# EVolSplat: Efficient Volume-based Gaussian Splatting for Urban View Synthesis

## Supplementary Material

## 1. Analysis of the convergence of $\Delta\mu_i$

In this section, we will show the position offset $\Delta\mu_i$ converge to a fixed value at the infinite training iterations. Recall that the $\Delta\mu_i$ in the iteration $k$ can be formulated as:

$$\mathbf{f}_i = \mathbf{F}(\mu_i^{init} + \Delta\mu_i^{k-1}) \tag{1}$$

$$\Delta\mu_i^k = \mathit{Tanh}(\mathcal{D}_{pos}(\mathbf{f}_i)) \cdot v_{\text{size}} \tag{2}$$

We can derive the $\Delta\mu_i^k$ as:

$$\Delta\mu_i^k = \mathit{Tanh}(\mathcal{D}_{pos}(\mathbf{F}(\mu_i^{init} + \Delta\mu_i^{k-1}))) \cdot v_{\text{size}} \tag{3}$$

We define $\mathcal{H} \triangleq \mathit{Tanh}(\mathcal{D}_{pos}(\mathbf{F}(\cdot))) \cdot v_{\text{size}}$ for simplicity, i.e.,

$$\Delta\mu_i^k = \mathcal{H}(\mu_i^{init} + \Delta\mu_i^{k-1}) \tag{4}$$

Given that $\Delta\mu_i$ small, we apply the first-order Taylor expansion to approximate Eq. 4:

$$\Delta\mu_i^k = \mathcal{H}(\mu_i^{init}) + \cdot \left.\frac{\partial\mathcal{H}}{\partial\mu}\right|_{\mu_i^{init}} \Delta\mu_i^{k-1} \tag{5}$$

Considering that $\boldsymbol{\beta} \triangleq \mathcal{H}(\mu_i^{init})$ and $\boldsymbol{\Gamma} \triangleq \left.\frac{\partial\mathcal{H}}{\partial\mu}\right|_{\mu^{init}}$ are both constants, we reformulate Eq. 5 as:

$$\Delta\mu_i^k = \boldsymbol{\beta} + \boldsymbol{\Gamma}\Delta\mu_i^{k-1} \tag{6}$$

The training converges in our experimental observations. Therefore, we assume $\Delta\mu_i^\infty$ converges as $k$ approaches the infinite step, i.e., $k \to \infty$. This allows us to derive $\Delta\mu_i^\infty$:

$$\Delta\mu_i^\infty = \boldsymbol{\beta} + \boldsymbol{\Gamma}\Delta\mu_i^\infty \tag{7}$$

$$= (\boldsymbol{I} - \boldsymbol{\Gamma})^{-1}\boldsymbol{\beta} \tag{8}$$

This indicates that $\mu_i^\infty$ converges to a constant value. In supplementary Sec. 4.2, we show that $\mu_i^t$ converges quickly at early steps during inference.

## 2. Implementation Details

### 2.1. Remove Depth Outliers

We begin by applying a depth consistency check to filter out noisy depth data. Specifically, we unproject the depth map $D_i$ of $i$th frame to 3D and reproject it to a nearby view $j$, obtaining a projected depth $D_{i\to j}$. Next, we compare $D_{i\to j}$ and $D_j$ and filter out depths where the absolute relative error exceeds an empirical threshold $\sigma = 0.2m$. We formulate this process as:

$$M_d = |D_{i\to j} - D_j| < \sigma, D_{i\to j} = D_j\left(\pi_j\pi_i^{-1}(\mathbf{u}_i)\right) \tag{9}$$

where $\mathbf{u}_i$ denotes pixel coordinates of $i$th frame and $M_d$ represents the geometric consistency mask. We further utilize the 3D statistical filter in Open3D [11] to remove the clear floaters in the unprojected point clouds for each frame. In our implementation, we set the number of neighbors to 20 and the standard deviation ratio to 2.0.

### 2.2. Foreground Gaussian Details

Following [4], $\boldsymbol{\Sigma}$ is decomposed into two learnable components: rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$ to holds practical physical significance, see the following formula:

$$\boldsymbol{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T \tag{10}$$

To allow independent optimization of both factors, we use a 3D vector $s$ representing scaling and a quaternion $q$ for rotation separately. Instead of directly learning scales $s$, we initialize the $s_{\text{init}}$ with the average distance of $K$ nearest neighbors using the KNN algorithm and learn the scales residual $\Delta s$ from the volume latent feature $\mathbf{f}_i$ via a decoder $\mathcal{D}_{cov}$. We experimentally observe that the residual learning strategy helps our network converge faster and enhances the model capacity.

$$\mathbf{s} = \mathbf{s}_{\text{init}} + \Delta\mathbf{s}, \quad \Delta\mathbf{s} = \mathcal{D}_{cov}(\mathbf{f}_i) \tag{11}$$

### 2.3. Hemisphere Background Details

We also develop the generalizable hemisphere model for the background which typically lies hundreds of meters away from the vehicle, as discussed in the main paper. Specifically, we initialize the background as a hemisphere with a fixed radius $r_{bg} = 100m$ as a hyperparameter, with its center positioned at the midpoint of the foreground volume. This background hemisphere moves along with the vehicle such that the relative distance is fixed wrt. the target camera. We project the points onto $K$ reference images to retrieve their 2D color $\{\mathbf{c}_k\}_{k=1}^K$ to regress all gaussian parameters through network $\mathcal{M}_{bg}$ as mentioned in the main paper. Similar to the foreground framework, we initialize the Gaussian scales using the KNN algorithm and learn their scale residuals $\Delta\mathbf{s}_{bg}$ for each Gaussian.

$$\mathbf{s}_{bg} = \Delta\mathbf{s}_{bg} + \mathbf{s}_{bg}^{init}, \quad \Delta\mathbf{s}_{bg} = \mathcal{M}_{bg}\left(\{\mathbf{c}_k\}_{k=1}^K\right) \tag{12}$$

For opacity and rotation, we explicitly set the opacity to 1 and the rotation to a unit quaternion $[1, 0, 0, 0]$ as a reasonable initialization, without involving them in the network optimization.

### 2.4. SparseCNN Network Architecture

We build a generalizable efficient 3DCNN $\psi^{3D}$ to provide the geometric priors for the foreground contents. Given the global point cloud $\mathcal{P} \in \mathbb{R}^{N_p \times 3}$, we quantize the point cloud with the voxel size $v_{\text{size}} = 0.1m$ and fed these sparse tensors into the $\psi^{3D}$ to predict the latent feature volume$\mathbf{F}$. The sparse 3DCNN uses a U-Net like architecture with skip connections, comprising some convolution and transposed convolution layers. The details of 3DCNN are listed in the Tab. 1. We use torchsparse as the implementation of $\psi^{3D}$.

| SparseCNN Network Architecture | | |
|---|---|---|
| **Layer** | **Description** | **In/Out Ch.** |
| $\text{Conv}_{i=0}$ | kernel = $3 \times 3 \times 3$, stride = 1 | 3/16 |
| $\text{Conv}_{i=1}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 16/16 |
| $\text{Conv}_{i=2}$ | kernel = $3 \times 3 \times 3$, stride = 1 | 16/16 |
| $\text{Conv}_{i=3}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 16/32 |
| $\text{Conv}_{i=4}$ | kernel = $3 \times 3 \times 3$, stride = 1 | 32/32 |
| $\text{Conv}_{i=5}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 32/64 |
| $\text{Conv}_{i=6}$ | kernel = $3 \times 3 \times 3$, stride = 1 | 64/64 |
| $\text{DeConv}_{i=7}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 64/32 |
| $\text{DeConv}_{i=8}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 32/16 |
| $\text{DeConv}_{i=9}$ | kernel = $3 \times 3 \times 3$, stride = 2 | 16/16 |

Table 1. **Architecture of SparseConvNet**. Each layer consists of sparse convolution, batch normalization, and ReLU.

## 3. Baselines

In this section, we discuss the state-of-the-art baselines used for comparison with our approach.

**Feed-Forward NeRFs:** We adopt the official implementations of MVSNeRF [2], MuRF [9] and EDUS [6]. For each method, we retrain the model using 160 sequences from KITTI-360 [5] under a 50% drop rate. We select the three nearest training frames of the target view as reference images for these methods. MVSNeRF and MuRF utilize multi-view stereo (MVS) algorithms to construct the cost volume and apply 3DCNN to reconstruct a neural field while EDUS leverages the depth priors to learn a generalizable scene representation.

**Feed-Forward 3DGS:** We adopt the official implementations of PixelSplat [1] and MVSplat [3]. We find that using three reference images caused color shifts at novel viewpoints in urban scenes, so we use the two nearest training frames to achieve optimal performance following the original paper. PixelSplat predicts 3D Gaussians with a two-view epipolar transformer and then spawns per-pixel Gaussians. MVSplat exploits multi-view correspondence information for geometry learning and predicts 3D Gaussians from image features. Both methods are trained on a single Nvidia RTX V100 using the full resolution of KITTI-360.

Additionally, as we illustrated in the teaser figure in the main paper, these pixel-align 3DGS methods predict inconsistent 3DGS when accumulating multiple local volumes. Note that to ensure a fair comparison, we conduct experiments using a single local volume following their default setting(use 2 reference images), as reported in Table 1 and Figure 4 in the main paper.

**Test-Time Optimization Methods:** To evaluate our fine-tuning results, we compare them against recent test-time optimization methods under the 50% drop rate. Specifically, we use the latest version of Nerfacto [8] provided by Nerfstudio and the official codebase of 3DGS [4]. Nerfacto is a combination of many published methods that demonstrate strong performance on real-world data, including pose refinement, appearance embedding, scene contraction, and hash encoding. For 3DGS, we initialize the 3DGS model with our global point cloud to ensure a fair comparison.

## 4. Additional Experimental Results

### 4.1. Monocular Depth Modularity

To further evaluate the sensitivity of our method to different depth estimation approaches, we conduct experiments using two distinct metric depth estimators: Metric3D [10] and UniDepth [7]. Specifically, we pretrain our model with Metric3D, and evaluate using depth maps of two different models for feed-forward inference on novel scenes. As shown in Tab. 2, our EVolSplat consistently outperforms the baseline methods on both depth estimators, demonstrating its robustness in handling depth predictions across varying distributions.

| Depth Method | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| Metric3D [10] | 24.43 | 0.786 | 0.202 |
| UniDepth [7] | 23.38 | 0.775 | 0.223 |

Table 2. **Depth Sensitivity Experiment** The results are averaged on five testsets from the Waymo dataset.

### 4.2. Additional Ablation for $\Delta\mu_i$

As mentioned in the main paper, $\Delta\mu_i$ depends on the previous estimation $\Delta\mu_i^{prev}$, but it stabilizes at a stationary point after infinite training iterations. Note that $\mathcal{D}_{pos}$ is designed to continually decode the offset $\Delta$ wrt. $\mu_{init}$, even at the ideal location, avoiding an infinite loop caused by toggling between the ideal offset and zero. We further conduct experiments by recursively updating the offsets ($i = 0, 1, 2, 3$) during inference to verify its convergence. As reported in Tab. 4, our pretrained model successfully refines the noisy primitive's position after the first inference (increasing PSNR by approximately 0.46 dB) and maintains a stable value of 23.78dB even with additional updates.

| Method | Waymo | | | KITTI-360 | | |
|---|---|---|---|---|---|---|
| | PSNR(dB)↑ | SSIM↑ | LPIPS↓ | PSNR(dB)↓ | SSIM↑ | LPIPS↓ |
| MuRF [9] | 23.66 | 0.746 | 0.256 | 19.83 | 0.669 | 0.340 |
| EDUS [6] | 23.41 | 0.769 | **0.147** | 20.13 | 0.659 | 0.257 |
| MVSplat [3] | 24.08 | 0.758 | 0.197 | 17.80 | 0.581 | 0.361 |
| Ours | **25.06** | **0.820** | 0.189 | **21.23** | **0.738** | **0.222** |

Table 3. **Quantitative results on Waymo and KITTI-360 datasets** with other generalizable methods. All models are trained on the Waymo dataset using drop50% sparsity level. Metrics are averaged on five validation scenes without any finetuning.

| | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| $i = 0$ | 23.329 | 0.794 | 0.175 |
| $i = 1$ | 23.787 | 0.819 | 0.171 |
| $i = 2$ | 23.778 | 0.819 | 0.171 |
| $i = 3$ | 23.786 | 0.819 | 0.171 |

Table 4. **Ablation Study** on recursion of $\Delta\mu_i$

| Layers | Width | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|
| 2 | 64 | 22.61 | 0.780 | 0.192 |
| 3 | 128 | 22.69 | 0.785 | 0.189 |
| 4 | 128 | 22.60 | 0.785 | 0.191 |

Table 5. **Ablation study** of the background MLP capacity.

### 4.3. Training on Waymo

To verify our method's performance given different training sequences, we train our method on the Waymo dataset and evaluate its feed-forward performance on Waymo and KITTI-360, as shown in the Tab. 3. Our method consistently achieves state-of-the-art performance in terms of PSNR and SSIM metrics, indicating its robustness for different driving data distributions.

### 4.4. Ablation Experiments for Background MLP

Our background model primarily blends colors from nearby reference images rather than learning textures from scratch. As shown in Tab. 5, increasing the layers and width of the background MLP does not yield significant improvements. A light-weight two-layer MLP provides sufficient capacity to reconstruct backgrounds while minimizing computational overhead.

## 5. More Qualitative Results

### 5.1. More Qualitative Results in Ablation Study

Removing offset refinement and occlusion check leads to visible artifacts in small regions, such as the car boundary in Fig. 7(b) in the main paper and the car light in Fig. 1. While these components don't significantly affect overall quantitative results, they improve local visual quality. Similarly, the color projection window, which compensates for inaccurate Gaussian positions, also improves local visual quality, see Fig. 2.


Figure 1. Qualitative Results of offset refine strategy


Figure 2. Qualitative Results of windows size strategy

### 5.2. More Feed-Forward Inference Qualitative Results

Our method enables efficient reconstruction and real-time photorealistic NVS from flexible sparse street view images. We provide more qualitative results on the KITTI-360 dataset via a feed-forward inference under drop50% setting, as shown in Fig. 4.

## 6. Limitions

We present some failure cases in Fig. 3. A key limitation of the proposed approach is its dependence on the metric depth estimation. Our method suffers degeneration when the depth model struggles to provide fine-grained depth estimates for thin structures.

Another limitation is that our generalizable hemisphere background only roughly approximates the geometry of distant landscapes, leading to artifacts on background region. However, high-quality rendering of the foreground is generally more critical for autonomous driving applications. A potential solution may be to leverage image-based rendering techniques to model the background, though this would reduce rendering efficiency.

## References

[1] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image

Figure 3. **Limitations in thin structure and background**.

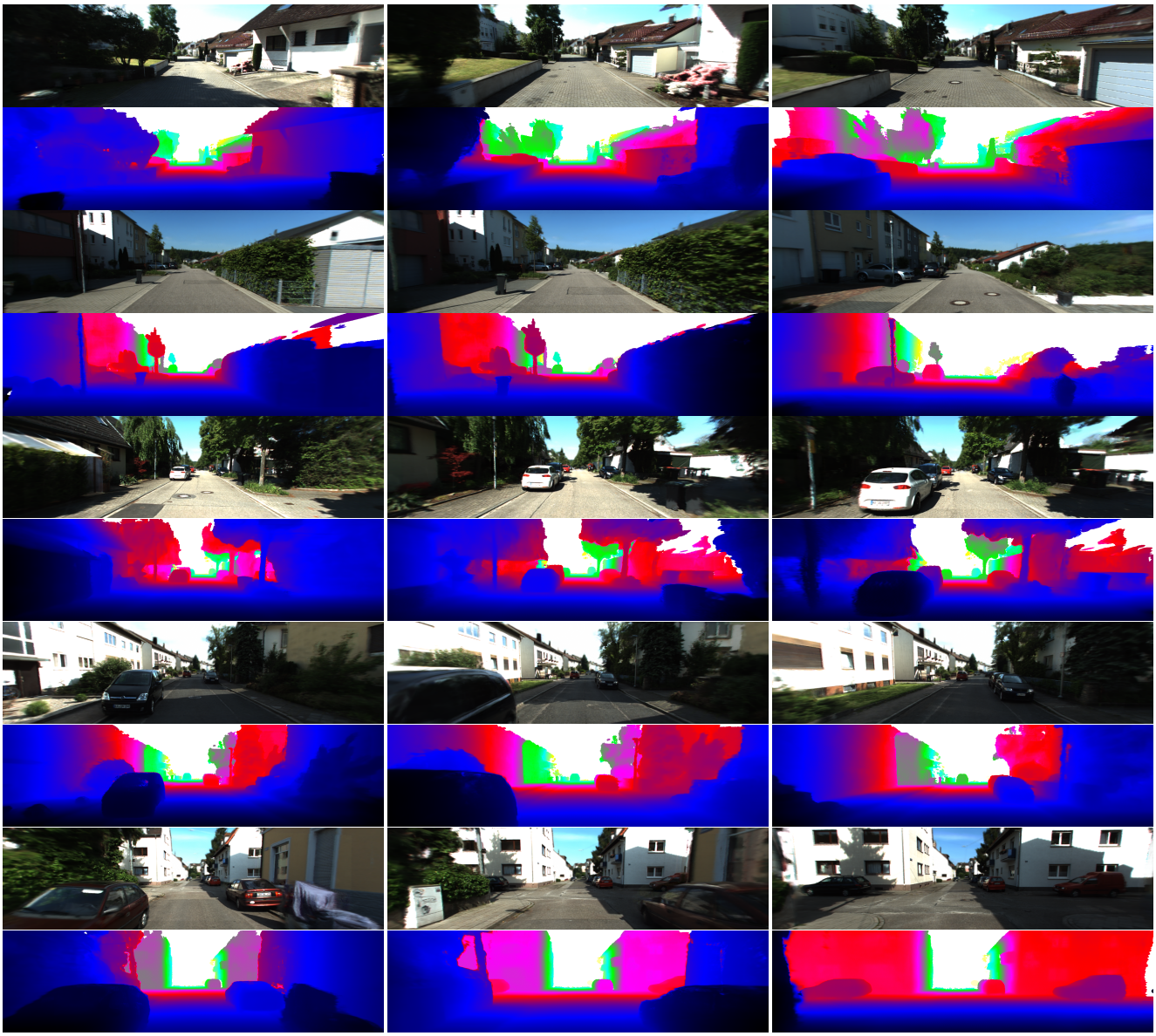

Figure 4. **More Qualitative Results on KITTI-360**. We visualize synthesized images (odd rows) and the corresponding proxy geometry (even rows) on novel scenes generated by our pretrained model through a feed-forward inference.

pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19467, 2024. 2

[2] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 2

[3] Yuedong Chen, Haofei 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. *arXiv preprint arXiv:2403.14627*, 2024. 2, 3

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2

[5] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 2

[6] Sheng Miao, Jiaxin Huang, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Andreas Geiger, and Yiyi Liao. Efficient depth-guided urban view synthesis. *arXiv preprint arXiv:2407.12395*, 2024. 2, 3

[7] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024. 2

[8] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 2

[9] Haofei Xu, Anpei Chen, Yuedong Chen, Christos Sakaridis, Yulun Zhang, Marc Pollefeys, Andreas Geiger, and Fisher Yu. Murf: Multi-baseline radiance fields. *arXiv preprint arXiv:2312.04565*, 2023. 2, 3

[10] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023. 2

[11] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 1