Supplementary Material for Texture-GS

Tian-Xing Xu[1], Wenbo Hu[2†], Yu-Kun Lai[3], Ying Shan[2], and Song-Hai Zhang[1†]

[1] Tsinghua University, China
xutx21@mails.tsinghua.edu.cn,shz@tsinghua.edu.cn
[2] Tencent AI Lab, China
wbhu@tencent.com,yingsshan@tencent.com
[3] Cardiff University, United Kingdom
LaiY4@cardiff.ac.uk

1 Implementation Details

Training Details. We introduce a multi-stage training pipeline for reconstructing the geometry, UV mapping MLP, and a high-quality texture from multi-view images captured from real-world scenes. Initially, we train vanilla 3D Gaussians using a combination of the losses proposed in Sec. 4.3 to obtain an initial geometry, written as:

$$\mathcal{L}_{\mathrm{GS}} = \mathcal{L}_1 + \mathcal{L}_{\mathrm{mask}} + \lambda_{\mathrm{ssim}}\mathcal{L}_{\mathrm{ssim}} + \lambda_{01}\mathcal{L}_{01} + \lambda_{\mathrm{n}}(\mathcal{L}_{\mathrm{norm}} + \mathcal{L}_{\mathrm{sm}}). \tag{1}$$

Here, $\lambda_{\mathrm{ssim}} = 0.2$, $\lambda_{01} = 0.001$ and $\lambda_{\mathrm{n}} = 0.1$ are used to balance the loss components. Following 3D-GS [3], we initialize the parameters of 3D Gaussians with the point clouds produced by Structure-from-Motion (SfM) [5] techniques and optimize them for 30K iterations. To ensure the stability of the optimization process, we only apply the additional constraints $\mathcal{L}_{01}$, $\mathcal{L}_{\mathrm{norm}}$ and $\mathcal{L}_{\mathrm{sm}}$ after 2K iterations. We prune any semi-transparent 3D Gaussians with opacity values less than 0.5 every 6K iterations. Furthermore, we flatten each 3D Gaussian along its shortest axis by resetting the scaling value of the shortest axis as e^{-20} every 1K iterations. Subsequently, we freeze the parameters of 3D Gaussians and render depth maps for training the UV mapping MLP, utilizing the loss function $\mathcal{L}_{\mathrm{UV}}$ introduced in Sec. 4.2. Finally, we reconstruct a high-quality texture from multi-view images and finetune the parameters of 3D Gaussians using differentiable texture mapping-based splatting. Owing to the highly under-constrained nature of the parameter space for jointly optimizing the position of 3D Gaussians, the UV mapping MLP, and the texture, we freeze the UV mapping MLP and optimize the remaining parts. We supervise them with the loss function defined in Sec. 4.3, written as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{GS}} + \lambda(\mathcal{L}_1^{\mathrm{noSH}} + \lambda_{\mathrm{ssim}}\mathcal{L}_{\mathrm{ssim}}^{\mathrm{noSH}}), \tag{2}$$

where the hyperparameter $\lambda = 2$ is used to encourage most (especially view-independent) appearance information to be stored in the texture. We start by optimizing only the texture image for 10K iterations, followed by jointly learning the parameters of 3D Gaussians and the texture for another 30K iterations.

† Corresponding authors.

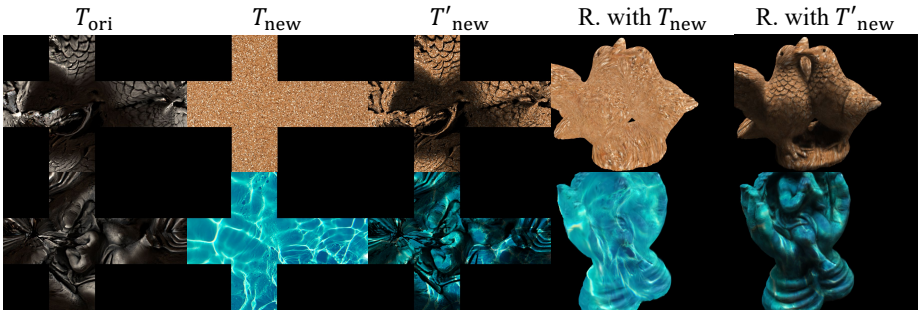


Fig. 1: Shadow-preserving texture swapping.

Network Architectures. We train a UV mapping network ϕ, which regresses a 2D UV coordinate $u \in \mathbb{R}^2$ for each 3D point $x \in \mathbb{R}^3$ either on or close to the underlying surface of the scene, and couple it with an inverse 3D-to-2D network ϕ^{-1}. We use Multi-Layer-Perceptrons (MLPs) to learn the functions ϕ and ϕ^{-1}, which consists of 4 linear layers with feature dimensions set as 128. Notably, we only employ multi-resolution hash encoding [4] on the inverse function ϕ^{-1} to ensure the local smoothness of the UV mapping ϕ, similar to NeuTex [7].

Shadow-preserving Texture Swapping. We employ a 2D texture to capture the view-independent appearance of the 3D scene from multi-view images. Notably, the appearance consists of the base color and ambient occlusion of the scene. To maintain the ambient occlusion during texture swapping, we follow NeuTex [7] and apply an ambient mask to the new texture, namely shadow-preserving texture swapping. Technically, let us denote the reconstructed texture and the new texture as $\mathcal{T}_{\text{ori}} \in \mathbb{R}^{H \times W \times 3}$ and $\mathcal{T}_{\text{new}} \in \mathbb{R}^{H \times W \times 3}$, respectively. We replace $\mathcal{T}_{\text{ori}}$ with an altered version of $\mathcal{T}_{\text{new}}$, denoted as $\mathcal{T}'_{\text{new}} \in \mathbb{R}^{H \times W \times 3}$, which is given by:

$$\mathcal{T}'_{\text{new}} = \mathcal{T}_{\text{new}} \times \text{mean}(\min(\mathcal{T}_{\text{ori}} \times 3, 1.0)), \tag{3}$$

where $\text{mean}(\cdot)$ is the mean operation along the channel axis. Fig. 1 shows the visual comparison between directly swapping with the texture image $\mathcal{T}_{\text{new}}$ and our method. Leveraging the ambient mask, our method can preserve the shadows present in the original texture, thereby achieving photo-realistic results.

2 More Analysis

Number of Gaussians. We present a visual comparison between our method and the vanilla 3D Gaussians under the same number of Gaussians to evaluate the improvement in representation power. As shown in Fig. 2, reducing the number of 3D Gaussians leads to a minor degradation in visual quality for scenes with rich textures when using our method, whereas 3D-GS [3] exhibits a significant loss of appearance details. These findings demonstrate that our

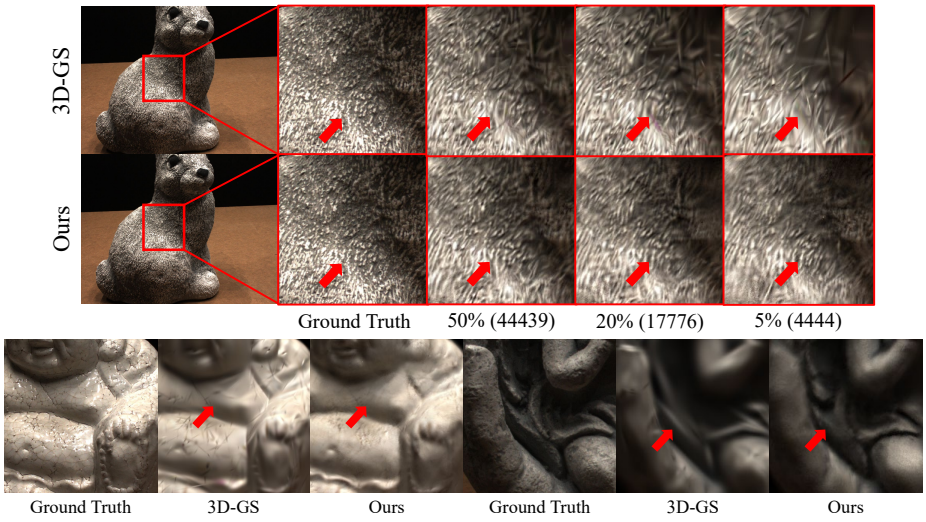


Fig. 2: Visual comparison with 3D-GS under the same number of 3D Gaussians. Please zoom in for a better view.

Table 1: Ablation studies of our method on the DTU dataset

Method	DTU			
	PSNR↑	L1↓	LPIPS↓	FPS
Ours	30.03	0.0135	0.1440	58
w.o. Pruning	30.39	0.0129	0.1310	31

proposed method significantly enhances the representational capabilities of each 3D Gaussian, ultimately improving the extensibility of 3D Gaussians to various computing platforms.

Pruning Strategy. We employ a pruning strategy and opacity regularization to remove 3D Gaussians with high transparency. Tab. 1 shows that ablating the pruning strategy (w.o. Pruning) only leads to a minor improvement in average metrics while increasing the geometry complexity and reducing the rendering speed (31 FPS v.s. 58 FPS). Fig. 3 shows the visual comparison for texture swapping. The pruning strategy can also deblur the view synthesis results for high-contrast texture swapping, such as the chessboard.

Limitations. The visual quality of appearance editing results relies heavily on the precision of UV mapping, which is dependent on the ray-Gaussian intersections described in Sec. 4.3 and the learned UV mapping MLP. The computation of the intersections depends on the normal vectors of 3D Gaussians, which are supervised by the pseudo ground truth normal maps derived from SfM [5] results under the local planarity assumption. Consequently, for objects with complex

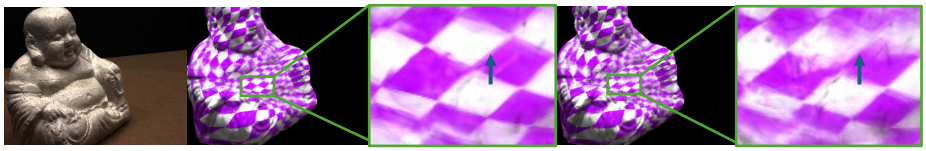
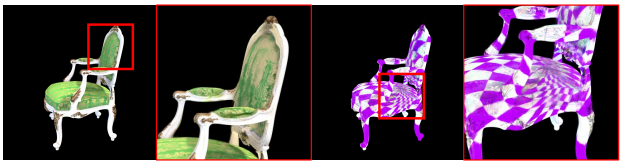


Fig. 3: Ablation study of the pruning strategy during 3D Gaussian optimization



(a) Wrong Texture Value

(b) Non-uniform UV Mapping

Fig. 4: Failure cases. Due to the limited representational power of the UV mapping MLP, our method fails to learn a uniform and reasonable texture space for objects that have thin plates or holes, thereby hindering downstream applications.

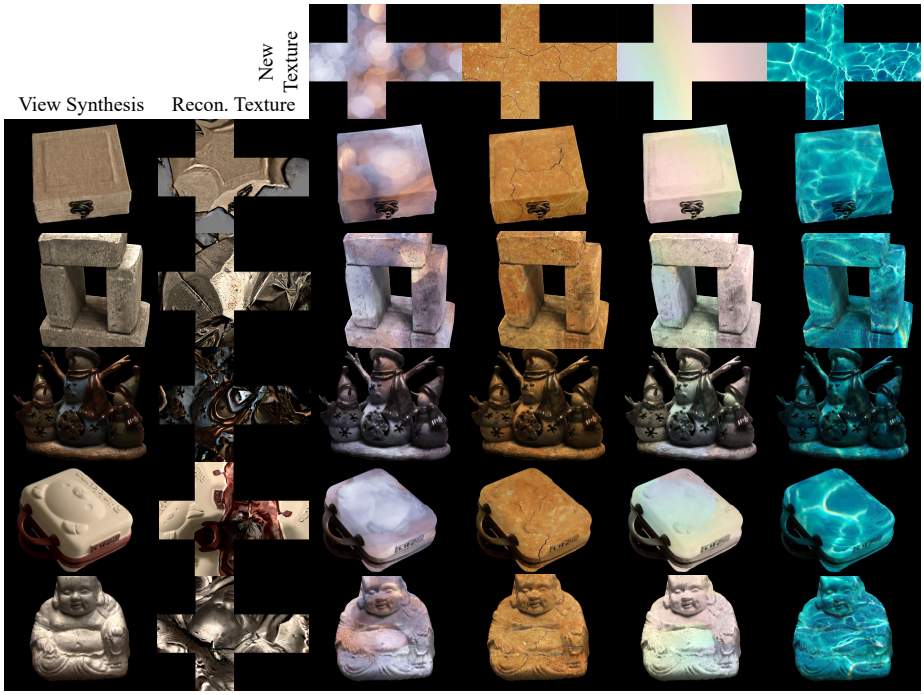
geometry that do not satisfy the assumption or the orientations of 3D Gaussians are not accurate enough (Fig. 3), our method may produce sub-optimal view synthesis results for texture swapping. Besides, we eliminate the positional encoding from the UV mapping MLP to ensure local smoothness, which inevitably restricts the representation capability of the MLP. We additionally provide a failure case from the NeRF synthetic dataset in Fig. 4a, where our method cannot differentiate the front face from the back, leading to the wrong texture values and UV mapping. Furthermore, we define the 2D UV space as a unit spherical domain, which poses challenges when dealing with 3D scenes containing multiple objects, thin plates or holes, as shown in Fig. 4b. In such cases, the UV mapping MLP struggles to generate accurate 2D coordinates, potentially hindering downstream applications. Representing the 2D UV space with multiple charts, such as Nuvo [6], offers a potential solution to address the challenges posed by complex geometries and multiple objects. However, multiple charts can also lead to discontinuities at the boundaries between charts, which may ultimately confound the global appearance editing operation such as texture swapping.

3 More Visual Results

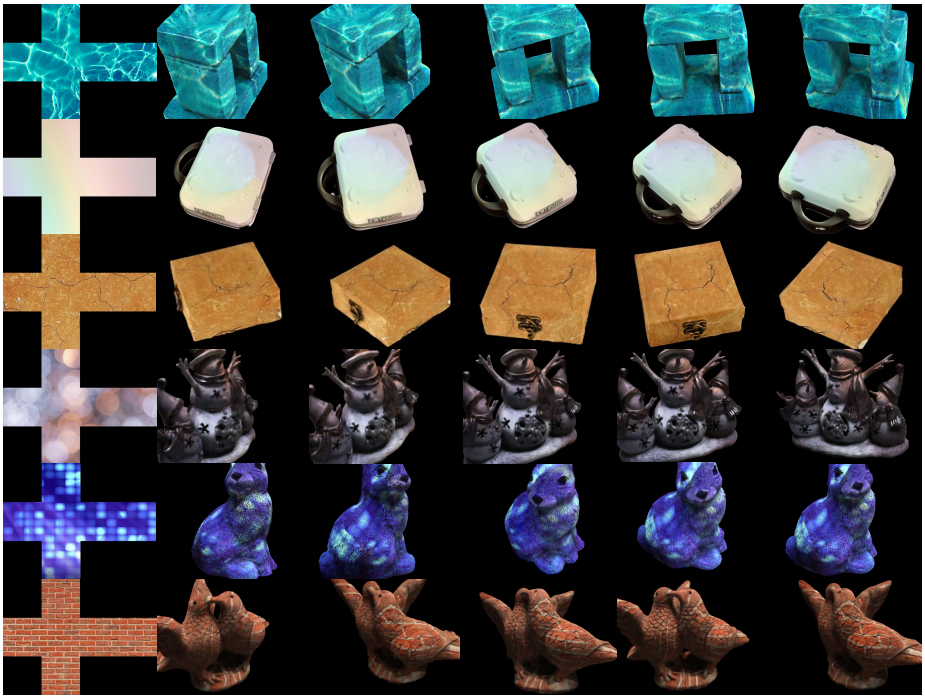
As illustrated in Fig. 5, we present additional qualitative results of texture swapping on real-world scenes from the DTU dataset [1] and the Omni3D dataset [2]. These results demonstrate the ability of our approach to generate photo-realistic images under various textures while maintaining a consistent appearance across different viewpoints.

References

1. Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision **120**, 153–168 (2016)
2. Brazil, G., Kumar, A., Straub, J., Ravi, N., Johnson, J., Gkioxari, G.: Omni3d: A large benchmark and model for 3d object detection in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13154–13164 (2023)
3. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
4. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
5. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
6. Srinivasan, P.P., Garbin, S.J., Verbin, D., Barron, J.T., Mildenhall, B.: Nuvo: Neural uv mapping for unruly 3d representations. arXiv preprint arXiv:2312.05283 (2023)
7. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7119–7128 (2021)



(a) Under various textures



(b) Under various views

Fig. 5: More visual results for texture swapping with Texture-GS.