GeoDiffuser: Geometry-Based Image Editing with Diffusion Models Supplementary Material

Rahul Sajnani[1,2] Jeroen Vanbaar[2] Jie Min[2] Kapil Katyal[2] Srinath Sridhar[1,2]
[1]Brown University [2]Amazon Robotics

A. Qualitative Results

We present more qualitative results towards the end of the document in Figure 10 and Figure 11. We also compare our method against prior works in Figure 9 for 2D edits.

B. Implementation Details

The shared attention along with the loss functions defined in the manuscript, enable performing geometry image edits as a reverse diffusion process by optimizing the latents and text embeddings. To make the optimization faster, we optimize every alternate step for the initial 32 diffusion steps. We set an initial learning rate of 1.5 and linearly decay it to 0. We share attention across all blocks within the UNet till step 45. All our experiments are performed on an Nvidia RTX3090 with a run time of 30 seconds (for removal) up to 45 seconds (for 2D and 3D edits) on image resolution of 512. Our timing is inclusive of the DDIM inversion, optimization with feature re-projection, and edit generation. We use [16] for projecting, splatting, and rendering in our attention sharing mechanism. Occasionally, the histogram of the edited image does not match the input image and we match color histograms between the two. We detail attention sharing mechanism in Algorithm 1 and editing with GeoDiffuser in Algorithm 2.

C. Evaluation and Baselines

We detail the procedure to perform geometric edits using all our baselines. We also perform a timing and performance analysis of each baseline.

C.1. FreeDrag [6]

Implementation: FreeDrag extends DragDiffusion [18] to perform drag edits with better point tracking. We use the diffusion version in the official FreeDrag implementation which works better on real-world images for our evaluation. For each edit, we first uniformly sample 40 points within the object mask and use the per pixel transform $\mathcal{F}$ to get the target points of the drag to edit images using FreeDrag. This ensures that the same geometric transform is used for editing for a fair comparison. Sampling more points increased the edit time and did not improve the results. Each FreeDrag edit performs 200 LoRA steps with text prompt followed by 1000 drag optimization steps. We had to increase the optimization steps from 300 to 1000 in their implementation as FreeDrag did not converge correctly for large edits tested in our work with 300 steps.
Timing Analysis: The 200 LoRa optimization steps runs for 116.26 seconds and the optimization using 1000 steps runs for 165.24 seconds.
Performance Analysis: We notice that FreeDrag optimization averages nearby drag vectors and does not adhere to the edit. Additionally, it often stretches objects as it does not have removal capabilities baked into the optimization and does not track points appropriately for large edits while our method produces plausible results while being significantly faster (see Figure 9 and Fig. 4 manu.).

C.2. Diffusion Handles [12]

Implementation: We use the official implementation from the authors of Diffusion Handles. Each edit utilizes the depth map and camera transformation to perform the geometric edit. Diffusion handles first performs a null-text inversion using the depth to image stable diffusion model and then inpaints the foreground region of the object using LaMa [19]. The inpainted image is then used to estimate the background depth of the scene. The background depth is blended with the transformed foreground object. This transformed depth map along with transformed activations of the depth to image SD model is then used to generate the edited image as detailed in [12]. Additionally, we had to change the camera FOV to 49.92° to ensure that the same transformation is applied during the edit.
Timing Analysis: Each edit requires 60 seconds of Null-text optimization followed by 35 seconds of edit.
Performance Analysis: We notice that [12] fails to preserve the image content and style, but adheres to the foreground transformation well. However, the image style is not preserved when the depth maps are not predicted using [15] because they are not in the training distribution of Depth to Image Stable Diffusion model. This leads to low Clip Similarity (CS) and degradation in content preservation as shown in the qualitative comparisons if our manuscript. However, we do not have this limitation and can leverage depth maps from any monocular depth estimator. Another limitation for Diffusion Handles is the reliance on multiple depth predictions (for foreground as well as background) and then merging the foreground depth with the background depth. The image generated using this merged depth map

Algorithm 1 Geometric Attention Sharing

Require: ${}^{e}Q$(edit query), ${}^{e}K$(edit key), ${}^{r}Q$(ref. query), ${}^{r}K$(ref. key), ${}^{r}V$(ref. value), $\mathcal{F}$(transformation), M_{obj}(object mask)
Ensure: ${}^{\mathcal{O}}Y_{\text{edit}} :=$ AttentionSharing$({}^{e}Q, {}^{e}K, {}^{r}Q, {}^{r}K, {}^{r}V, \mathcal{F}, M_{obj})$

```
^G Y_ref := Attention(F(^r Q), ^r K, ^r V)        ▷ Reference Guidance and Applying Transform F
if SelfAttention then                              ▷ If Self-attention block
    ^G Y_edit := Attention(^e Q, ^r K, ^r V)       ▷ Use reference key
else
    ^G Y_edit := Attention(^e Q, ^e K, ^r V)       ▷ Use edit key
end if
if DiffusionCorrection then                        ▷ If Diffusion Correction (see Appendix J)
    ^O Y_edit := ^G Y_edit   ▷ ^G Y_edit automatically finds correspondences between ^e Q and ^r K to correct the transformation enabling plausible edits.
else
    ^O Y_edit := F(M_obj) · ^G Y_ref + (1 − F(M_obj)) · ^G Y_edit
end if
return ^O Y_edit
```

Algorithm 2 Geometric Editing with GeoDiffuser

Require: ${}^{r}z_0$(reference latent), $\mathcal{F}$(transformation), M_{obj}(object mask), Φ(null-prompt or optional text)
Ensure: ${}^{e}z :=$ GeometricEdit$({}^{r}z_0, \mathcal{F}, M_{obj}, \Phi)$

```
{^r z_T, ^r z_{T−1}..., ^r z_1} ← DDIMInversion(^r z_0, Φ)        ▷ Reference Inversion
^e z := ^r z_T; ^r z := ^r z_T                                    ▷ Initialize edit latent with reference latent
for t = T → 1 do
    if (t ≤ 30) AND (t %2 == 0) then                              ▷ Optimize
        _, _, L_dict := DiffusionStep([^r z, ^e z], Φ, F, M_obj, t)   ▷ Diffusion Step with Attention Sharing and Loss Dictionary Computation
        L := AdaptiveLoss(L_dict)                                 ▷ Weigh losses adaptively and sum
        ^e z := ^e z − ∇_{^e z} L; Φ := Φ − ∇_Φ L                 ▷ Optimization Update by backpropagating through the diffusion model
    end if
    _, ^e z, _ := DiffusionStep([^r z, ^e z], Φ, F, M_obj, t)
    ^r z := ^r z_{t−1}                                            ▷ Update reference latent with inversion trajectory for Direct Inversion [5]
end for
return ^e z
```

produces improper object removal and at times replaces the object with another instance of the same type. 2D edits with [12] were not good as a constant depth for foreground object was not producing good results even after null-text optimization.

C.3. Dragon Diffusion [10, 11]

Implementation: We use the 2D movement feature of the official Dragon Diffusion implementation for 2D edits and 40 drag points for 3D edits. We use the camera projection, mask, and depth maps to get the target point locations similar to the FreeDrag implementation. We also tried using 100 drag points to perform 3D edits, but this made results worse as the edit moved objects partially, introduced holes, and did not preserve its appearance. For 2D edits, its movement feature utilizes an object mask, a source point and a target drag location. We use the IP adapter [11] that is trained for editing as well for this benchmark, but it did not edit real images very well. We had to increase the weights for ϵ_{opt} and $\epsilon_{content}$ losses for better object removal and content preservation to perform real-world edits.

Timing Analysis: Dragon Diffusion performs inversion in 4 seconds and uses an optimized implementation that edits images in 20 seconds. This method is quick as it doesn't deal with 3D geometry projection and uses the memory bank to speed up the generation process. We can leverage the memory bank to speed up our model as a future work.

Performance Analysis: Dragon Diffusion does not perform well to inpaint disocclusions or preserve the foreground. It has a marginally high clip similarity score as it does not completely remove the object from the source introducing duplicates.

C.4. Zero123-XL + LaMa [8, 19]

Implementation: For this baseline, we first use [19] to inpaint the region of the removed foreground object. We then Zero123-XL to predict the novel view of the transformed object and composite it to the in-painted background image using Laplacian pyramid blending.

Timing Analysis: Zero123-XL + LaMa takes about 5 seconds to run for each edit

Performance Analysis: Zero123-XL moves the object and LaMa removes the object, but it fails to preserve the foreground accurately as it is not in the model's training distribution. It is also difficult to control the per-pixel transform accurately with Zero123-XL as it infers object geometry from the model's learned distribution resulting in high **MD** and **WE** metrics compared to our work.

C.5. Diffusion Self Guidance (DSG) [4]

Implementation: We ran the official implementation of DSG from the authors but it did not perform well for real-images as the authors provide code only for running on generated images. We instead use the implementation of [22] and incorporated DDIM inversion to preserve details of the input image that improved the quality of results using Stable Diffusion V1.4 model. The original work uses Imagen model which is not available. We transform the shape using the transform $\mathcal{F}$ in our paper and use the shape guidance from Eqn. 9 of the DSG paper to penalize for movement which works better according to authors compared to centroid guidance. We had to double the shape and appearance guidance from the default implementation for real images.
Timing Analysis: This implementation uses 50 DDIM steps to perform edits and takes 50 seconds to edit.
Performance Analysis: DSG often loses appearance details when the shape guidance is large or does not move the object when the appearance guidance is large. This primarily occurs because it does not dis-entangle appearance and geometry accurately leading to improvement of appearance at the cost of movement or vice versa. The geometric attention sharing mechanism of our work dis-entangles geometry from appearance leading to more accurate edits both qualitative and quantitatively (see manuscript Tab. 1, Fig. 4 and supplement Fig. 9)

Note that we use prompts for baselines: FreeDrag, Dragon Diffusion, Diffusion Handles, Diffusion Self Guidance and do not require prompts for editing using GeoDiffuser. Additionally, we perform all timing analysis using Nvidia RTX 3090 on the same node. The metric evaluations for all the methods use the default editing parameters from the official implementation unless mentioned otherwise above.

D. Edit Attention Progression

We show the edit progression over different reverse diffusion time-steps in Figure 1. We visualize the top principal component of the self-attention map and show the movement of the car as the optimization progresses. Note that the shadow (dark) region in the attention map also shifts with the car. Transforming the reference query and then computing the attention map transforms the shadows as well (see Figure 1).

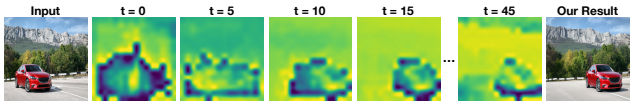

Figure 1. **Attention Progression**: We visualize the principal components of the self attention maps within the first *up-block* layer during editing. At earlier time steps ($t = 5$), the attention is transitioning to move the car, but eventually moves the car to the desired location at $t = 45$. Transforming the attention map shifts the attention corresponding to the shadow of the car.

Camera Projection: We set the camera FOV 49.92° for all edits in our work and we do not require any dataset specific camera intrinsic matrix.

E. Metrics

Mean Distance (MD): We use the mean distance metric from [18]. [18] perform drag based edits in their work and have source as well as their corresponding target drag locations. The mean distance metric computes correspondences between the input and the edited image using DiFT [20] and then estimates the difference between the target edit location and the predicted target location obtained using DiFT. In our case, all pixels in the object foreground become the source edit location, however, finding DiFT correspondences for each foreground pixel is very compute intensive. Hence, we find interest points using SiFT [9] in the foreground of the source image and treat them as the source edit location. We can then obtain the target edit location using the transform $\mathcal{F}$ estimated using camera projection. We then compute DiFT correspondences for these interest points and compute the mean distance metric.
Warp Error (WE): The mean distance metric only measures edit adherence for interest points. We instead warp foreground of the source image and compute an L1 error. This metric measures the error between the warped foreground source image and the edited image. It measures preservation of the foreground object as well as how well it adheres to the edit.

The mean distance is analogous to Re-projection error and the Warp Error is analogous to Photometric error from the Computer Vision literature.
Clip Similarity (CS): We often notice degrade in background and content preservation after the edit. To ensure that the edits do not degrade the contents of the image, we compute the clip image embeddings [14] of the source and the edited image. We then use these embeddings to estimate the cosine similarity between them to measure content preservation between them.

A good editing approach should have low Mean Distance (MD) and Warp Error (WE) as well as have high Clip Similarity (CS).

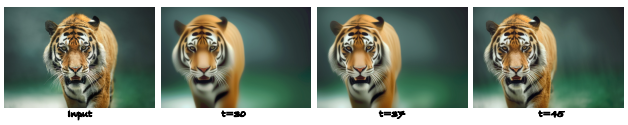

Figure 3. **Geometry Guidance**: Increasing steps t for geometric attention sharing better preserves object style (translation edit).

F. Ablations

We perform a visual ablation of our design choices. Figures 2 and 3 shows the importance of the attention sharing mechanism and adaptive optimization. We can see a degradation in style preservation of the edit when we don't per-

Figure 2. Ablation of adaptive optimization. Without adaptive optimization, the same losses successfully inpaint some images while others fail (middle row). With our adaptive optimization, the same loss function works well for any image.

form geometric attention sharing till step 45. Without the adaptive optimization, we need image specific tuning for loss weights which is not scalable.

In Figure 4, we use our general editing framework to perform the same edit using various Stable Diffusion models.

Figure 4. **Editing ablation using different Stable Diffusion Models**: We perform the same edit using different versions of Stable Diffusion. Notice how the line is incomplete in some cases and the inpainted backgrounds are different. Our geometric attention sharing mechanism ensures that the foreground adheres to the edit and stays the same.

G. Perceptual Study

We conducted a perceptual study with 53 participants to measure the efficacy of inpainting the background and benchmark GeoDiffuser against Zero123-XL. Our perceptual study was conducted using Qualtrics [1]. We first conducted a pilot study having 2 images per division type with 3 users to ensure that all questions are clear. These users did not participate in the final study. After getting feedback from the pilot study we conducted the full study. Each participant completed the study within 10 minutes. They were allowed to click and enlarge images for better inspection. We randomized the order of options presented in the study to avoid biases. In total we presented 70 images (30 for removal, 40 for other transforms) from our dataset. The questions were divided in three categories: edit realism (ER), edit adherence (EA), and removal edit realism (RRE).

For removal, we generated results with LaMa [19], and for the remaining two categories, results were generated with Lama followed by Zero123-XL [8]. Each participant answered 12 ER questions, 12 EA questions and 6 RRE questions, for a total of 30 visual questions. In total 53 users participated in the study for which they received no compensation.

Figure 5 shows the participant preference rate for each division of the study. For RRE, out of the 318 choices, participants preferred our method in 94.06% of the time, which shows that GeoDiffuser is better able to inpaint the disoccluded background regions, especially removing shadows (see Figure 11).

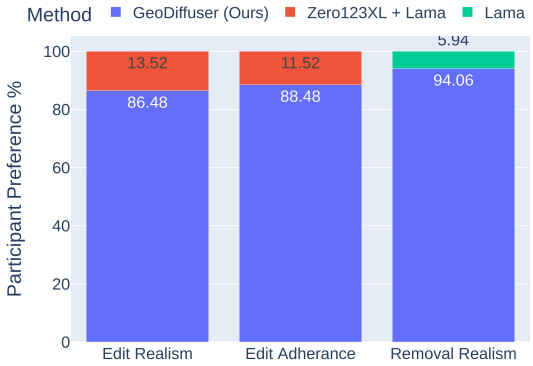

Figure 5. Results from perceptual study show that participants prefer our edits over [7] and [19] in a majority of the cases.

For ER, our method was preferred 86.48% out of 636 cases. This demonstrates that GeoDiffuser preserves object style better than other methods, especially in transforming shadows and reflections. Finally, for EA we included included 16 2D and 24 3D edits. Our method was preferred 88.48% out of 636 cases. This demonstrates that our method more faithfully performs the intended edit, even challenging ones such as 3D rotation. Whereas the baseline is only capable of performing edits from a more narrow range.

H. Failure Cases

Figure 6 displays examples where our method does not perform well. The generation capabilities of the diffusion model at times produces sub-optimal solutions for foreground and background of the image. Additionally, similar to prior works, we can not generate novel views with large rotations and this is a future direction to explore.

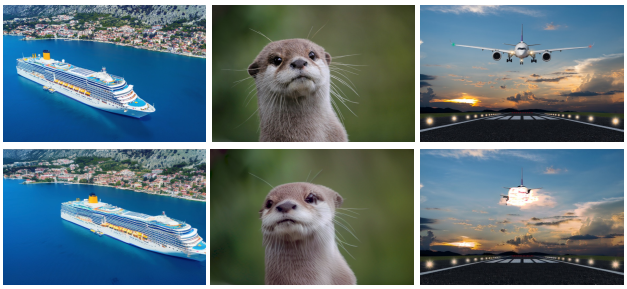

Figure 6. **Failure Cases**: Each example presents the input image at the top followed by the edited image at the bottom. As our geometric edits are performed in a lower dimensional latent space, we face aliasing and interpolation artefacts as shown in the yellow regions of the ship (left). Occasionally our optimization results in sub-optimal solutions for foreground (middle) and background dis-occlusions (right).

Algorithm 3 Object Removal Loss Algorithm

Require: ${}^{r}Q, {}^{r}K, {}^{e}Q, {}^{e}K$
Ensure: $\mathcal{L}_{remove} :=$ RemovalLoss$({}^{r}Q, {}^{r}K, {}^{e}Q, {}^{e}K)$
  if SelfAttentionBlock **then**
    ${}^{\mathcal{G}}A_{edit} :=$ AM$({}^{e}Q, {}^{r}K)$ ▷ Shared Attention Map
  else if CrossAttentionBlock **then**
    ${}^{\mathcal{G}}A_{edit} :=$ AM$({}^{e}Q, {}^{e}K)$ ▷ Shared Attention Map
  end if
  ${}^{\mathcal{G}}A_{ref} :=$ AM$({}^{r}Q, {}^{r}K)$
  $\rho_{obj\to bg}, u_{bg} :=$ torch_max(torch_bmm$({}^{\mathcal{G}}A_{edit}, {}^{\mathcal{G}}A_{ref}) \odot M_{bg}, -1)$ ▷ Foreground to background correlation
  $\rho_{obj\to obj}, \_ :=$ torch_max(torch_bmm$({}^{\mathcal{G}}A_{edit}, {}^{\mathcal{G}}A_{ref}) \odot M_{obj}, -1)$ ▷ Foreground to foreground correlation
  $d_{obj\to bg} :=$ NormalizedCoordinateDistance(u_{bg}) ▷ Coordinate distance to the background location having maximum correlation
  $\mathcal{L}_{remove} :=$ mean $\left(e^{-d_{\text{obj}\to\text{bg}}}(ln(\rho_{\text{obj}\to\text{obj}}) - ln(\rho_{\text{obj}\to\text{bg}}))\right)$

I. Miscellaneous Edits

Our method enables object duplication by turning off the optimization or setting the removal loss to 0 (Figure 7).

Figure 7. Foreground duplication by reducing the turning off optimization or setting the removal loss weight to zero.

J. Diffusion Correction

Occasionally, edit transforms $\mathcal{F}$ are incorrect. For instance, a straight line might be mapped to a jagged curved line. In these cases, it is important for the editing method to marginally disregard the desired edit and preserve the content of the image. This reduces adherence to the edit and produces better results. We can also control this in our attention sharing mechanism by allowing the diffusion model to self-correct and find correspondences for more realistic results as shown in Algorithm 1. This plays a crucial role in edits with sharp geometric structures such as buildings etc (see Figure 8). We enable Diffusion Correction for the last 15 reverse diffusion steps in our experiments.

Figure 8. Diffusion Correction to correct transforms and aliasing.

K. Object Removal

We detail the object removal loss in Algorithm 3.

L. Amodal Loss

Transforming foreground objects drastically introduces depth smearing. We add a small penalty to each edit to force inpainting of the foreground object in these smeared regions using the amodal loss on the amodal mask M_{amodal} obtained by interpolating features after reprojection as

$$\mathcal{L}_{amodal} := \text{mean}(M_{amodal} \cdot ||{}^{\mathcal{G}}Y_{\text{edit}} - \text{interp}(Y_{\text{ref}})||_1). \quad (1)$$

M. Future Work & Impact

We present GeoDiffuser, a method that performs geometric transform on objects to edit real-world images. Our method only requires performing geometric manipulation to the attention layers of the model along with optimization to perform the desired edit. This assumption makes our method very general and better adhere to edits that can be leveraged by future works for geometric analysis of diffusion models and editing in video diffusion models. Another interesting future direction is to perform unsupervised novel view synthesis for real-world scenes by leveraging key ideas from our work that might be able to improve Score Distillation Sampling [13].

N. Discussion on Concurrent Works that Train on Video Data

Concurrent works such as InstaDrag [17], DragNUWA [21], & MagicFixup [3] perform drag edits by training over video data. We detail the advantages & disadvantages of these works and similar works without testing some of these implementations as they are not public. Two advantages of these works include: 1) the inpainting for in/near-distribution images will be accurate with better novel view synthesis of foreground object and 2) faster inference. However, these methods and in-general video diffusion models have the following dis-advantages that need further exploration: 1) They require large scale training datasets and heavy compute for training & do not leverage the capabilities of existing diffusion models as in our work. 2) moving

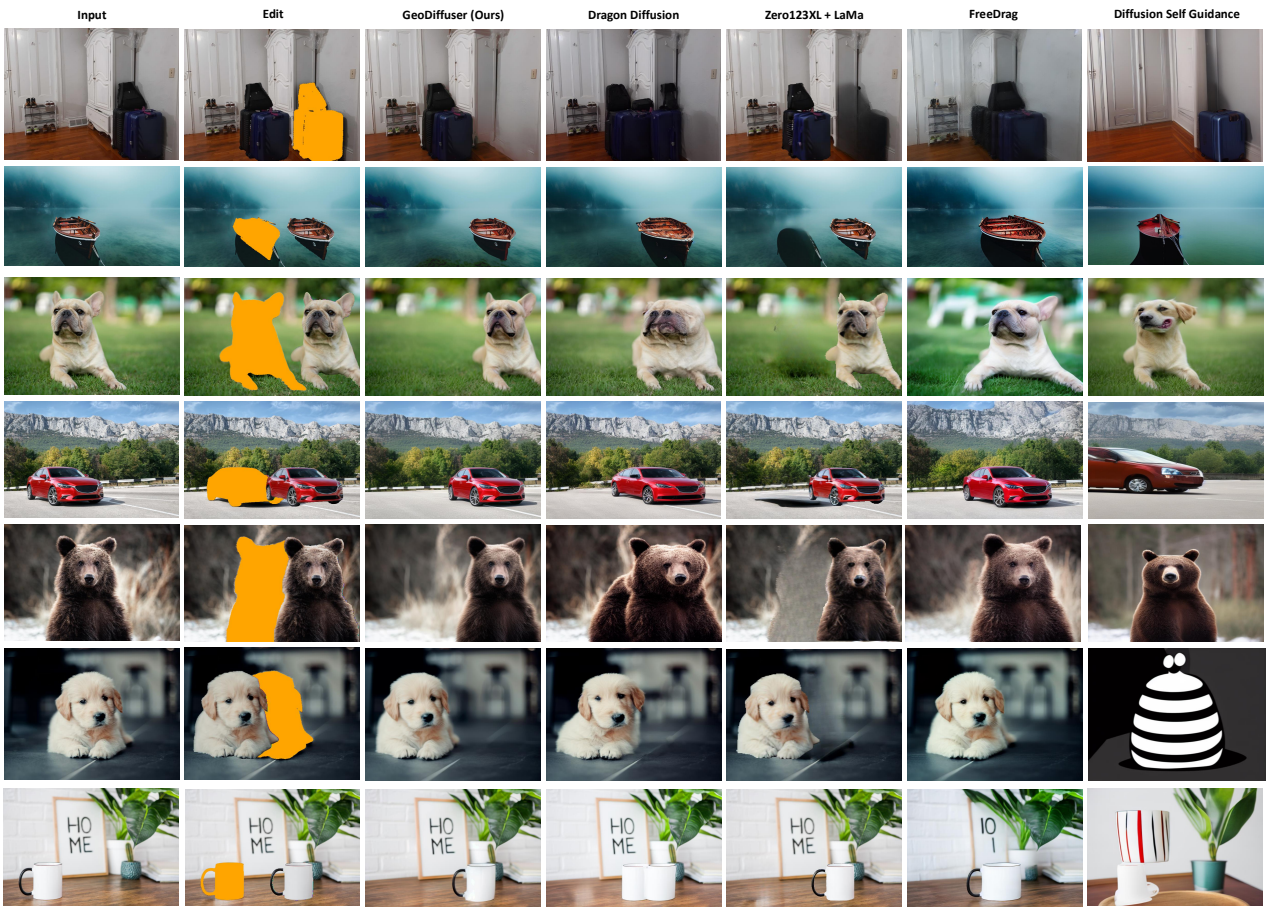

Figure 9. We perform the same edit using prior works and compare with out work. We show 2D edits here as Dragon Diffusion can not perform 3D edits. We show the intended 2D edit in column 2 where the orange mask determines the region to be inpainted and the green regions determine the region to be filled with foreground. Note that Dragon Diffusion [10] & FreeDrag [6] requires prompts along with the edit and our method does not. FreeDrag does not remove the object from the source location appropriately resulting in stretching it.

foreground most often introduces background movement as video datasets do not distinguish between foreground and background motion, 3) these methods do not bake geometry within their architecture leading to edits that may not be 3D consistent, 4) they are trained with optical flow within a bounded range and often lose object details and identity when the desired edit motion is beyond this range, and lastly 5) they do not explore having inference time optimization disabling the user to control different aspects of the edit by merely changing loss weights. We believe that the geometry attention sharing mechanism and loss functions from GeoDiffuser can help improve these models to ensure edits and generation that are consistent with geometry in future works.

O. Discussion on Slider based UI as opposed to Drag UI

We follow the slider UI of zero123 [8]. It is easy to control precise rotations as well as preserve the geometry using sliders as compared to a drag-based UI. However, we can also have a drag-based UI if the user prefers, however, this makes controlling rotations difficult.

P. User Interface

See Figures 12 and 13 that display the user interface used to perform edits using GeoDiffuser. We develop this user interface using Gradio [2]. We also submit a video along with this supplement that displays the editing process performed by a user and a website that shows gifs of edits using GeoDiffuser.

Q. Complex Shapes and Human Edits

Our method generates plausible edits for complex 3D shapes and close-up humans images (Figure 14). However, our method finds it challenging to preserve arms and legs in far shots of humans.

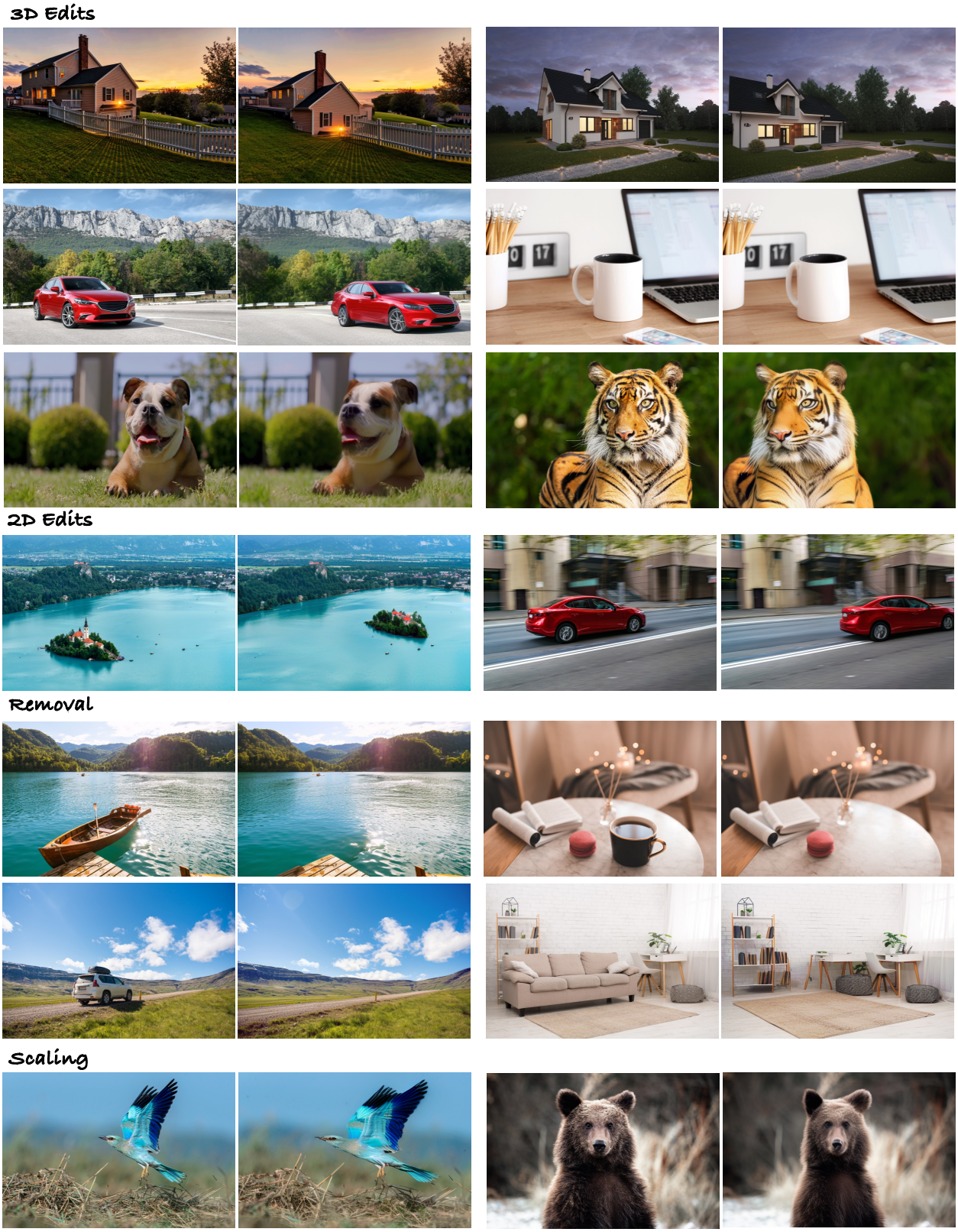

Figure 10. Qualitative results showing all variations of 2D and 3D edits performed by **GeoDiffuser** on natural images. Notice how our method not only removes/transforms objects but also the object's reflection and shadows (car, couch, boat). For 3D edits, our method produces plausible results for rotations as high as 30°. For scaling, we can perform both uniform and non-uniform scaling operations.

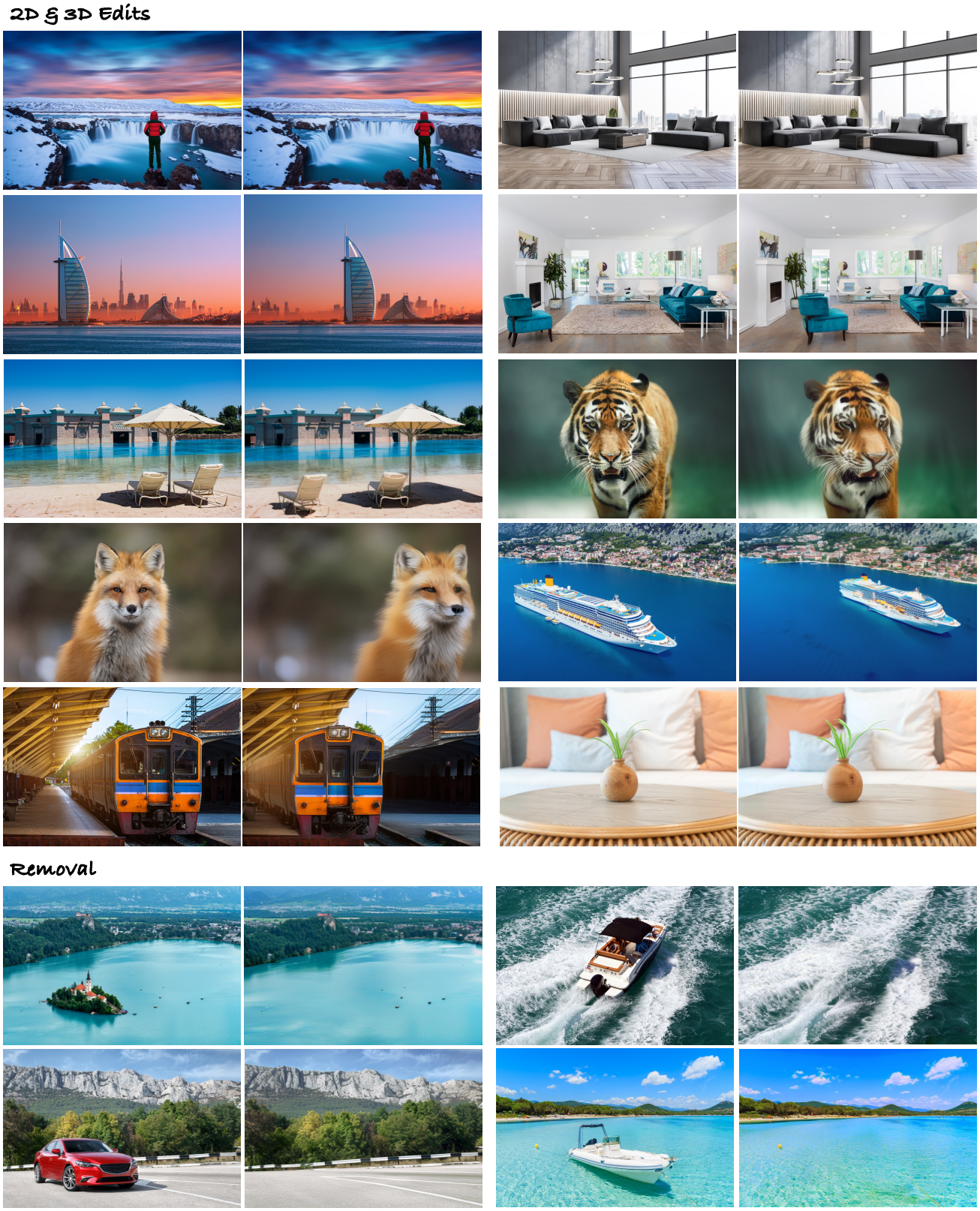

Figure 11. We display more qualitative results of our method. Each example has the input image in the left and the result of the edit in the right.

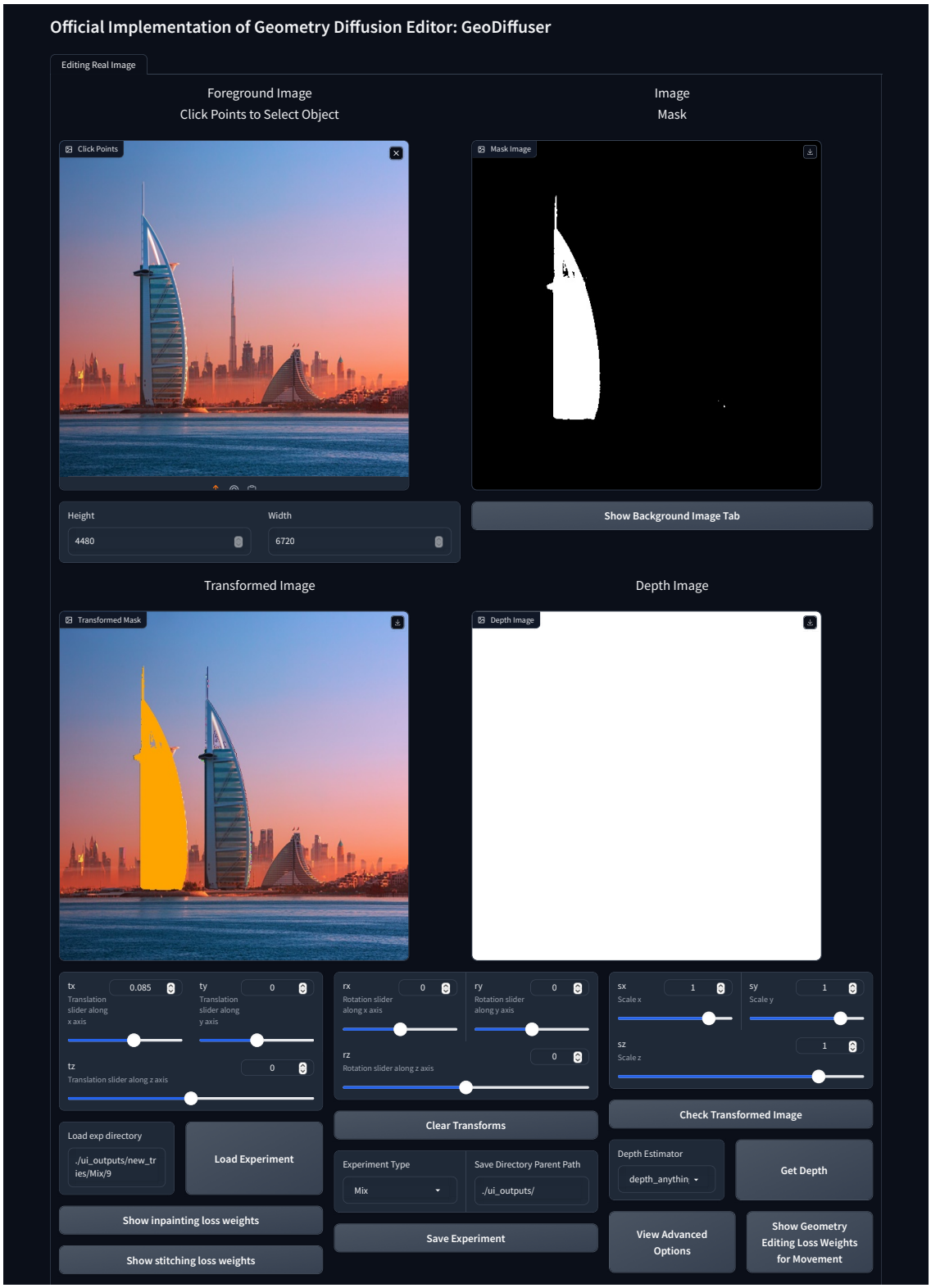

Figure 12. GeoDiffuser UI that allows users to edit images in the wild. We provide options for users to choose a monocular depth model for geometric editing. The transformed image represents the edit that the user wishes to perform. Here, the orange mask displays the region that needs to be inpainted.

Figure 13. GeoDiffuser UI also provides options for varying parameters for editing. The edited image in the bottom displays the image after the edit is complete.

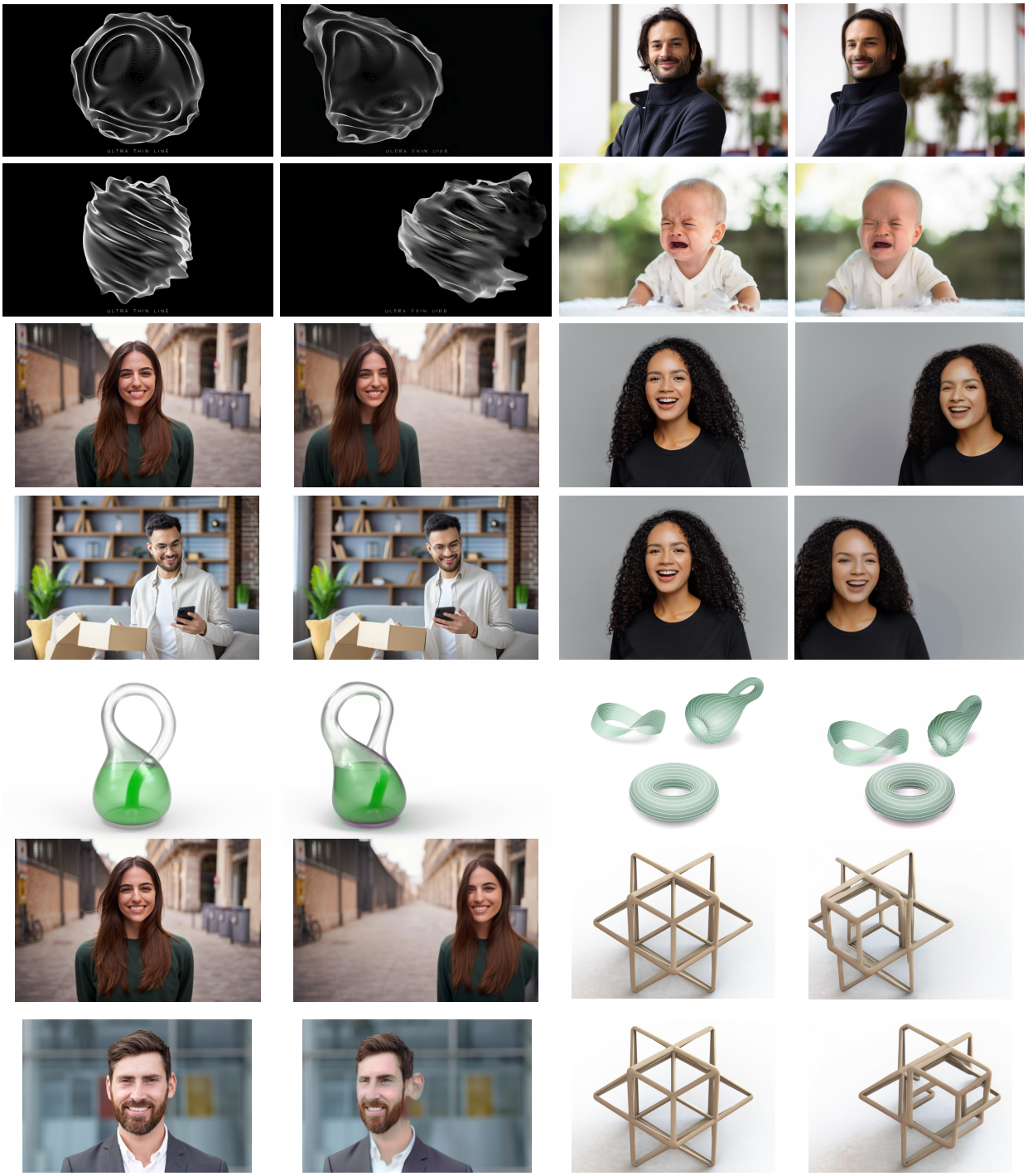

Figure 14. Editing Complex Geometries and Humans. For each row, the left shows the input image and the right shows the result of the edit. Our method provides plausible edits for most cases of complex 3D shapes and humans even when the model has not seen this. **Last row** shows some limitations of our work where the ear is interpolated because of editing at low resolution and smearing in depth maps. Our edits are limited by the base model wherein there are some cases where the face/complex shape loses detail because the model has not seen these during training. We also notice that at times the model opens eyes even when the eyes are closed in the input image because of training bias in the stable diffusion base model.

References

[1] Qualtrics. https://www.qualtrics.com. 4

[2] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019. 6

[3] Hadi Alzayer, Zhihao Xia, Xuaner Zhang, Eli Shechtman, Jia-Bin Huang, and Michael Gharbi. Magic fixup: Streamlining photo editing by watching dynamic videos. *ArXiv*, abs/2403.13044, 2024. 5

[4] Dave Epstein, Allan Jabri, Ben Poole, Alexei Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[5] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023. 2

[6] Pengyang Ling, Lin Chen, Pan Zhang, Huaian Chen, Yi Jin, and Jinjin Zheng. Freedrag: Feature dragging for reliable point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6860–6870, June 2024. 1, 6

[7] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023. 4

[8] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 2, 4, 6

[9] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004. 3

[10] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models, 2023. 2, 6

[11] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Diffeditor: Boosting accuracy and flexibility on diffusion-based image editing. *arXiv preprint arXiv:2402.02583*, 2024. 2

[12] Karran Pandey, Paul Guerrero, Matheus Gadelha, Yannick Hold-Geoffroy, Karan Singh, and Niloy J. Mitra. Diffusion handles enabling 3d edits for diffusion models by lifting activations to 3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7695–7704, June 2024. 1, 2

[13] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 5

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 3

[15] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:1623–1637, 2019. 1

[16] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 1

[17] Yujun Shi, Jun Hao Liew, Hanshu Yan, Vincent Y. F. Tan, and Jiashi Feng. Instadrag: Lightning fast and accurate drag-based image editing emerging from videos. *ArXiv*, abs/2405.13722, 2024. 5

[18] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023. 1, 3

[19] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022. 1, 2, 4

[20] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023. 3

[21] Sheng-Siang Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory. *ArXiv*, abs/2308.08089, 2023. 5

[22] Shengzhe Zhou. Diffusion self guidance implementation. https://github.com/Sainzerjj/Free-Guidance-Diffusion. 3