# SINE: Semantic-driven Image-based NeRF Editing with Prior-guided Editing Field Supplementary Material

Chong Bao[1*] Yinda Zhang[2*] Bangbang Yang[1*] Tianxing Fan[1]
Zesong Yang[1] Hujun Bao[1] Guofeng Zhang[1†] Zhaopeng Cui[1†]
[1]State Key Lab of CAD&CG, Zhejiang University [2]Google
https://zju3dv.github.io/sine/

In this supplementary material, we describe more details of our method, including model architecture in Sec. A, dataset preparation in Sec. B, and implementation details in Sec. C. Besides, we also conduct more experiments in Sec. D. More qualitative results can be found in our supplementary video, and the source code will be released upon the acceptance of this paper.

## A. Model Architecture

We first explain the details of the model architecture. Specifically, we adopt the multi-resolution voxel-hashing encoder by Müller *et al*. [14] as the coordinate-based encoder, and build the template NeRF and the editing field in a decoupled manner. The voxel-hashing encoder is constructed with 16 levels with 2-dimensional features for each level. For the template NeRF, we use the voxel-hashing encoder to encode the queries' coordinates and use spherical harmonics with 4 degrees to encode the ray direction. The density and color heads for model output consist of 1 hidden layer with 128 hidden size and 2 hidden layers with 64 hidden size, respectively. As introduced in Sec. 3.1, the editing field consists of a geometric modification field $F_{\Delta G}$ and a texture modification field $F_{\Delta T}$. The geometric modification field $F_{\Delta G}$ and the corresponding forward modification field $F'_{\Delta G}$ are both constructed with an MLP of 1 hidden layer and 128 hidden size with the ReLU activation, and we adopt the positional encoding [13] (with 4 frequencies) to all input query points. The texture modification field $F_{\Delta T}$ is constructed with a voxel-hashing encoder (same size as the template NeRF), followed by an MLP of 1 hidden layer and 128 hidden size with ReLU activation. During the dual volume rendering stage, we follow Mildenhall *et al*. [13] by using 64 coarse samples and 128 fine samples for each ray, and render the deformed template image

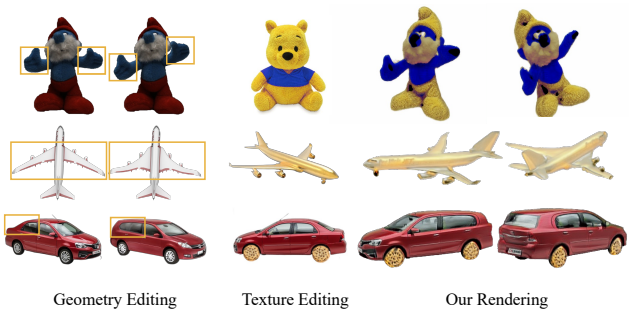



Figure A. We show examples of hybrid object editing by combining geometric and texture editing.

$\hat{I}_o$ and color modification image $\hat{I}_m$ with the same density values. Then, as explained in Sec. 3.3, we use a color compositing layer to obtain the edited view $\hat{I}$ by blending $\hat{I}_m$ into $\hat{I}_o$, where the color compositing layer is constructed using a compact UNet-like structure (with 2-layer encoder (3 $\rightarrow$ 16 $\rightarrow$ 32) and a symmetrical decoder, all layers comprise 3$\times$3 convolutions). Besides, we can integrate temporal attribute [17, 18] (from 0 to 1) to the input of $F_{\Delta G}$ (with the positional encoding of 4 frequencies), and train the geometric editing on the edited transitions with temporal attributes as conditions, *e.g*., the dynamic motion effect shown in the supplementary video.

## B. Dataset Preparation

We evaluate SINE on both real-world/synthetic and object/scene datasets. Specifically, for the real-world car datasets [19], each sequence contains 72 images with a car rotating on the turntable. We use Colmap [20] to recover camera poses w.r.t the cars' centers for all the images. For the data of Photoshape [16], EditNeRF [10] does not provide edited GT images, so we regenerate all testing cases using Blender, which is more challenging than the original ones (*e.g*., we stretch the whole chair or enlarge holes, while EditNeRF [10] only fills a tiny hole or removes legs).

*Authors contributed equally.
†Corresponding authors.

For the data Blenderswap [11], we render the scenes with Blender's Cycle engine with realistic environment HDR maps. For users' 2D image editing, we use Adobe After Effect / Photoshop to deform images (geometric editing) and paint patterns (texture editing), and use EditGAN [9] and Text2LIVE [1] to edit images with semantic strokes or text-prompts. For users' target images (*e.g.*, cars and chairs) from the Internet, we remove their backgrounds using remove.bg [4] before conducting texture editing.

## C. Implementation Details

**Training details.** As introduced in the main paper, our method performs semantic-driven editing upon the given NeRF model. Specifically, for each object or scene, we first train a generic template NeRF model. Then, we learn geometric editing and texture editing with editing from a single perspective. For geometric editing, since the geometric changes are sometimes combined with minor color changes, we also fine-tune the color modification field with a photometric loss (see the first term in Eq. (3)). For texture editing, the texture transferring loss (Eq.(6))is defined on a complete image, which is not compatible with NeRF's sparse ray supervision. Therefore, we adopt the deferred back-propagation technique from Zhang *et al*. [24] for texture editing. Practically, we first render the full-sized $\hat{I}_o$ and $\hat{I}_m$ and forward the color compositing layer, and then compute the losses to cache the complete image gradient w.r.t the $\hat{I}_m$ and the color compositing layer, and re-render the $\hat{I}$ and back-propagate the gradients to the $F_{\Delta T}$ and the color compositing layer in a patch-wise manner. To make a smooth convergence, we take the coarse-to-fine regularization [17] on the color modification field by progressively increasing the frequency band of the input features during the training process. Furthermore, we randomly perturb the pose to augment the data distribution and avoid the overfitting of the texture editing. The whole training process of the template NeRF and our editing field takes about 12 hours on a single Nvidia RTX 3090 graphics card.

**Preparation of proxy mesh in geometric editing.** As introduced in Sec. 3.2, we use a proxy mesh to represent NeRF's geometry during geometric editing. In practice, we directly obtain the proxy mesh using off-the-shelf tools (*i.e.*, implicit surface reconstruction method NeuS [23]). Since we optimize DIF [3] latent code $\hat{\mathbf{z}}$ and deform the proxy mesh $\hat{M}$ during the editing, the initial proxy mesh should be binding to a latent code beforehand. Therefore, we obtain the initial latent code $\hat{\mathbf{z}}$ to the corresponding initial proxy mesh $\hat{M}_\sigma$ in an auto-decoding manner [3, 15] before training.

**Cycle loss in geometric editing.** During the training of geometric editing, we additionally train a forward modification field $F'_{\Delta G}$ to map the template proxy mesh to the edited space. The forward modification field $F'_{\Delta G}$ and the implicit geometric modification field $F_{\Delta G}$ are both supervised with a cycle loss [8, 12], which is defined as:

$$\mathcal{L}_{\text{cycle}} = \frac{1}{M}\sum_{i=1}^{M}||F_{\Delta G}(F'_{\Delta G}(\mathbf{p}_i)) - \mathbf{p}_i||+ ||F'_{\Delta G}(F_{\Delta G}(\mathbf{p}_i)) - \mathbf{p}_i||, \tag{1}$$

where $\{\mathbf{p}_i|i = 1, ..., M\}$ is the uniform point samples in 3D space, and we set $M = 1000$ in our experiment.

**Feature-cluster-based semantic masking.** As introduced in Sec. 3.4, we train a 3D feature field with DINO-ViT's feature maps, and generate feature clusters from the user-painted regions, which will be used to compute semantic masks to distinguish foreground editing areas and background areas. Specifically, we first render the feature map under the specific editing view, and sample 1000 feature points on the user's painted region (which is directly accessible from the editing tools). Then, we use K-Means to generate $K = 15$ clusters from the sampled feature points. During the training stage, we first render the current training view's feature map, and compute the L2-normalized pixel-wise feature distance (from 0 to 1) to the nearest clusters. The pixels with distances smaller than 0.5 would be marked as foreground objects, and the others would be marked as background. These computed editing masks would be used to regularize both geometric and texture editing (see Eq. (7)) to maintain the irrelevant content unchanged.

**User study.** The questionnaire contains 17 cases, 8 for target-image-based editing (*e.g.*, Fig. 7 (a)) and 9 for text-prompts editing (*e.g.*, Fig. 7 (b)). We show the participants a source image, a target image/text prompts, as well as the results produced by different methods. Participants are asked to select one result that best matches the style of the target image or the text meaning.

## D. More Experiments

**Hybrid editing with geometric and texture changes.** We can combine geometric and texture editing on the same object by optimizing geometric-related losses and texture-transferring losses in turns. As shown in Fig. A, we can edit objects' geometries while transferring textures with users' target images, *e.g.*, the plush toy raises its hands and is painted in new textures from a yellow bear, and the airplane extends its wings and is painted golden. Please refer to our supplementary video for a vivid animation of these effects.

**3D editing field vs. template NeRF fine-tuning.** In this experiment, we compare our 3D editing field with the naïve fine-tuning template NeRF (which is adopted by CLIP-NeRF [22] and DFF [7]). Editing NeRF with only a single image is fairly ambiguous without external supervision (*e.g.*, semantic hints). For a fair comparison, we provide external supervision for the baseline method (vanilla NeRF).


(a) Direct fine-tuning NeRF's density layers vs. Ours with geometric modification field

(b) Direct fine-tuning NeRF's color layers vs. Ours with texture modification field

Figure B. We compare our editing field with directly fine-tuning template NeRF.

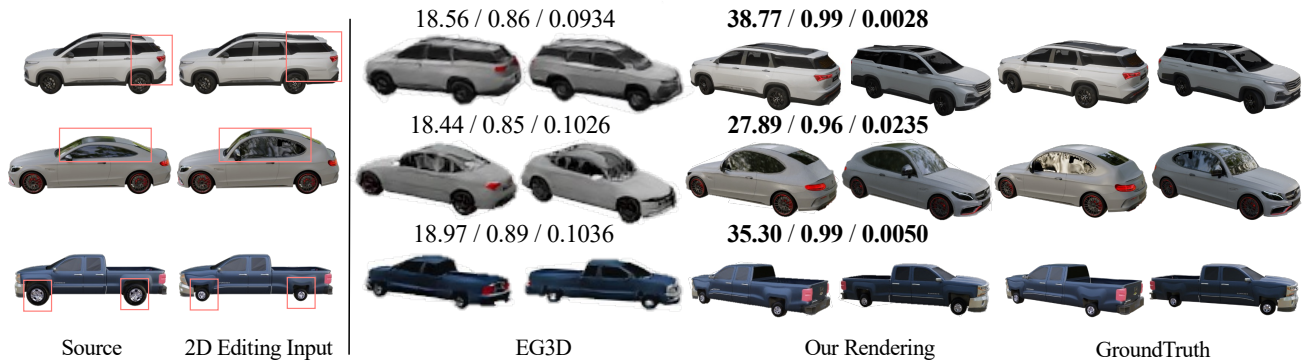



Figure C. We show the quantitative comparison between our method and EG3D [2] on the synthetic cars [11], where the metrics of PSNR↑ / SSIM↑ / LPIPS↓ are annotated above.

Specifically, for texture editing, we enable NeRF's related network layers to be optimized and use the same texture editing losses. Directly fine-tuning NeRF's color layers can change the objects' texture to some extent but cannot reach the same quality as our full model (*e.g.*, uncovered cookie tires and snowy flowers in Fig. B (b)). For geometry editing, we fine-tune vanilla NeRF with $\mathcal{L}_{\text{gt}}$ and $\mathcal{L}_{\text{reg}}$ since it is not trivial to apply SDF-based shape priors to vanilla NeRF. As demonstrated in Fig. B (a), naïvely fine-tuning NeRF on geometric editing would lead to the overfitting to a single view, and the multi-view consistency is no longer ensured (*e.g.*, broken wings and green floaters in Fig. B (a)).

**Quantitative comparison with EG3D on synthetic cars.** We conduct quantitative comparisons with the SOTA 3D-aware GAN method EG3D [2] on the synthetic car dataset. To obtain the ground-truth images of the edited results, we use Blender to render the training and testing views, and modify the cars' geometry within the software. As shown in Fig C, our method achieves better rendering quality than EG3D on both visual quality and all the metrics (PSNR, SSIM, and LPIPS). For example, we can preserve the specular effect even after the editing (*e.g.*, the specular area facing the light source and the reflection on the windshield), while EG3D struggles to produce photo-realistic results due to the limitation of its learned 3D latent representation.

**Comparison with 2D GANs.** We compare our method against the SOTA 2D semantic editing method EditGAN [9] on the real-world car dataset [19]. To make a fair comparison, we train our NeRF's backbone and EditGAN's style codes on all the multi-view images (*i.e.*, each style code corresponds to one view). Then, we perform semantic 2D editing on one single view using EditGAN. For our method, we use the edited view to train our editing field. And for EditGAN, we save the intermediate editing vector and add the editing vector to all the style codes, which yields multi-view edited images. As demonstrated in Fig D, since EditGAN is agnostic to the 3D geometry, its results suffer from the inconsistent issue between different views, *e.g.*, poor inversion results for the head and tail of the car, and the semantic editing result cannot be precisely applied to all views. In contrast, our methods can synthesize cars with multi-view consistency and high-quality editing results.

**More ablation studies on geometric supervision.** As shown in Fig. F, since ablating $\mathcal{L}_{\text{gt}}$ (Eq. (3)) makes it no supervision on editing, we split it into photometric (b) and silhouette (c) terms, and the absence of either will result in distorted or washed-out texture. **(d)**: When ablating deformation reg. loss $\mathcal{L}_{\text{gr}}$ (Eq. (4)), the edited object is severely


Figure D. We show the comparison of our method with EditGAN [9] on the real-world car dataset [19].

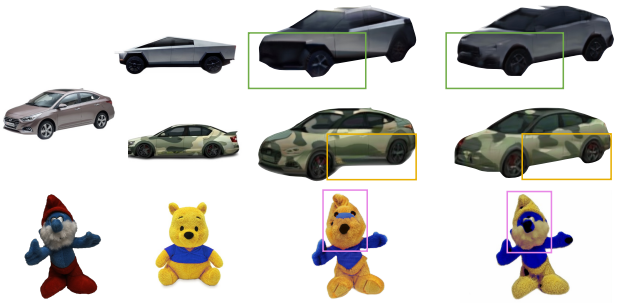



Figure E. We compare our method with SPLICE-ViT [21] on real-world cars [19] and toys [5]. Since our editing method is built upon 3D-aware models, we consistently achieve better texture-transferring results than SPLICE-ViT when the source and target are observed from different perspectives of views (*e.g.*, cars) or with significant different shapes (*e.g.*, plush toys).

distorted (*e.g.*, the letters are stretched). **(e)**: The cycle loss $\mathcal{L}_{\text{cyc}}$ (Eq. (1) in supp.) brings constraints from shape prior to geometric mod. field $F_{\Delta G}$, and ablating it would lose the efficacy of semantic guidance (*e.g.*, the twisted airplane).

**Ablation study of texture supervision.** In Fig. G, we disable texture transfer loss $\mathcal{L}_{\text{tex}}$ (Eq. (6)) and utilize photometric loss to paint the target texture, which leads to incomplete

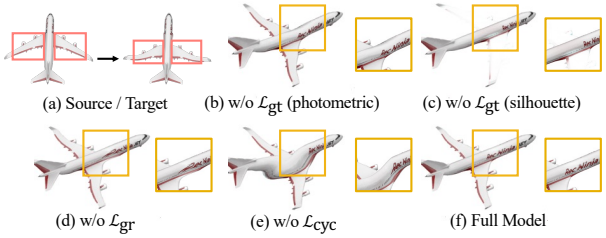



Figure F. We inspect the efficacy of different constraints in geometric editing.

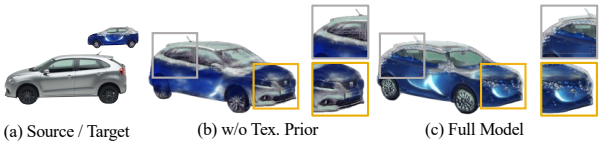
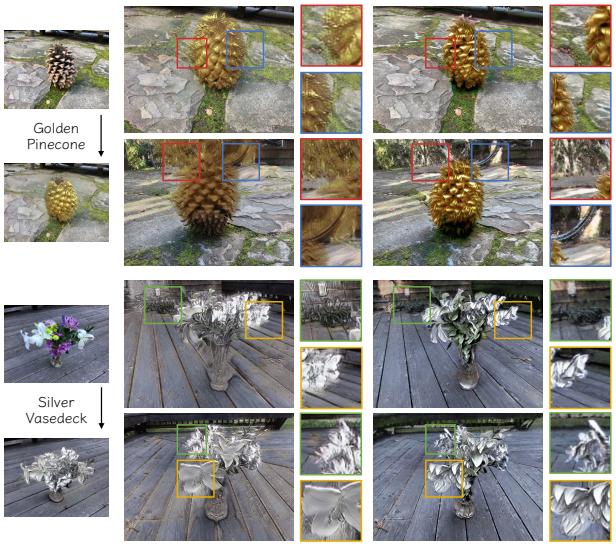



Figure G. We inspect the efficacy of the texture prior constraint in texture editing

texture transferring results for invisible parts as shown below. Besides, without texture prior supervision, we cannot transfer textures between objects with different shapes.

**Comparison of texture editing with image-based SPLICE-ViT.** Our texture transferring loss (Eq. (6)) is inspired from SPLICE-ViT [21], but fully leverages the multi-view training scheme. Therefore, we compare our texture editing with image-based SPLICE-ViT in Fig. E. As shown in Fig. E, SPLICE-ViT is sensitive to the perspective difference of the source and target images, which results in overfitting appearances on the edited view, *e.g.*, horizontal straight patterns of cars when observing cars from a slightly tilted view, distorted faces of the plush toy. By contrast, our method consistently achieves better texture-transferring results with color patterns properly aligned to the cars' geometries and the plush toy's body parts.

**Comparison of texture editing with Text2LIVE.** As shown in Sec. 4.4 from the main paper, since our method only requires one single-view image as editing input, we can naturally achieve text-prompt-based texture editing by cooperating with off-the-shelf text-driven editing methods (such as Text2LIVE [1]). A follow-up question is, how does the Text2LIVE itself perform to the same 360° dataset in our texture editing task? For video editing, Text2LIVE uses layered atlas [6] to convert objects and backgrounds into separated 2D layers. However, in the unbounded 360° dataset (*e.g.*, pinecone and vasedeck [13]), there is no proper way to unwrap 3D objects and scenes into 2D layers (and we also failed to train layered atlas on these 360° datasets). Therefore, we directly apply its converged editing generator to the multi-view images. As shown in Fig. H, although Text2LIVE produces similar-looking edited images, it cannot maintain multi-view consistency when the viewpoint changes (*e.g.*, blurry edges at the golden pinecone,




(a) Source/Target (b) Text2LIVE Multi-Views (c) Our Multi-Views

Figure H. We compare our method with Text2LIVE on texture editing, where our method achieves better multi-view consistency. See the text for details.


Figure I. We show the robustness of geometric editing on proxy meshes with different qualities. The proxy meshes are jittered by adding gaussian noise with different variances (from $0.001^2$ to $0.01^2$).

uncovered petals at the silver vasedeck, and the occasionally affected background). On the contrary, our method naturally takes advantage of multi-view training and consistently delivers more plausible and realistic novel views.

**Robustness to the noise of proxy mesh.** The geometry prior guidance uses the proxy mesh to supervise the geometry modification field. Therefore, we analyze the effect of mesh quality on our editing results. Specifically, we add 3 groups of gaussian noise to the vertices of the proxy mesh and conduct training of our editing field. As shown in Fig I, our method can robustly learn geometric editing even with noisy proxy mesh (*e.g.*, with the gaussian noise of $N(0, 0.004^2)$ in the third column).

**More comparison results on texture editing.** We show more comparison results on the texture editing task with ARF [24], CLIP-NeRF [22] and DFF [7] in Fig. K (where Fig. K (a) is the source view and the target view produced by Text2LIVE [1]). For CLIP-NeRF [22], since the official codebase has not been fully released, we use our own implementation by fine-tuning NeRF's color-related field with CLIP loss, and both use the target features from text embedding and the image embedding (with the same target images in Fig. K (a)), which are denoted as CLIP-NeRF (Image) and CLIP-NeRF (Text), respectively. For DFF [7], we adopt the official codebase and use the texts for NeRF editing and background ray-filtering according to the document. In Fig. K, we omit the DFF's object-centric comparison on the car, since it mainly focuses on scene-level decomposition and editing. As demonstrated in Fig. K, NeRF stylization methods like ARF cannot precisely edit fine-grained effects on the desired location. NeRF fine-tuning approaches like CLIP-NeRF and DFF only change appearance colors, but cannot produce vivid effects (*e.g.*, the burning pinecone or ice sculpture cars). Note that although DFF uses the semantic-field guided decomposed rendering to maintain the background color unchanged, this strategy is not compatible with our color compositing mechanism since we introduce an additional 2D CNN layer to blend the template and editing color for better visual appearance. By contrast, our method both achieves realistic and appealing editing effects, and also effectively preserves background content, and the results are consistently preferred by most of the participants in the user study (see Sec. 4.4).

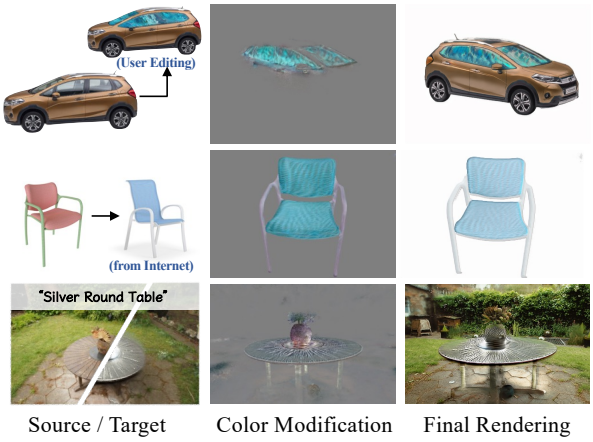



Figure J. We show more rendering results from color modification field and compositional layer.

**Impact of texture modification field & color compositional layer.** The texture modification field learns detailed modifications and the compositional layer blends the original and modified rendering to produce the final edited results, as demonstrated in Sec. 4.5 and Fig. 8 (a). Here we show more rendered texture mod. field (a.k.a. color mod. $\hat{I}_m$) in Fig J.

**Deformation with topology changes.** Our method does not support deformation with topology changes such as breaking the plate, but can provide a visually plausible result by making the "broken part" white, as shown in Fig L. In the future, we can integrate more flexible representations such as ambient slicing surface [18] into our model.

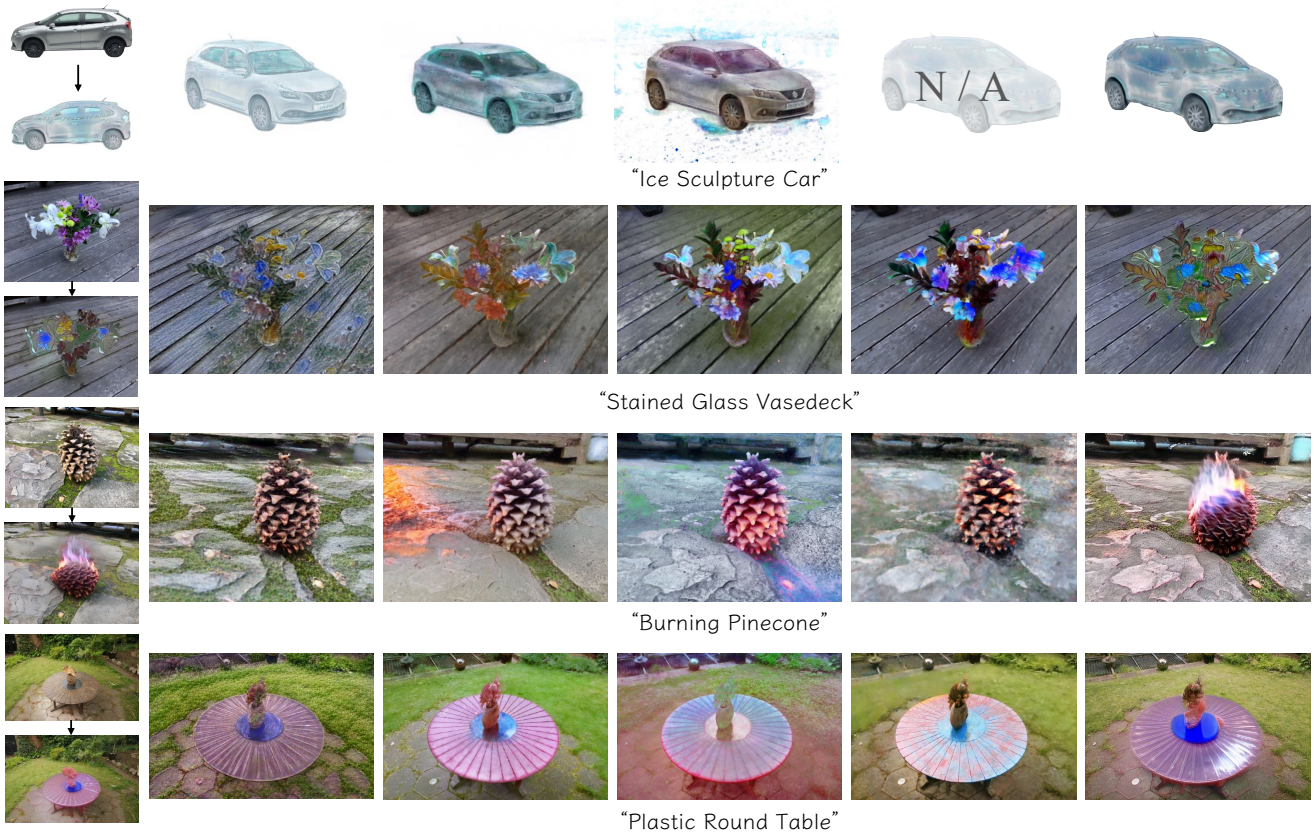



Figure K. We show more comparison results of texture editing with ARF, CLIP-NeRF, and DFF on the real-world car [19] and 360° scene dataset [13]. Our method consistently achieves more realistic and appealing editing results than the others.

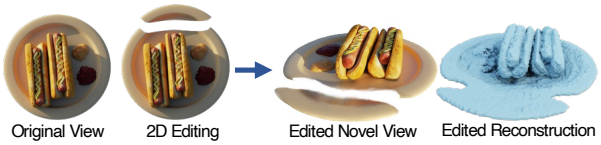



Figure L. We show the results of geometry editing with topology changes.

## References


[1] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. *arXiv preprint arXiv:2204.02491*, 2022. 2, 4, 5

[2] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021. 3

[3] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021. 2

[4] Kaleido AI GmbH. removebg.bg. https://www.remove.bg/, 2022. Accessed: 2022-11-10. 2

[5] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large Scale Multi-view Stereopsis Evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 4

[6] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021. 4

[7] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *arXiv preprint arXiv:2205.15585*, 2022. 2, 5

[8] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[9] Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. Editgan: High-precision semantic image editing. *Advances in Neural Information Processing Systems*, 34:16331–16345, 2021. 2, 3, 4

[10] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5773–5783, 2021. 1

[11] John Roper Matthew Muldoon. Blenderswap. https://www.blenderswap.com/, 2022. Accessed: 2022-11-10. 2, 3

[12] Marko Mihajlovic, Yan Zhang, Michael J Black, and Siyu Tang. Leap: Learning articulated occupancy of people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10461–10471, 2021. 2

[13] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 4, 6

[14] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 1

[15] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2

[16] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), Nov. 2018. 1

[17] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1, 2

[18] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 1, 5

[19] Arun Kumar Sahlam. Carwale. https://www.carwale.com/, 2022. Accessed: 2022-11-10. 1, 3, 4, 6

[20] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 1

[21] Narek Tumanyan, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Splicing vit features for semantic appearance transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10748–10757, 2022. 4

[22] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2, 5

[23] 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. *NeurIPS*, 2021. 2

[24] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 2, 5