Localized Gaussian Splatting Editing with Contextual Awareness Supplementary Material

Hanyuan Xiao[1,2] Yingshu Chen[3] Huajian Huang[3]
Haolin Xiong[4] Jing Yang[1,2] Pratusha Prasad[1,2] Yajie Zhao[1,2,*]

[1]University of Southern California [2]Institute for Creative Technologies
[3]HKUST [4]University of California, Los Angeles

[1,2]{hxiao, jyang, bprasad, zhao}@ict.usc.edu [3]{ychengw, hhuangbg}@connect.ust.hk [4]{xiongh}@ucla.edu

Abstract

In this supplementary material, we first detail the implementation specifics in Sec. 1, including the pseudocode of our Anchor View Proposal algorithm, the training configurations, and the hardware setup and training speed. Secondly, we supplement additional qualitative results in diverse scenes in Sec. 2. Thirdly, we present a user study demonstrating the superiority of our method compared to baseline approaches (Sec. 3). Fourthly, we showcase our graphical user interface in an example generation task (Sec. 4). Lastly, we discuss on the limitations and potential future directions of our research (Sec. 5). In addition to this document, we have attached several 360-degree videos and a live demo of our GUI usage.

1. Implementation Details

1.1. Pseudocode of Anchor View Proposal Algorithm

We provide pseudocode for the proposed Anchor View Proposal (AVP) algorithm in Alg. 1. For the `addConstrast` function, we compute the minimum and maximum values in the V-channel image and map pixel values from the range [minimum, maximum] to [0, 1]. For the image plane rotation `R`, we select eight angles, each two views being 45 degrees apart in the 2D image plane.

1.2. Training Details

Resolution. In the coarse step, we gradually increase the resolution of sampled random views from 64×64 to 128×128, and then to 256×256 at iterations 0, 200, and 300, respectively. For the conditioning view, we increase the resolution from 128×128 to 256×256, and then to 512×512 at the same iteration milestones. We found that the coarse step often converges around 3500 iterations, generating relatively fine detail. During the Texture Enhancement step, the resolution of both random views and the conditioning view is set to 512×512.

Classifier-free Guidance Scale. We use a guidance scale of 3.0 in the coarse step and 100 for DI-SDS in the Texture Enhancement step.

Algorithm 1 Anchor View Proposal
Input: multi-view RGBA images I, leftIsBrighter flag f_l
Output: anchor view index $\hat{i}$

function ANCHORVIEWPROPOSAL(I, f_l)
  $V \leftarrow$ getValueChannel(RGBA2HSV(I)) ▷ Value Images
  $M \leftarrow$ getAlphaChannel(I) ▷ Mask
  $R \leftarrow [1 \dots 8] \times 45$ ▷ Image Plane Rotations
  $B \leftarrow \mathbf{0}$ ▷ Image Brightness Balance
  for all v_i, m_i **in** V, M **do**
    $S \leftarrow \mathbf{0}$
    $v_i[m_i] \leftarrow$ addContrast($v_i[m_i]$)
    for all r **in** R **do**
      $g_{i,r} \leftarrow$ rotateImage(v_i, r)
      $m_{i,r} \leftarrow$ rotateImage(m_i, r)
      $S[r] \leftarrow$ calculateBrightnessBalance($g_{i,r}$, $m_{i,r}$, f_l)
    end for
    $B[i] \leftarrow \min(S - 1)$
  end for
  $B \leftarrow$ abs(B)
  return $\hat{i} \leftarrow$ argmin(B)
end function
function CALCULATEBRIGHTNESSBALANCE(v, m, f_l)
  $h, w \leftarrow$ imageHeight(v), imageWidth(v)
  $I_l, I_r \leftarrow$ leftHalf($v[m]$), rightHalf($v[m]$)
  $b_l, b_r \leftarrow$ mean(I_l), mean(I_r) ▷ Brightness of Masked Half Pixels
  if f_l **is True then**
    return $b_l/(b_l + b_r)$
  else
    return $b_r/(b_l + b_r)$
  end if
end function

Loss Weights. In the coarse step, the weight of the RGB loss λ_{rgb} and the weight of the mask loss λ_{mask} both start

*Corresponding Author

from 0, increase to 1500 at iteration 1000, and stay at 1000 for the remaining iterations. The weight of the optimization loss by the 3D-aware diffusion prior λ_{3D-SDS} remains constant at 1. During the Texture Enhancement step, λ_{rgb} and λ_{mask} are constantly 1500, and the weight of the DI-SDS loss λ_{DI-SDS} is constantly 0.1.

Learning Rates. The learning rate for the position of Gaussian splats starts from 1×10^{-3} and decreases to 10^{-5} with exponential decay from iteration 0 to 1000 in the coarse step, and from 10^{-4} to 10^{-5} from iteration 0 to 10000 in the Texture Enhancement step. The learning rate for the scaling of Gaussian splats is $[5 \times 10^{-3}, 5 \times 10^{-4}]$ in the coarse step and $[5 \times 10^{-4}, 5 \times 10^{-5}]$ in the Texture Enhancement step, with the same milestones as the position. The rotation, color, and alpha of Gaussian splats are constantly $5 \times 10^{-3}, 10^{-2}, 10^{-2}$, respectively, in both steps.

1.3. Hardware Requirement & Speed

We train our pipeline on a single NVIDIA A40 GPU. The VRAM usage is approximately 20GB for the coarse step and 43GB for the Texture Enhancement step. Each iteration takes around 0.5 seconds for the coarse step and 7.8 seconds for the Texture Enhancement step, as per the settings detailed in Sec. 1.2. The total number of iterations depends on complexity in geometry and texture. Typically, we found the coarse step converges after 6500 iterations. While texture enhancement step highly depends on user's preference, we usually stops this step after 2000 iterations incremented from the coarse step. We sample random views with a batch size of 6 for the coarse step and 4 for the Texture Enhancement step. We discovered that increasing the batch size significantly reduces color deviation in the Texture Enhancement step.

2. More Results

We provide more qualitative results in Fig. 2, 3 and 4. In the figures, we show the anchor view images of the original scenes with 3D bounding box for editing, inpainted anchor view images, text prompts for generation, and multi-view rendering results from our pipeline. In addition to LERF [7], MipNeRF360 [2] and our self-captured datasets, we use a commercially purchased synthetic scene (the first case in Fig. 2 and the third case in Fig. 4). For the synthetic scene, we rendered RGB images in 800×800 resolution under 332 virtual camera views (48.5-degree field of view) around the scene to densely supervise 3DGS training. For our self-captured datasets, we use a Canon EOS-1D camera to capture around 400 photos of each scene and send the raw colored images to COLMAP for calibration.

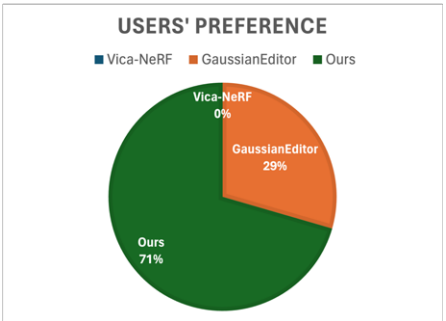

Figure 1. User Study statistics

3. User Study

To further substantiate our claim that our method generates more photorealistic results than state-of-the-art methods – GaussianEditor [3] and Vica-NeRF [5], we conducted an anonymous user study. The statistics of users' preferences are provided in Fig. 1.

We created a questionnaire featuring output examples from different methods shown in Fig.5 of the main paper. The order of the methods was randomized to mitigate potential bias. Participants were asked to choose the result that best aligned with the input text prompt and appeared the most photorealistic in each question comparing the three methods.

We also requested participants to share their age group (options were "'below 18", "18-25", "26-35", "36-45", "46+") and their familiarity with the fields of computer vision (CV) or computer graphics (CG) (options were "Yes", "No", "Maybe"). All 26 questionnaires were collected anonymously.

According to the results, no users preferred Vica-NeRF, 29.49% preferred GaussianEditor, and 70.51% preferred our method. Among the participants, 19.2% fell into the age group of 18-25, 76.9% into 26-35, and 3.8% did not provide their age. As for familiarity with CV or CG, 61.5% claimed familiarity, 26.9% claimed no familiarity, 7.7% were uncertain ("Maybe"), and 3.8% did not provide their familiarity level.

4. Graphical User Interface

We also package our pipeline into an application with graphical user interface as an extra contribution. An example of the interface can be found in Fig. 5. The system is built upon Viser [1], an open-source interactive 3D visualization library. The system supports original GSGen [4] and 3D Gaussian Splatting [6] rendering and live progress check of training. We also leave APIs easy to change in a separate configuration file. This allows users and researchers to conduct future experiments in both text-to-3D and image-to-3D generative tasks.

5. Discussion

5.1. Limitations

Our current method guarantees photorealism between the generated object and input scene, but it does not ensure physically accurate lighting. In scenes with complex lighting, a single conditioning anchor view often provides insufficient lighting information. Additionally, current text-guided generation schemes do not support shadow editing, which requires disentanglement and editing of background. Therefore, it is necessary to involve conventional inverse rendering and ray tracing similar to [8–13]. Also, our pipeline may still require manual tuning in AVP step, where the collection of renderings for anchor view proposal needs to avoid occlusion and surface intersection. When the directional illumination is cast from top to bottom, the manual set elevation angle is crucial. It will be intuitive and straightforward to extend Azimuth rotation to further sample also elevation rotation, but this will also require manual tuning.

5.2. Future Works

Our current task involves object-centered replacement and editing. This process requires the editing region to be enclosed in a bounding box, separate from the other point clouds in the scene. As a result, detailed editing that interacts with existing scene objects is not yet possible. In such cases, we need a more precise segmentation and generation method to optimize both the generated point cloud and the scene point cloud, to ensure multi-view consistency.

References

[1] viser. 2

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2

[3] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *CVPR*, 2024. 2

[4] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 2

[5] Jiahua Dong and Yu-Xiong Wang. ViCA-neRF: View-consistency-aware 3d editing of neural radiance fields. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[6] Kerbl, Bernhard and Kopanas, Georgios, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 2

[7] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023. 2

[8] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering, 2024. 3

[9] Linjie Lyu, Ayush Tewari, Thomas Leimkuehler, Marc Habermann, and Christian Theobalt. Neural radiance transfer fields for relightable novel-view synthesis with global illumination, 2022. 3

[10] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis, 2020. 3

[11] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields, 2021. 3

[12] Chong Zeng, Guojun Chen, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. Relighting neural radiance fields with shadow and highlight hints. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23, page 1–11. ACM, July 2023. 3

[13] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics*, 40(6):1–18, Dec. 2021. 3

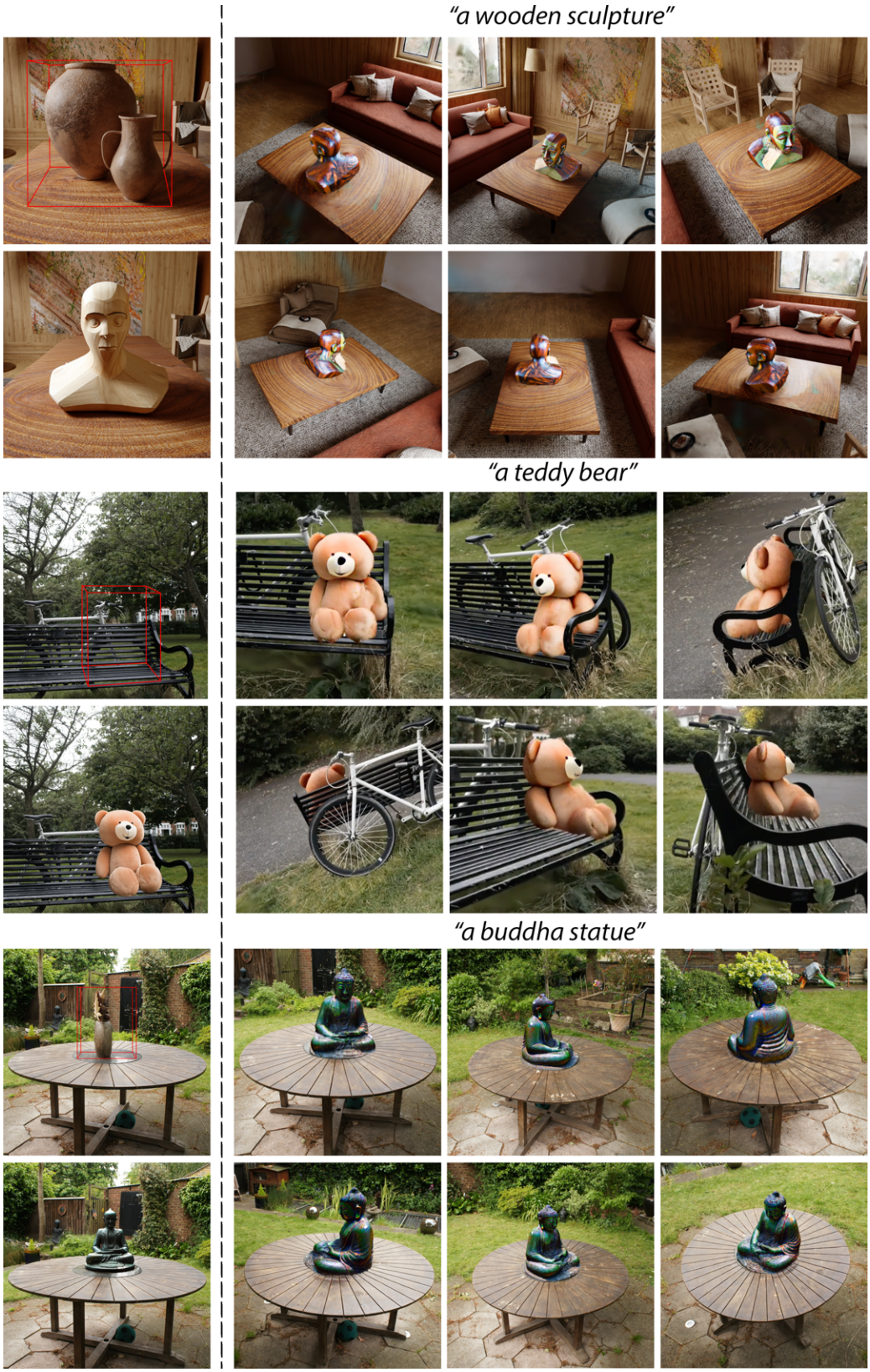

Figure 2. More Results 1 of 3. We show original and inpainted anchor view in the first column, and multi-view rendering results of our method in the right three columns.

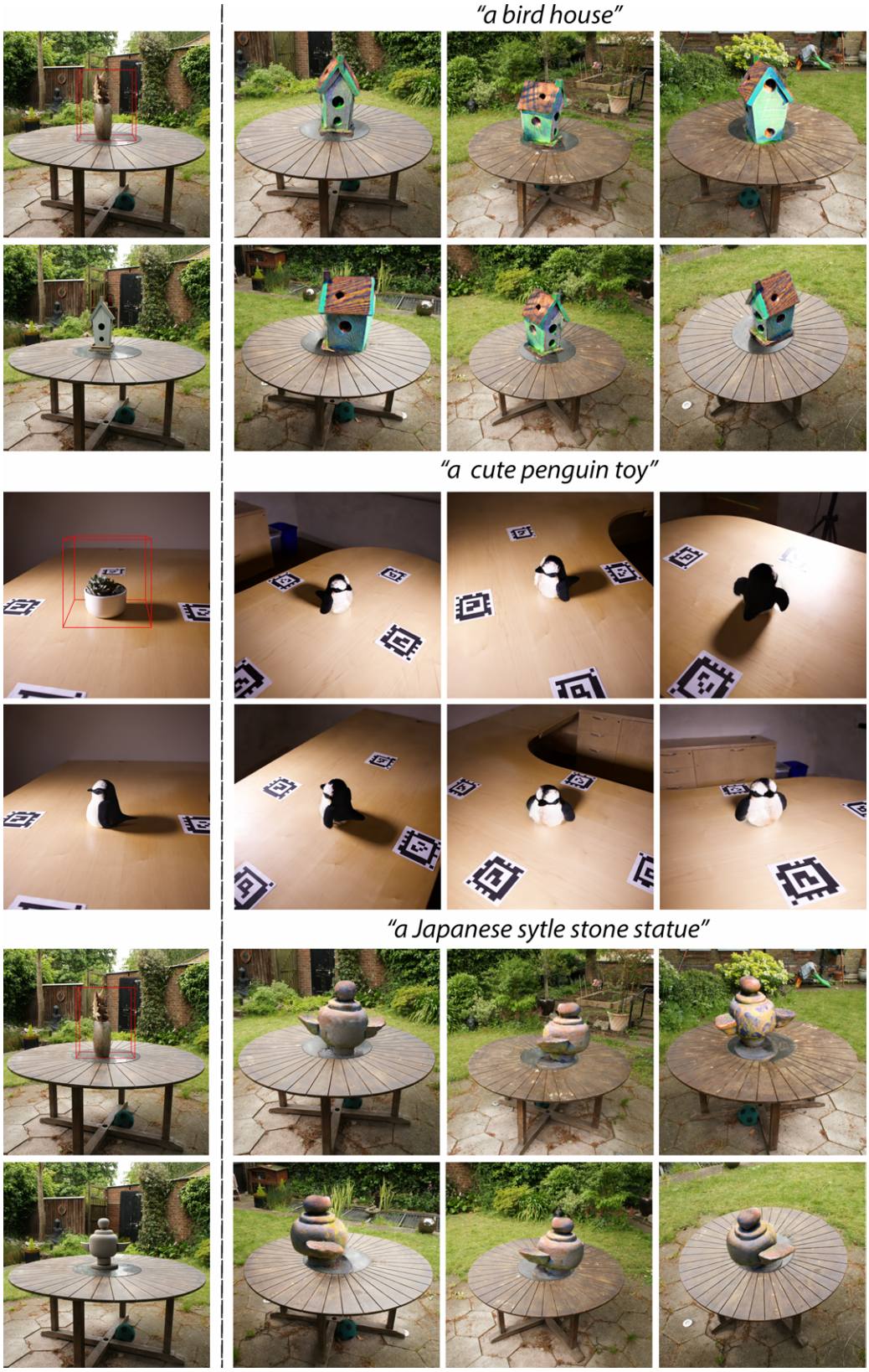

Figure 3. More Results 2 of 3. We show original and inpainted anchor view in the first column, and multi-view rendering results of our method in the right three columns.

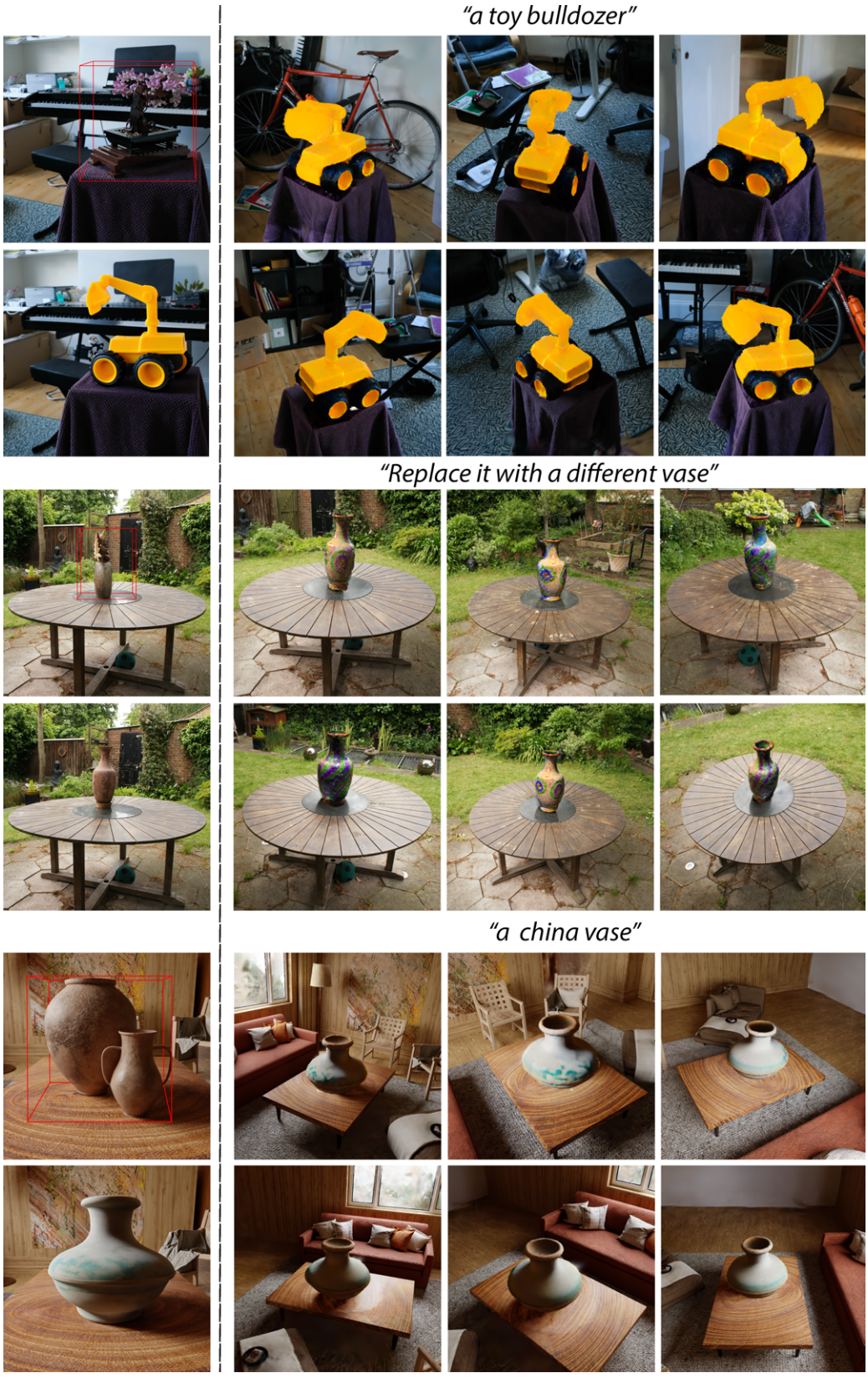

Figure 4. More Results 3 of 3. We show original and inpainted anchor view in the first column, and multi-view rendering results of our method in the right three columns.

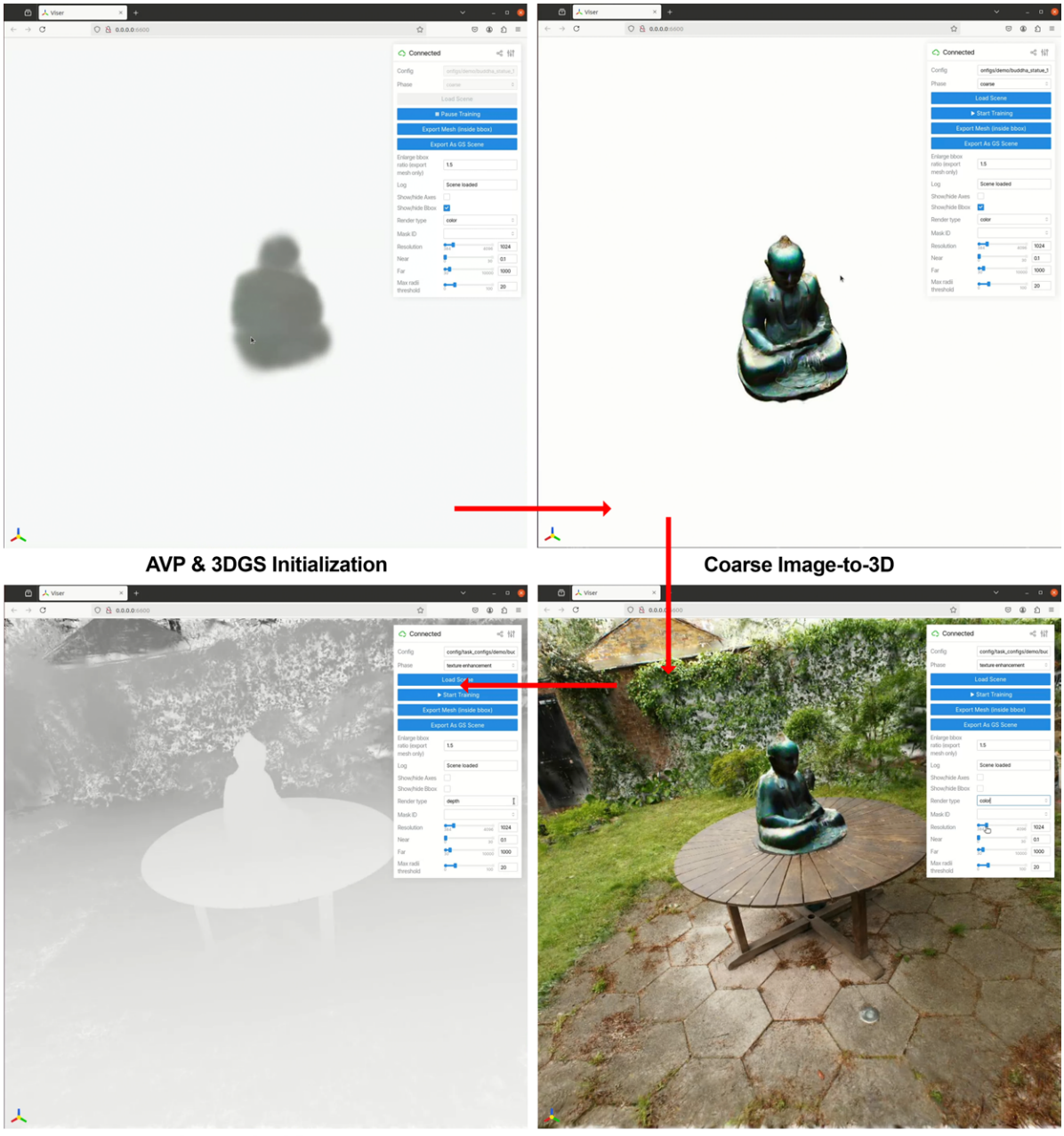

Figure 5. Graphical User Interface of our pipeline. We develop a GUI that includes Anchor View Proposal, coarse image-to-3D generation and texture enhancement steps. The interactive viewer is especially useful to visualize training procedure and results. In the lower left figure, we show rendering of depth images of the same scene besides rendering of RGBA frames.