G3R: Gradient Guided Generalizable Reconstruction for Large Scenes

Yun Chen[1,2⋆] Jingkang Wang[1,2⋆] Ze Yang[1,2]
Sivabalan Manivasagam[1,2] Raquel Urtasun[1,2]

Waabi[1] University of Toronto[2]
{ychen, jwang, zyang, smanivasagam, urtasun}@waabi.ai

Abstract. In this supplementary material, we provide additional information on G3R, experimental setup, additional quantitative and qualitative results, limitations, and broader implications. We first provide additional information and motivation on G3R (Sec A). In Sec. B, we provide details on baseline implementations and how we adapt them to urban-driving and drone datasets. Next, we provide the experimental setup for evaluation on urban-driving and drone datasets in Sec. C. We then show more qualitative comparison with baselines (Sec. D.1), multi-camera simulation results (Sec. D.2) and a generalization study across datasets (Sec. D.3). Finally, we discuss the limitations and future works (Sec. E), broader impact (Sec. F) and responsibility to human subjects (Sec. G). Additionally, we include a supplementary video, **supplementary_video.mp4** providing an overview of our methodology and videos to demonstrate the efficacy of G3R.

A G3R Implementation Details

We first discuss three major paradigms for scene reconstruction as shown in main-paper-Fig. 2 and then present implementation details for G3R.

A.1 Comparison of Three Paradigms for Scene Reconstruction

For better understanding, we provide detailed algorithms for three paradigms for scene reconstruction discussed in the main paper. Each algorithm box depicts the paradigm's approach to reconstruct a new scene at inference time.

Algorithm 1 and 2 show the generalizable novel view synthesis (Fig. 2a) and per-scene optimization paradigms (Fig. 2b) separately. Specifically, existing generalizable approaches select a few reference images (usually ≤ 5) for feed-forward prediction of intermediate representation and then decode/render the feature representation to produce the rendered images. These approaches learn data-driven priors across multiple scenes and enable fast reconstruction. They need to reconstruct the scene again with different source images when rendering a new view. Existing generalizable approaches work only for small objects/scenes

⋆ Equal contributions.

Algorithm 1 Generalizable Novel View Synthesis

Inputs: Source Images $\mathbf{I}^{\text{src}}$, target view Π^{tgt}, reconstruction encoder G_θ, decoder network $D_\theta : \mathcal{S} \to \mathbf{I}$
$\mathbf{I}^{\text{src}}_{\text{nn}} = \text{Select}(\mathbf{I}^{\text{src}}, \Pi^{\text{tgt}})$ # select nearest neighboring source views
$\mathcal{S}_{\text{nn}} \leftarrow G_\theta(\mathbf{I}^{\text{src}}_{\text{nn}}, \Pi^{\text{tgt}})$ # predicted representation depends on view selection
$\hat{\mathbf{I}}^{\text{tgt}} = D_\theta(\mathcal{S}_{\text{nn}}, \Pi^{\text{tgt}})$ # render single target image from target view
Return $\mathcal{S}_{\text{nn}}$ # only renders views close to Π^{tgt}, need to re-run if it changes

Algorithm 2 Per-Scene Reconstruction by Gradient-Descent

Input: Initial scene representation $\mathcal{S}^{(0)}$, source images $\mathbf{I}^{\text{src}}$, renderer $f_{\text{render}} : \mathcal{S} \to \mathbf{I}$, optimiaztion iterations T (usually > 1000)
for $t = 0, 1, 2, \ldots, T-1$ **do**
  $\hat{\mathbf{I}}^{\text{src}} = f_{\text{render}}(\mathcal{S}^{(t)})$
  $\nabla_{\mathcal{S}^{(t)}} \leftarrow \nabla \|\mathbf{I}^{\text{src}} - \hat{\mathbf{I}}^{\text{src}}\|_2$
  $\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} + \nabla_{\mathcal{S}^{(t)}}$
end for
Return $\mathcal{S}^{(T)}$

and small view changes due to limited network capacity and handle a small number of source images due to memory constraints.

Recently, neural rendering approaches such as NeRF and 3D Gaussian Splatting have achieved realistic reconstructions for large scenes. These methods take all source images and reconstruct a 3D representation via energy minimization and differentiable rendering to the source views. However, they require a costly per-scene optimization process which usually takes several hours ($T > 1000$) and often exhibit artifacts when the view changes are large due to overfitting.

To enable fast large scene reconstruction while achieving high-fidity rendering performance, we instead propose to learn a network that iteratively refines a 3D scene representation with 3D gradient guidance (Algorithm 3). We highlight the major differences of G3R paradigm compared to the other two paradigms in red. Our key idea is to learn a single reconstruction network that iteratively updates the 3D scene representation, combining the benefits of data-driven priors from fast prediction methods with the iterative gradient feedback signal from per-scene optimization methods. G3R can be viewed as a "learned optimizer" that leverages spatial correlation and data-driven priors for fast scene reconstruction.

A.2 G3R Training Algorithm

We further show the presudocode algorithm for G3R reconstruction network training in Algoirthm 4 (See Eqn. 1, 4 and 6). G3R-Net takes current 3D neural Gaussians $\mathcal{S}^{(t)}$ and 3D gradient $\nabla_{\mathcal{S}^{(t)}}$ and output the refinement $\Delta\mathcal{S}^{(t)}$. We update the parameters of the reconstruction network and transformation MLP at every update step t.

Algorithm 3 Gradient-Guided Generalizable Reconstruction (G3R)

Input: Initial scene representation $\mathcal{S}^{(0)}$, source Images $\mathbf{I}^{\text{src}}$, renderer $f_{\text{render}} : \mathcal{S} \to \mathbf{I}$, reconstruction network G_θ, update iterations $T = 24$
for $t = 0, 1, 2, \ldots, 24$ **do**
  $\hat{\mathbf{I}}^{\text{src}} = f_{\text{render}}(\mathcal{S}^{(t)})$
  $\nabla_{\mathcal{S}^{(t)}} \leftarrow \nabla \|\mathbf{I}^{\text{src}} - \hat{\mathbf{I}}^{\text{src}}\|_2$ # lift 2D to 3D as gradients
  $\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} + \gamma(t) \cdot G_\theta(\mathcal{S}^{(t)}, \nabla_{\mathcal{S}^{(t)}}; t)$ # iteratively refine the 3D representation
end for
Return $\mathcal{S}^{(T)}$

Algorithm 4 G3R-Net Training

Input: Data $\mathcal{D}$: collection of (scene $\mathcal{S}^{(0)}$, images $\mathbf{I}$, poses Π) pairs, f_{rast}: differential tile renderer, G_θ: generalizable reconstruction network, f_{mlp}: transformation MLP, $\gamma(t)$ update scheduler
while G_θ not converged **do**
  $\mathcal{S}^{(0)}, \mathbf{I}, \Pi = \text{Sample}(\mathcal{D})$
  $(\mathbf{I}^{\text{src}}, \Pi^{\text{src}}), (\mathbf{I}^{\text{tgt}}, \Pi^{\text{tgt}}) = \text{Split}(\mathbf{I}, \Pi)$
  for $t = 0, 1, 2, \ldots, T-1$ **do**
    $\nabla_{\mathcal{S}^{(t)}} = \nabla \|\mathbf{I}^{\text{src}} - f_{\text{rast}}(f_{\text{mlp}}(\mathcal{S}^{(t)}); \Pi^{\text{src}})\|_2$
    $\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} + \gamma(t) \cdot G_\theta(\mathcal{S}^{(t)}, \nabla_{\mathcal{S}^{(t)}}; t)$
    $\text{loss} = \mathcal{L}(f_{\text{rast}}(f_{\text{mlp}}(\mathcal{S}^{(t+1)}); \Pi), \mathbf{I})$
    loss.backward()
    update G_θ and f_{mlp}
  end for
end while

A.3 G3R Implementation Details

Scene Representation: We develop our model based on the 3DGS implementation[1] [5]. We disable spherical harmonics in our model for simplicity and efficiency following [8]. Moreover, we empiricially find the performance drops are minor when disabling spherical harmonics, as also observed in 3DGS [6]. The dimension C of the feature vector $h_i \in \mathbb{R}^C$ is set to 46, with 32 for the latent feature and the remaining 14 for Gaussian attributes including position ($\mathbb{R}^3$), scale ($\mathbb{R}^3$), orientation ($\mathbb{R}^4$), color ($\mathbb{R}^3$), and opacity ($\mathbb{R}^1$).

Reconstruction Network (G3RNet): We use two generalizable networks with the same architecture for the static background and dynamic scene. We borrow the encoder-decoder UNet architecture from SparseResUNet in `torchsparse` [13] and do not tune the architecture. The 3D neural Gaussians and gradients are concatenated as the input of G3R-Net. The timestep positional encodings are concatenated with points' features output from the last encoder layer and fed to the decoder. For the background reconstruction network, we use a 2D CNN with 2 residual blocks, without downsampling or upsampling. For the transformation

[1] `https://github.com/wanmeihuali/taichi_3d_gaussian_splatting`

MLP network f_{mlp} that converts the 3D neural Gaussians to a set of explicit 3D Gaussians, we adopt one linear layer with a `tanh` activation. The output is combined with a learning rate decay factor $\gamma(t)$ to ensure gradual updates. The input raw gradient values are normalized for each channel by dividing them by the maximal absolute value in that channel.

Training and Inference: During training, we subsample 800k points in total for the static background and dynamic actors to fit into GPU memory. During inference, we subsample 3 million points for higher photorealism. To model the sky, we use a sphere image with a fixed radius (*i.e.*, 2048 meters to the center of the ego vehicle at the last frame). As most parts of the sky scene are not visible in the camera, we further crop the top and bottom part of the sphere to only keep the region between $30°N$ and $15°S$ to reduce the memory usage. We initialize the sky points with a resolution of 512×2048 during training, while using 1024×4096 during inference. We select closest 10 source and target frames to train the model. To produce the camera simulation results in D.2 and **supplementary_video.mp4**, we use all source images. λ_{lpips} and λ_{reg} are both set to 0.01. We train our model on front-facing camera and filter actors/points that are not visible in the field of view. For multi-camera simulation, we finetune the model on all cameras for 100 iterations. To further speed up the reconstruction while slightly reducing the photorealism, we also introduce G3R (turbo), where we reduce the number of static/dynamic points to 1.5 million, the sky resolution to 512×2048, and the number of reconstruction steps to 12.

Additional details for BlendedMVS: We initialize 3D Gaussian points by sampling on the surface of provided mesh. We use the high-resolution (1536×2048) images. During training we take 25 input source images, and 25 as novel views. There are no dynamic actors in BlendedMVS, so we only model the static background in G3R. We also do not model a sky-region, as distant regions not covered by the mesh are masked out in the input images. During training, we subsample 1.5 million points, while during inference we subsample 3.5 million points. The *turbo* version for BlendedMVS is with 2.5 million points and 24 update steps. However, in each update step, half of the source images are subsampled (each scene has an average of 381 images).

B Implementation Details for Baselines

We now review generalizable reconstruction baseline methods and per-scene optimization methods we compare against. Unless stated otherwise, we train all generalizable approaches using the same training data as G3R and optimize 3D representations of validation scenes individually with the same source frames for per-scene optimization approaches.

B.1 MVSNeRF

MVSNeRF [2] is a generalizable radiance field reconstruction method that employs a deep neural network to process a few nearby input views and generate

the radiance fields representation. Specifically, it builds a plane-swept 3D cost volume by warping 2D image features (inferred by a 2D CNN) from input views. Then it leverages a 3D CNN to reconstruct a neural scene volume, encoding both local scene geometry and appearance information. This 3D neural scene volume is decoded with a multi-layer perceptron (MLP) to infer density and radiance at arbitrary continuous locations using tri-linearly interpolated neural features inside the scene volume. Following the original paper, to enhance the rendering realism and leverage more input frames, we fine-tune the neural scene volume along with the MLP decoder for one epoch (around 30 minuites). We run the official repository[2] on PandaSet in our experiments. To handle unbounded driving scenes, we set the maximum rendering range to be 300 meters for each frame and sample 128 points for each ray during volume rendering.

B.2 ENeRF

ENeRF [7] constructs a sequential cost volume to predict the approximate geometry and conducts efficient depth-guided sampling. To meet the requirements of the CNN used in ENeRF, we crop the image to 1920×1056 on PandaSet so that the image dimensions are divisible by 32. Due to GPU memory contraints, we downscale the images $2\times$ on PandaSet and BlendedMVS during training, but during inference we use the original full resolution. We train two models from scratch on PandaSet and BlendedMVS training scenes for 300 epochs using the official repository[3]. We adopt the expoential learning rate decay schedule with `gamma=0.5` and `decay_epochs=50` During training, we select 4 source images with the closest viewpoints to each target view. We choose 2 source images for PandaSet and 4 for BlendedMVS during inference as it empirically produces the best performance. When taking more source images (*i.e.* 5), ENeRF produces more blurry results (-0.73 drop in PSNR on PandaSet) due to geometry inaccuracy and dynamics.

B.3 GNT

GNT [14] samples points along each target ray and predicts the pixel color by learning the aggregation of view-wise features from the epipolar lines using transformers. We adopt the official repository[4] and use `gnt_realestate` config to train the models on PandaSet and BlendedMVS. Specifically, we use the original image resolution and train each model for 250k and adjust the batch size to fit within 24GB GPU memory. We choose 2 source views on PandaSet and 10 for BlendedMVS to increase the coverage. When taking more source images (*i.e.* 5), GNT produces more blurry results (-1.98 drop in PSNR on PandaSet) due to geometry inaccuracy and dynamics. During inference, we sample 192 points per pixel as suggested by the official guidelines.

[2] `https://github.com/apchenstu/mvsnerf`
[3] `https://github.com/zju3dv/ENeRF`
[4] `https://github.com/VITA-Group/GNT`

B.4 PixelSplat

Concurrent work PixelSplat [1] predicts 3D Gaussians with a 2-view epipolar transformer to extract features and then predict the depth distribution and pixel-aligned Gaussians. We adopt the official repository[5] and use 2× A6000 (48GB) to train the models. Due to the GPU memory constraint, we downscale the image resolution to 360×640 for PandaSet and 384×512 for BlendedMVS. We note that the original work uses an 80GB A100 for training and handles 256×256 resolution. We use `re10k` config and train each model for 100k iterations with `batch_size=1`.

PixelSplat cannot handle large view changes and produces rendering results with significant visual artifacts due to inaccurate geometry estimation (*e.g.*, blurry appearance) especiallly on BlendedMVS. To address this issue, we enhance PixelSplat, named PixelSplat++, to leverage the 3D scaffold to reduce ambiguity and take all available source images for good coverage. Specifically, we first initialize a unified 3D Gaussian representation, unproject DINO [10] image features to 3D points and then use a shared decoder to predict the 3D Gaussian residues. Similar to G3R, we randomly select one target view, and then choose 10 nearest source views and additional 9 nearest target views during training. We use both the source and target views to supervise the shared decoder and adopt L2 and LPIPS losses. Compared to PixelSplat, PixelSplat++ takes all source images (original resolution: 1536×2048) as inputs and predicts a higher-quality 3D representation, achieving a signficiant performance boost at novel views.

B.5 Instant-NGP

Instant-NGP [9] introduces efficient hash encoding, accelerated ray sampling, and fully fused MLPs to neural volumetric rendering. In our experiments, we use the official repository[6] and normalize the scenes to occupy the unit cube and set `aabb_scale` as 32 for PandaSet and 8 for BlendedMVS to handle the background regions (*e.g.*, far-away buildings and sky) outside the unit cube. We further enhance Instant-NGP with depth supervision for better performance. Sepcifically, we aggregate the recorded LiDAR data and create a surfel triangle representation based on estimated per-point normals. Then we render a pseudo-ground-truth depth image at each camera training viewpoint, which is used for depth supervision. The models are trained for 20k iterations on PandaSet scenes and 100k on BlendedMVS, and converge on the training views.

B.6 3DGS

The vanilla version of 3D Gaussian Splatting (3DGS) does not support dynamic scenes or unbounded regions such as the sky. We therefore employ the same

[5] `https://github.com/dcharatan/pixelsplat`
[6] `https://github.com/NVlabs/instant-ngp`

extended version with decomposed foreground, background, and distant regions as in G3R. The 3DGS baseline used in this study can be considered as replacing G3RNet during inference with a fixed Stochastic Gradient Descent (SGD) update. More specifically, we utilize the Adam optimizer with a learning rate of 0.1 and apply learning rate decay by a factor of 0.5 at iterations 200, 300, 400, and 450. The training process is conducted for a total of 500 iterations. Training for longer iteration does not further improve the performance on the validation views. It is worth noting that, in each iteration, we aggregate gradients from all source images, which contrasts with other approaches that typically use a single source image per iteration. Aggregating gradients from all source frames improves performance and enables more stable training. We employ the same number of Gaussian points in 3DGS optimiaztion as in G3R inference stage. Note that we remove adaptive density control in our experiments as it does not help 3DGS much in test views when it has dense initialization, unless we allow it to grow significantly more points (PSNR+0.58 with 50% more points (5.3M) in BlendedMVS), at the cost of increased resources. We also note that enhancing 3DGS with neural Gaussians leads to better results (+0.38 PSNR) and faster early convergence.

B.7 Efficiency Comparison

Tab. A1, reports the model capacity and training efficiency of baselines and G3R. G3R's capacity and efficiency is on par with generalizable methods.

Table A1: Model capacity and training effiency of generalizable approaches.

Method	Train time	Train mem	Recon mem	#param
ENeRF	108 hours	24GB	10GB	4.3×10^5
GNT	49 hours	23GB	21GB	8.8×10^5
PixelSplat	110 hours	48GB	11GB	1.3×10^8
G3R	60 hours	20GB	24GB	2.6×10^7

C Experiment Details

C.1 Experiment Setup

We conduct experiments on two public datasets with large real-world scenes: PandaSet [17] and BlendedMVS [20]. PandaSet contains 103 urban driving scenes, each with 6 HD (1920×1080) cameras and LiDAR sweeps. We select 7 diverse scenes (`001, 030, 040, 080, 090, 110, 120`) for testing and the remaining are used for training. We consider the front camera only for all baselines and G3R for quantitative evaluation experiments. BlendedMVS-large is a collection of 29 real-world scenes captured by a drone. We use high-resolution

Fig. A1: Large view changes on BlendedMVS. We highlight the target view in red and 4 closest source views in blue. The distance and view-orientation changes between the source views and the target view are large. The image warping (rightmost column, colored by image source index, missing regions in black) shows that limited source views cannot get full coverage to synthesize the target view.

(1538 × 2048) images in our experiments. The list of *large scenes* are based on github split[7] We select 4 scenes for testing (`58eaf1513353456af3a1682a, 5b69cc0cb44b61786eb959bf,5bf18642c50e6f7f8bdbd492, 5af02e904c8216-544b4ab5a2`), each containing 68 to 836 images (381 on average). Unless stated otherwise, for both datasets, we use every other frame as source and the remaining for test. We use all available images in the supplementary camera simulation demonstrations for novel scene manipulations such as sensor shifts and actor editing in Sec 4.2 and Sec D.2.

C.2 Metrics

We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [15] and perceptual similarity (LPIPS) [21] to evaluate the photorealism of novel view synthesis. To measure the efficiency of different approaches, we also report the reconstruction time and rendering FPS using a single RTX 3090. We note that the generalizable approaches (*e.g.*, ENeRF, GNT, PixelSplat) usually need to reconstruct the scene again with different source images when rendering at new target views We report the reconstrucion time for one feed-forward prediction. For MVSNeRF, we report the prediction + finetuning time in Tab. 1. In contrast, the per-scene optimization methods, PixelSplat++, and G3R obtain a unified representation that takes all input views into account.

C.3 Evaluation on BlendedMVS

BlendedMVS has more challenging novel views, as the distance between two nearby views can be large as shown in Fig. A1. We note that there is no explicit interpolation/extrapolation split for BlendedMVS as the multi-pass drone trajectories are not available.

C.4 Comparison with Generalizable Baselines

We note that generalizable baselines including ENeRF, GNT and PixelSplat can access all source images but cannot take all images at once due to their

[7] `https://github.com/kwea123/BlendedMVS_scenes/`

limitations. In our experiments, we run baselines in PandaSet for each test frame using 2 closest source images. When taking more source images (*i.e.* 5), warping-based methods such as ENeRF and GNT produce more blurry results (-0.73/-1.98 PSNR) due to geometry inaccuracy and dynamics. PixelSplat cannot take more than 2 views due to the memory constrains (48GB) as it predicts pixel-aligned Gaussians and the memory increases linearly with the number of input views. PixelSplat++ takes all source images as input but it is still worse than G3R as the single-step prediction has limited capacity.

D Additional Experiments and Analysis

We provide additional results and analysis for scene reconstruction on PandaSet and BlendedMVS. We then showcase more camera simulation examples and a generalization study on Waymo Open Dataset (WOD) using G3R.

D.1 Additional Qualitative Examples

We provide additional qualitative comparison with state-of-the-art (SoTA) scene reconstruction approaches on PandaSet. As shown in Fig. A2, compared to G3R, exsiting SoTA generalization approaches suffer from noticeable artifacts such as blurry rendering results, unnatural discontinuities and inaccurate color palette. In Fig. A3, we further compare G3R with SoTA per-scene optimization approaches. Instant-NGP has severe artifacts on dynamic actors due to lack of dynamics modelling and 3DGS can produce noticeable artifacts (*e.g.*, black holes) sometimes. In contrast, G3R leads to the most robust rendering results while shortenning the reconstruction times to 2 minutes (10× speedup).

Fig. A2: Qualitative comparison to generalizable approaches on PandaSet.

We also present more qualitative comparison with SoTA scene reconstruction approaches on BlendedMVS in Fig. A4 and Fig. A5. As shown in Fig. A4, ENeRF, GNT and PixelSplat cannot handle large view changes and produces

Fig. A3: Qualitative comparison to per-scene optimization approaches on PandaSet.

rendering results with signficant visual artifacts, including blurry appearance and unnatural discontinuities due to the challenges of estimate high-quality geometry from limited views. PixelSplat++ achieves a significant performance boost but still produces blurry results compared to G3R due to the chalenge of one-step prediction with limited network capacity. In Fig. A5, we compare G3R with Instant-NGP and 3DGS, and show comparable or better rendering performance with signficiant reconstruction acceleration.

Robust 3D Gaussian Prediction : To understand why our method achieves superior performance over 3DGS per-scene optimization, we compare the rendering performance at source and novel views. We show a qualitative comparison between 3DGS and G3R where each method gets 20 consecutive frames as input, and then renders the target view several meters forward from the last source view pose (Fig. A6). As shown in Tabs. A2 and A3, while 3DGS has sufficient capacity to memorize the source frames, it suffers a significant performance drop (*e.g.*, 1.59 PSNR decrease and 0.054 LPIPS increase) when rendering at novel views. This may be due to the 3DGS-optimized Gaussians having alpha, covariance scales, and orientations that only work well for the source views it's optimized on, resulting in poor underlying geometry [3, 4]. In contrast, G3R yields more robust Gaussian representations and achieves better rendering performance at novel views on unseen scenes. This is because G3R is trained with novel view supervision across many scenes, which helps regularize the 3D neural Gaussians to generalize rather than merely memorize the source views. As a result, G3R predicts 3D gaussians in a more robust way and produces more realistic rendering performance in both training and extrapolated views.

Fig. A4: Qualitative comparison to generalizable approaches on BlendedMVS.

Fig. A5: Qualitative comparison to per-scene optimization approaches on BlendedMVS.

Table A2: 3DGS overfits to source views while G3R is more robust.

	PSNR↑	SSIM↑	LPIPS↓
3DGS (source views)	26.73	0.805	0.318
Ours (source views)	25.94	0.779	0.356
3DGS (novel views)	25.14	0.747	0.372
Ours (novel views)	25.22	0.742	0.371

Table A3: Comparison to 3DGS at extrapolated views (future 3 frames).

	PSNR (1st)	PSNR (2nd)	PSNR (3rd)
3DGS [6]	23.96	22.43	21.58
Ours	**24.13**	**23.35**	**22.82**

Fig. A6: Qualitative comparison of G3R to 3DGS on novel views in PandaSet.

D.2 Additional Camera Simulation Examples

We now showcase applying G3R for high-fidelity multi-camera simulation for a wide variety of large-scale driving scenes. In Fig. A7 and Fig. A8, G3R produce consistent and high-fidelity multi-camera or panorama image simulation for diverse scenarios. Please see Appendix E for additional anlaysis on the challenges of multi-camera simulation.

G3R can reconstruct an explicit standalone representation that models the dynamics, which allows us to control, edit and simulate different variations for robotics simulation. In Fig. A9 and Fig. A10, we show realistic and controllable multi-camera and panorama simulation results by either manipulating the positions of dynamic actors (scene manipulation) or changing the sensor locations (SDV camera sensor shifts). Please see **supplementary_video.mp4** for complete simulation videos. These results demonstrate the potential of G3R for scalable self-driving simulation for autonomy validation and training.

D.3 Additional Generalization Study

Finally, we supplement additional results on generalization study across different datasets. In Fig. A11, we directly apply a pretrained G3R model (on PandaSet)

Fig. A7: Multi-camera simulation on PandaSet.

Fig. A8: Panorama image simulation on PandaSet.

Fig. A9: Realistic and controllable multi-camera simulation.

Fig. A10: Realistic and controllable panorama image simulation.

and show it generalizes to new scenes in Waymo Open Dataset [12] (WOD). As shown in Fig. A11, G3R can generalize well across datasets with different sensor configurations (placements, sensor type, appearance etc) and can reconstruct new scenes in under a few minutes. This demonstrates the potential of G3R for scalable real-world camera simulation. Please see **supplementary_video.mp4** for the complete videos.

Fig. A11: Scene reconstruction on WOD with PandaSet-trained model.

D.4 Adaptive Density Control and Robustness Analysis

We experiment with adding density control to G3R and observe enhanced performance. Specifically, we initialize G3R with 25% points (0.9M), and grow the points at the 5th step (adding 8 new points around each point and downsample to 3.5M). The PSNR increases 1.04 compared to no densification, and is 0.42 lower than the original G3R. While achieving better performance, we notice that G3R has difficulty in handling extremely sparse initialization. Moreover, we test G3R with dense noisy points from MVS [16] (Fig. A12) and find G3R is robust to the noisy initialization (only 0.36 PSNR drop). For robotics applications, dense points from either LiDAR or fast MVS (∼2 min) is typically available.

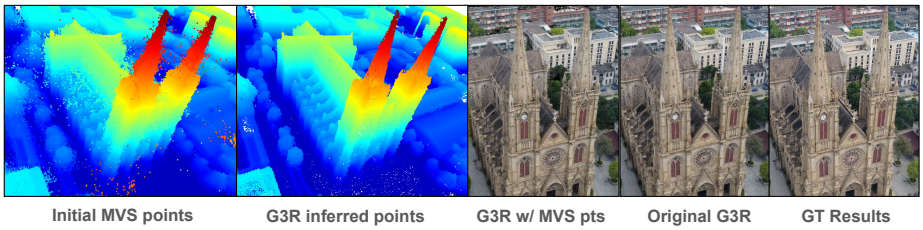

Fig. A12: G3R is robust to point initialization (zoom-in).

E Limitations and Future Works

While G3R can reconstruct unseen large scenes efficiently with high photorealism, there are several limitations as shown in Fig. A13. First of all, as shown in Fig. A13-leftmost, our approach has artifacts in large extrapolations (*e.g.*, $5 \sim 10$ meters shift), which may require scene completion and larger scale training to predict novel views with larger differences. Better surface regularization [3, 4] and adversarial training [11, 19] may mitigate these issues. Moreover, although G3R shows strong generalizability and robustness thanks to the 3D gradients and recursive updates (Fig. A12), it relies on dense points as initialization and it is an open problem to build effective adaptive density control mechanism for G3R similar to original 3DGS [6] to prune and grow 3D Gaussians. We notice that the reconstruction quality of G3R degrades on sparse initialization.

We also do not model non-rigid deformations [8] and emissive lighting for more controllable simulation. We also notice more artifacts in multi-camera simulation (Fig. A13-second-column), primarily due to the different exposure and white balance settings across cameras, misalignment due to calibration errors, as well as motion blur and rolling shutter for the side cameras. Additionally, nearby dynamic actors have more artifacts, particularly due to the resolution of the Gaussian points (Fig. A13-third-column). Incorporating multi-resolution or level-of-detail modelling to the neural 3D Gaussians could improve this. Lastly,

there are artifacts when points are missing for some regions (e.g., the higher part of the building, particularly in the WOD dataset), because these regions are not scanned by the LiDAR and are thus modeled as part of the sky (Fig. A13-rightmost). SfM and MVS points can be added to mitigate this problem [18].

Fig. A13: Failure cases of G3R.

F Broader Impact

G3R provides a scalable and efficient way to reconstruct large-scale real-world scenes for high-quality and real-time rendering. Its ability to generate controllable camera simulation videos (*e.g.*, scene manipuation and sensor shifts) can potentially improve the robustness and safety of robotic systems for real-world environments or can be used to build immersive experience in VR/AR applications.

G Responsibility to Human Subjects

We use two public self-driving datasets in this research: PandaSet [17] and Waymo Open Dataset [12], both of which involve 2D/3D scanning of real-world street scenes with different sensors such as camera and LiDAR. They have undergone vetting for ethical considerations pertaining to data collection. Despite containing pedestrian data, neither dataset reveals any personally identifiable information or offensive content. Furthermore, our research does not involve generating new content related to human subjects.

References

1. Charatan, D., Li, S., Tagliasacchi, A., Sitzmann, V.: pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. arXiv (2023)
2. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: ICCV (2021)
3. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: Gaussianpro: 3d gaussian splatting with progressive propagation. arXiv (2024)
4. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. arXiv (2023)
5. Hu, Y., Li, T.M., Anderson, L., Ragan-Kelley, J., Durand, F.: Taichi: a language for high-performance computation on spatially sparse data structures. ACM Transactions on Graphics (TOG) **38**(6), 201 (2019)
6. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. TOG (2023)
7. Lin, H., Peng, S., Xu, Z., Yan, Y., Shuai, Q., Bao, H., Zhou, X.: Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH Asia 2022 Conference Papers (2022)
8. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. arXiv (2023)
9. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding (2022)
10. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv (2023)
11. Roessle, B., Müller, N., Porzi, L., Bulò, S.R., Kontschieder, P., Nießner, M.: Ganerf: Leveraging discriminators to optimize neural radiance fields. ACM Trans. Graph. (2023)
12. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020)
13. Tang, H., Yang, S., Liu, Z., Hong, K., Yu, Z., Li, X., Dai, G., Wang, Y., Han, S.: Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In: IEEE/ACM International Symposium on Microarchitecture (MICRO) (2023)
14. Wang, P., Chen, X., Chen, T., Venugopalan, S., Wang, Z., et al.: Is attention all nerf needs? arXiv (2022)
15. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. TIP (2004)
16. Wu, J., Li, R., Xu, H., Zhao, W., Zhu, Y., Sun, J., Zhang, Y.: Gomvs: Geometrically consistent cost aggregation for multi-view stereo. In: CVPR (2024)
17. Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al.: Pandaset: Advanced sensor suite dataset for autonomous driving. In: ITSC (2021)
18. Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., Lang, X., Zhou, X., Peng, S.: Street gaussians for modeling dynamic urban scenes. arXiv (2024)
19. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: Unisim: A neural closed-loop sensor simulator. arXiv (2023)

20. Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. CVPR (2020)
21. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. CVPR (2018)