w_1 = 1, w_2 = 0, w_3 = 0$ to $w_1 = 0.6, w_2 = 0.2, w_3 = 0.2$ by the 45th step. This gradual adjustment helps prevent excessive smoothing of fine details. In the final 5 steps, only the original sequence is used, focusing on refining coherence and finalizing the generated sequence.

8.3.3. Deep-Guided Fusion

The Deep-Guided Fusion (DGF) algorithm is a critical component of Edit360, designed to ensure consistent and high-quality outputs by leveraging the U-Net architecture's skip connections to integrate reference and anchor views seamlessly.

The U-Net architecture, commonly used in video diffusion models, consists of a symmetric encoder-decoder structure. The encoder gradually reduces spatial dimensions while increasing the feature channel depth, and the decoder restores spatial dimensions while refining details. Skip connections play a vital role by directly transferring intermediate feature maps from encoder layers to their corresponding decoder layers, preserving high-resolution spatial details that might otherwise be lost during downsampling.

In our approach, to reduce inconsistencies stemming from the randomness inherent in the denoising process (as described in Sec. 4.3), we preserve the skip features from the first six layers (of a total of twelve) within the encoder that processes the reference video sequence V_0. Throughout the downsampling phase of the anchor view sequence V_p, we integrate these skip features from V_p with those retained from V_0, applying spatial alignment and spatial weighting techniques as specified in Eq. (2) and Eq. (3), respectively. Once the integration is complete, these enhanced skip features are channeled into the upsampling layers of V_p via skip connections. This strategic insertion ensures that the reference details from V_0 effectively steer the generation of more consistent and detailed outputs in V_p.

Figure 12. Here are more examples of outfit customization and style transfer. Starting with the front and back views of a little boy (generated by a text-to-image model), we used Edit360 to apply various styles, such as "Magician Style," "Iron Man Style," and more, achieving seamless outfit customization and style transfer. The outputs feature multi-angle views of the customized designs, demonstrating the flexibility and effectiveness of style transfer in character customization.

Figure 13. Dense views of the examples in Fig. 1.

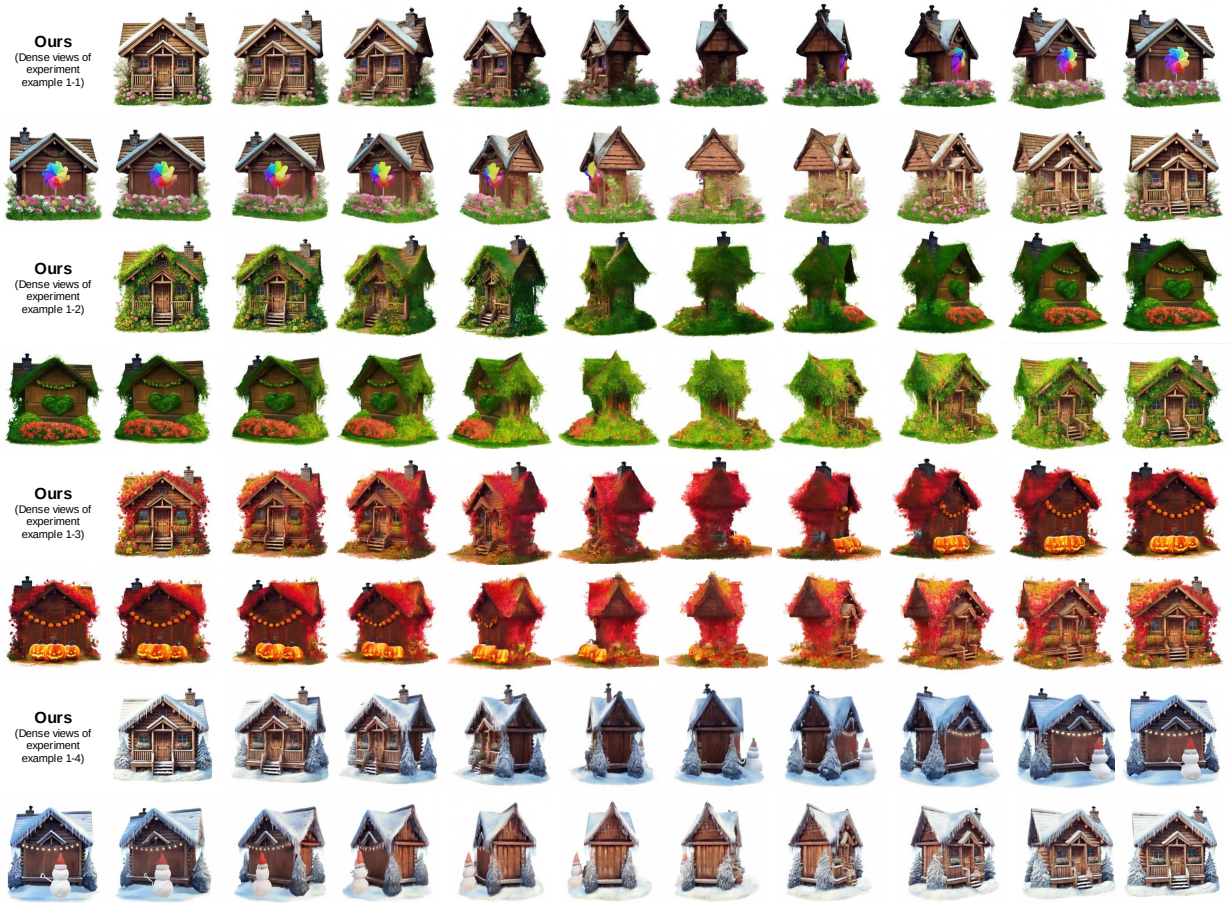

Figure 14. Dense views of seasonal element edits on houses in Fig. 5.

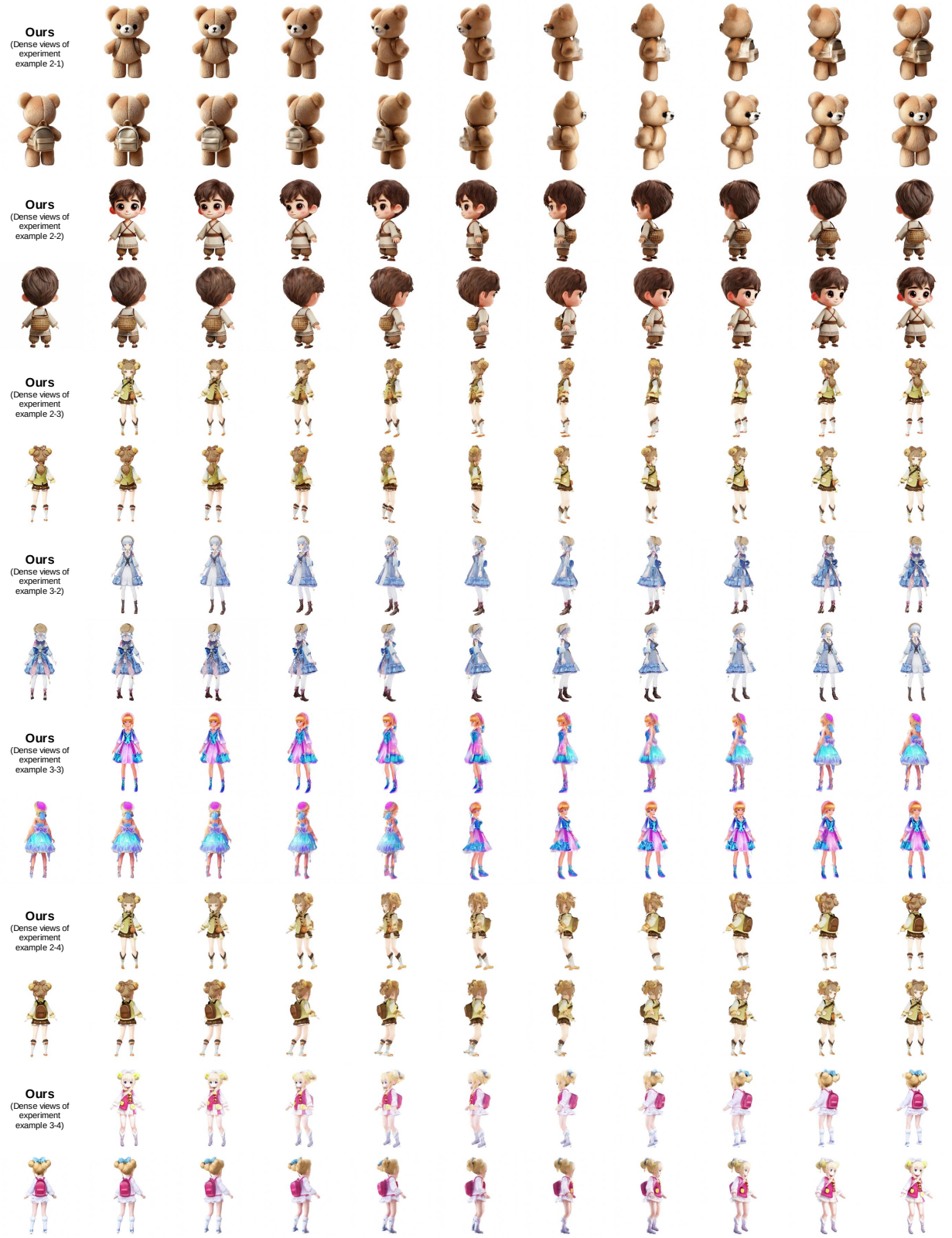

Figure 15. Dense views of the experimental results in Fig. 5.

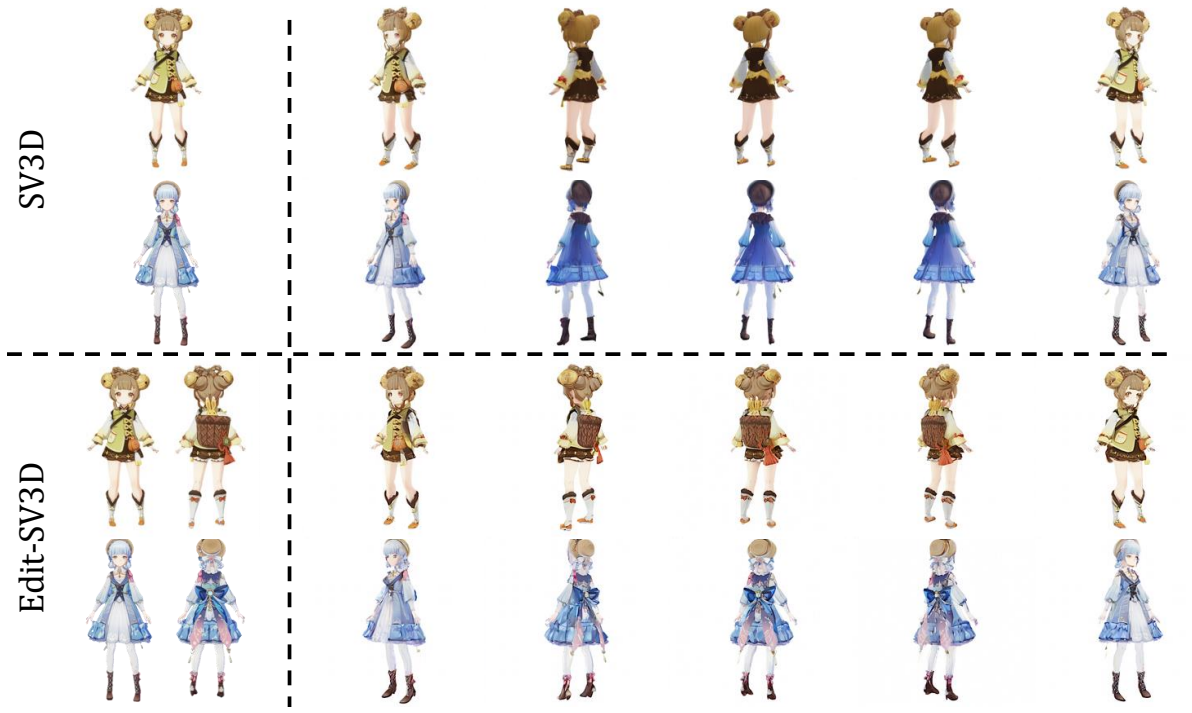

Figure 16. Additional examples illustrating the limitations of the SV3D in outfit customization when using single-view input.

Figure 17. Dense views of qualitative comparison in Fig. 6.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1

[2] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM CSUR*, 1992. 2, 3

[3] Zilong Chen, Yikai Wang, Feng Wang, Zhengyi Wang, and Huaping Liu. V3d: Video diffusion models are effective 3d generators. *arXiv preprint arXiv:2403.06738*, 2024. 1

[4] Daren BH Cline. Admissibile kernel estimators of a multivariate density. *The Annals of Statistics*, 1988. 1

[5] I. Y. Fogel and Dov Sagi. Gabor filters as texture discriminator. In *Biological Cybernetics*, 1989. 5

[6] N. Kanopoulos, N. Vasanthavada, and R.L. Baker. Design of an image edge detection filter using the Sobel operator. In *IEEE Journal of Solid-State Circuits*, 1988. 5

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *ACM TOG*, 2023. 1, 2

[8] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. In *IJCV*, 2000. 1

[9] Yuanxun Lu, Jingyang Zhang, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, Long Quan, Xun Cao, and Yao Yao. Direct2. 5: Diverse text-to-3d generation via multi-view 2.5 d diffusion. In *CVPR*, 2024. 1

[10] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM TOG*, 2022. 1

[11] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 1

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, 2021. 2

[13] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. 1

[14] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *ECCV*, 2025. 1, 2

[15] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *NeurIPS*, 2021. 1

[16] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 2003. 2, 3