

# Geometry-guided Online 3D Video Synthesis with Multi-View Temporal Consistency Supplementary Material

Hyunho Ha<sup>1\*</sup>   Lei Xiao<sup>2</sup>   Christian Richardt<sup>2</sup>   Thu Nguyen-Phuoc<sup>2</sup>  
 Changil Kim<sup>2</sup>   Min H. Kim<sup>1</sup>   Douglas Lanman<sup>2</sup>   Numair Khan<sup>2</sup>  
<sup>1</sup>KAIST   <sup>2</sup>Meta

## A.1. Image-based TSDF

We utilize an image-based TSDF [6] that integrates depth along each ray on the fly from  $K$  input depth maps  $\{D_k, k \in (1, \dots, N)\}$ . Specifically, for a world space point  $\mathbf{p} \in \mathbb{R}^3$  along a viewing ray, we use the known camera intrinsics  $\mathbf{K}$  and pose parameters  $\mathbf{R}, \mathbf{t}$  of each view to project the point into the  $K$  cameras as  $(x_k, y_k, z_k) = \mathbf{K}_k(\mathbf{R}_k \mathbf{p} + \mathbf{t}_k)$ . Then, we calculate the signed distance  $s_k$  as the difference between the transformed point’s  $z$ -value, and  $D_k$  sampled at the normalized pixel locations  $(u_k, v_k) = (x_k/z_k, y_k/z_k)$ :

$$s_k = z_k - D_k[u_k, v_k] \quad (1)$$

where the square brackets denote the sampling operation. We drop the superscript  $t$  indicating time for convenience.

The truncated signed-distance value  $s$  at point  $\mathbf{p}$  is then computed by fusing  $s_k$  from all input views as,

$$s = \sum_{k=1}^K \omega_k \cdot \text{clamp}(s_k, -\tau, \tau), \quad (2)$$

where  $\tau = 0.02$  m is the truncation threshold. The fusion weight  $\omega_k$  handles noise in the input depth maps and is computed as the depth variation in a  $w \times w$  pixel neighborhood  $\mathcal{N}$  around the projected location  $(u_k, v_k)$ :

$$\omega_k = \min\left(0.001 \cdot \left(\frac{\nu_k}{w \times w}\right)^{-1/2}, 1.0\right), \quad (3)$$

$$\nu_k = \sum_{(p,q) \in \mathcal{N}} \min\left((D_k[u_k, v_k] - D_k[p, q])^2, \tau^2\right). \quad (4)$$

We use  $w = 7$  for our experiments and, following Lawrence et al. [6], set  $\omega_k = 0$  if  $s_k < -\tau$ . We advance along the ray with a step size of  $0.8s$  until a surface intersection is detected. This is indicated by a change of sign in the value of  $s$ . We subsequently perform three steps of a bisection search to refine the intersection depth.

\*Work done during an internship at Meta Reality Labs.

## A.2. View and Temporally Consistent Depth

We encourage temporal consistency in the TSDF depth map  $\mathcal{D}^t$  by integrating the rendered depth  $\mathcal{D}^{t-1}$  from the previous time frame, along with the input-view depth maps  $D_k^t$ . To account for dynamic regions we use the difference masks  $M_k^t$  (Equation (1), main paper) to estimate fusion weights for  $\mathcal{D}^{t-1}$ . Specifically, we forward project all  $K$  difference masks into the novel view using Gaussian splatting [5], and apply a channel-wise maximum to estimate the difference mask  $\mathcal{M}^t$  in the novel-view. We then fuse the signed distance computed from  $\mathcal{D}^{t-1}$  into Equation (2), with the temporal fusion weight  $\omega_{\text{tmp}}^t$  defined as:

$$\omega_{\text{tmp}}^t = \min(\beta \cdot \omega_{\text{acc}}^t \cdot \max(1 - \mathcal{M}^t, 0), \eta), \text{ with} \quad (5)$$

$$\omega_{\text{acc}}^t = \omega_{\text{tmp}}^{t-1} + \sum_k \omega_k^{t-1}, \quad (6)$$

where  $\eta = 15$  is the maximum fusion weight for  $\mathcal{D}^{t-1}$ , and  $\beta$  controls the relative contribution of previous frames.

## A.3. Dense Pixel-sized 3D Gaussians

As mentioned in Section 3.1 (main paper), we analytically compute the scale, rotation, color, and opacity parameters of dense per-pixel 3D Gaussians from each input view.

This approach contrasts with previous work [12, 15] that uses a neural network to predict per-pixel Gaussian parameters. In our experiments, we observed no advantage from using a network to predict parameters for the kind of dense Gaussian point cloud that we get from projecting a depth map to 3D. One possible motivation for using a network is that it can learn to inpaint disocclusions by adjusting the scale of background Gaussians. However, we found that this ability often comes at the cost of overall reconstruction PSNR. Thus, we chose to address disocclusions using our geometry-guided blending network instead.



Figure 1. **Forward rendering a single image.** Traditional point splatting suffers from aliasing artifacts such as empty pixels, and jagged edges. Network-based models seek to inpaint disocclusions by allowing the Gaussians’ scale to vary. However, this affects overall PSNR. Our pixel-scaled Gaussians generate high-frequency details while avoiding aliasing.

Why use 3D Gaussians at all? We noticed that compared to traditional point splatting [1], Gaussians enable occlusion-handling and suffer from fewer aliasing artifacts, such as jagged edges and gaps between neighboring pixels Figure 1.

#### A.4. Network Architecture

Our blending network  $\Theta(\cdot)$  is a four-layer U-Net [11]. The input to the network consist of,

1. The  $K$  forward-rendered images  $\{\mathcal{I}_k^t\} \in \mathbb{R}^{3K \times H \times W}$ ,
2. Depth maps  $\{D_k^t\} \in \mathbb{R}^{K \times H \times W}$ ,
3. Alpha maps  $\{\alpha_k^t\} \in \mathbb{R}^{K \times H \times W}$ ,
4. The TSDF depth  $\mathcal{D}^t \in \mathbb{R}^{1 \times H \times W}$ .

Additionally, we provide camera distance, and viewing angle information. Specifically, we use

5. The dot product between each input-view ray direction, and the normals from the TSDF depth  $\mathcal{D}^t$ ,  $\in \mathbb{R}^{K \times H \times W}$
6. The dot product between each input-view ray direction, and the target view direction,  $\in \mathbb{R}^{K \times H \times W}$ .
7. The distance between the input and target cameras, repeated spatially to get a map  $\in \mathbb{R}^{K \times H \times W}$ .
8. The dot product between the input and target viewing directions, repeated spatially to get a map  $\in \mathbb{R}^{K \times H \times W}$ .

In summary, the input to  $\Theta(\cdot)$  is a tensor  $\in \mathbb{R}^{(9K+1) \times H \times W}$

#### A.5. Training Procedure

We use the ScanNet [3], DyNeRF [7], and Google Spaces dataset [4] to train our network. ScanNet provides a dense views of general indoor scenes captured with a depth sensor. We use three scenes from DyNeRF to account for large-baseline stereo scenarios, and utilize the remaining scenes for testing. The Google Spaces dataset is used to provide training in small-baseline stereo settings. For the ScanNet dataset, we first select a novel time frame and then choose corresponding input frames from a range of  $\pm 30$  frames, excluding frames  $[-4, 4]$  to avoid selecting the closest frames to the novel view. In contrast, for the DyNeRF and Spaces datasets, we manually curate stereo pairs and generate depth

maps for each view using RAFT-Stereo [10]. We then randomly choose input and novel views from at a fixed time frame for training.

#### A.6. Testing Datasets

The DyNeRF dataset [7] consists of 18–22 cameras with a resolution of  $2704 \times 2028$  pixels, and features well-synchronized indoor scenes. We use two scenes containing 22 cameras as the test set. We test all methods at half the original resolution ( $1352 \times 1014$  pixels).

The ENeRF-outdoor dataset [8] consists of 18 cameras capturing multiple actors in an outdoor setting. Each camera has a resolution of  $1920 \times 1080$  pixels. Again, we use half the original resolution ( $960 \times 540$ ) for testing all methods. Furthermore, we found that this dataset has imperfect camera synchronization, and suffers from color calibration errors. To mitigate the impact of these on quantitative metrics, we further downscale the rendered images to  $480 \times 270$  for the quantitative evaluation in Tables 1 and 2 of the main paper.

The D3DMV dataset [9] includes 10 cameras capturing outdoor scenes. We use the compressed version of the dataset with a resolution of  $640 \times 360$  pixels.

#### A.7. Evaluation Procedure

For all three datasets, we use the provided camera parameters. We evaluate the metrics for the first 100 frames. We select a subset of cameras as the test views. We use nine test views for DyNeRF, three test views for ENeRF, and four test views for the D3DMV dataset. We select the  $K$  input cameras closest to the test view – excluding the test view itself – as inputs. We use  $K = 4$  for DyNeRF and ENeRF-outdoor, and  $K = 2$  for D3DMV in accordance with existing online multi-view methods [6, 8, 9, 12, 15].

#### A.8. Additional Results

**Background image blending.** Figure 2 shows the impact of background image blending in our network, allowing it to inpaint disocclusion holes in forward-warped images.

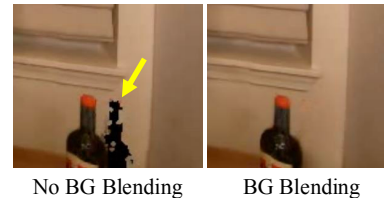


Figure 2. **Impact of background blending.** Our background image blending successfully inpaints the empty regions in the forward-warped images caused by occluded areas in the input views.

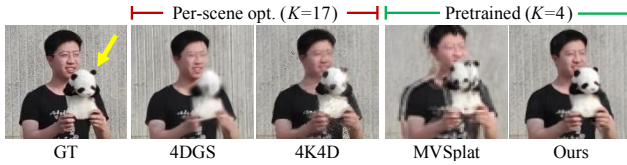


Figure 3. **Qualitative comparison on the ENeRF-Outdoor [8] dataset.** The baseline methods produce blurry results despite per-scene optimization with dense views (4DGS, 4K4D) or a pretrained network with sparse views (MVSplat). Our method reconstructs fine details, including both moving humans and the background.

**Additional comparisons.** We additionally compare our method with offline dynamic scene reconstruction methods (4DGS [13] and 4K4D [14]), and sparse multi-view reconstruction method (MVSplat [2]) in Figure 3. Even though 4K4D and 4DGS optimize per scene and use more views ( $K=17$ ), our method shows better and sharper results.

## References

- [1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 2
- [2] Yuedong Chen, Haoqi Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian splatting from sparse multi-view images. In *ECCV*, 2024. 3
- [3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 2
- [4] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *CVPR*, 2019. 2
- [5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1
- [6] Jason Lawrence, Dan Goldman, Supreeth Achar, Gregory Major Blascovich, Joseph G. Desloge, Tommy Fortes, Eric M. Gomez, Sascha Häberling, Hugues Hoppe, Andy Huibers, Claude Knaus, Brian Kuschak, Ricardo Martin-Brualla, Harris Nover, Andrew Ian Russell, Steven M. Seitz, and Kevin Tong. Project Starline: a high-fidelity telepresence system. *ACM Trans. Graph.*, 40(6):1–16, 2021. 1, 2
- [7] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and ZhaoYang Lv. Neural 3D video synthesis from multi-view video. In *CVPR*, 2022. 2
- [8] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia*, 2022. 2, 3
- [9] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3D mask volume for view synthesis of dynamic scenes. In *ICCV*, 2021. 2
- [10] Lahav Lipson, Zachary Teed, and Jia Deng. RAFT-stereo: Multilevel recurrent field transforms for stereo matching. In *3DV*, 2021. 2
- [11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2
- [12] Hanzhang Tu, Ruizhi Shao, Xue Dong, Shunyuan Zheng, Hao Zhang, Lili Chen, Meili Wang, Wenyu Li, Siyan Ma, Shengping Zhang, et al. Tele-Aloha: A telepresence system with low-budget and high-authenticity using sparse RGB cameras. In *SIGGRAPH*, 2024. 1, 2
- [13] GuanJun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *CVPR*, 2024. 3
- [14] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4K4D: Real-time 4D view synthesis at 4K resolution. In *CVPR*, 2024. 3
- [15] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. GPS-Gaussian: Generalizable pixel-wise 3D Gaussian splatting for real-time human novel view synthesis. In *CVPR*, 2024. 1, 2