Block-NeRF: Scalable Large Scene Neural View Synthesis Supplement

Matthew Tancik[1*] Vincent Casser[2] Xinchen Yan[2] Sabeek Pradhan[2]
Ben P. Mildenhall[3] Pratul Srinivasan[3] Jonathan T. Barron[3] Henrik Kretzschmar[2]

[1]UC Berkeley [2]Waymo [3]Google Research

1. Model Parameters / Optimization Details

Our network follows the mip-NeRF structure. The network f_σ is composed of 8 layers with width 512 (Mission Bay experiments) or 1024 (all other experiments). f_c has 3 layers with width 128 and f_v has 4 layers with width 128. The appearance embeddings are 32 dimensional. We train each Block-NeRF using the Adam [2] optimizer for 300 K iterations with a batch size of 16384. Similar to mip-NeRF, the learning rate is an annealed logarithmically from $2 \cdot 10^{-3}$ to $2 \cdot 10^{-5}$, with a warm up phase during the first 1024 iterations. The coarse and fine networks are sampled 256 times during training and 512 times when rendering the videos. The visibility is supervised with MSE loss and is scaled by 10^{-6}. The learned pose correction consists of a position offset and a 3×3 residual rotation matrix, which is added to the identity matrix and normalized before being applied to ensure it is orthogonal. The pose corrections are initialized to 0 and their element-wise $\ell 2$ norm is regularized during training. This regularization is scaled by 10^5 at the start of training and linearly decays to 10^{-1} after 5000 iterations. This allows the network to learn initial geometry prior to applying pose offsets.

Each Block-NeRF takes between 9 and 24 hours to train (depending on hyperparameters). We train each Block-NeRF on 32 TPU v3 cores available through Google Cloud Compute, which combined offer a total of 1680 TFLOPS and 512 GB memory. Rendering an 1200×900px image for a single Block-NeRF takes approximately 5.9 seconds. Multiple Block-NeRF can be processed in parallel during inference (typically fewer than 3 Block-NeRFs need to be rendered for a single frame).

2. Block-NeRF Size and Placement

We include qualitative comparisons in Figure 2 on the Mission Bay dataset to complement the quantitative comparisons in (§5.3, Table 2). In this figure, we provide comparisons on two regimes, one where each Block-NeRF contains the same number of weights (left section) and one where the total number of weights across all Block-NeRFs is fixed (right section).

*Work done as an intern at Waymo.

3. Block-NeRF Overlap Comparison

In the main paper, we include experiments on Block-NeRF size and placement (§5.3). For these experiments, we assumed a relative overlap of 50% between each pair of Block-NeRFs, which aids with appearance alignment.

Table 1 is a direct extension of Table 2 in the main paper and shows the effect of varying block overlap in the 8 block scenario. Note that varying the overlap changes the spatial block size. The original setting in the main paper is marked with an asterisk.

The metrics imply that reducing overlap is beneficial for image quality metrics. However, this can likely be attributed to the resulting reduction in block size. In practice, having an overlap between blocks is important to avoid temporal artifacts when interpolating between Block-NeRFs.

Overlap	Size	PSNR↑	SSIM↑	LPIPS↓
0%	77 m	26.77	0.895	0.262
25%	97 m	26.75	0.894	0.269
50%*	116 m	26.59	0.890	0.278
75%	136 m	26.51	0.887	0.283

Table 1. Effect of different NeRF overlaps in the 8 block scenario with 0.25M weights per block (2M weights in total). The original setting used in the main paper is marked*.

4. Block-NeRF Interpolation Details

We experiment with multiple methods to interpolate between Block-NeRFs and find that simple inverse distance weighting (IDW) in image space produces the most appealing videos due to temporal smoothness. We use an IDW power p of 4 for the Alamo Square renderings and a power of 1 for the Mission Bay renderings. We experiment with 3D inverse distance weighting for each individual pixel by projecting the rendered pixels into 3D space using the expected

ray termination depth from the Block-NeRF closest to the target view. The color value of the projected pixel is then determined using inverse distance weighting with the nearest Block-NeRFs. Artifacts occur in the resulting composited renders due to noise in the depth predictions. We also experiment with using the Block-NeRF predicted visibility for interpolation. We consider imagewise visibility where we take the mean visibility of the entire image and pixelwise visibility where were directly utilize the per-pixel visibility predictions. Both of these methods lead to sharper results but come at the cost of temporal inconsistencies. Finally we compare to nearest neighbor interpolation where we only render the Block-NeRF closest to the target view. This results in harsh jumps when transiting between Block-NeRFs.

5. Structure from Motion (COLMAP)

We use COLMAP [3] to reconstruct the Mission Bay dataset. We first split the dataset into 8 overlapping blocks with $97\,\mathrm{m}$ radius each based on camera positions (each block has roughly 25% overlap with the adjacent block). The bundle adjustment step takes most of the time in reconstruction and we do not see significant improvements if we increase the radius per block. We mask out movable objects when extracting feature points for matching, using the same segmentation model as Block-NeRF. We assume a pinhole camera model and provide camera intrinsics and camera pose as priors for running structure-from-motion. We then run multi-view stereo within each block to produce dense depth and normal maps in 3D and produce a dense point cloud of the scene. In our preliminary experiments, we ran Poisson meshing [1] on the fused dense pointcloud to reconstruct textured meshes but found that the method fails to produce reasonably-looking results due to the challenging geometry and depth errors introduced by reflective surfaces and the sky. Instead, we leverage the fused pointcloud and explore two alternatives, namely, point rendering and surfel rendering, respectively. To render the test view, we selected the nearest scene and use `OSMesa` off-screen rendering assuming the Lambertian model and a single light source.

In Table 2, we compare two different rendering options for the densely reconstructed pointcloud. We discard the invisible pixels when computing the PSNR for both methods, making the quantitative results comparable to our Block-NeRF setting.

In Figure 1, we show the qualitative comparisons between two rendering options with PSNR on the corresponding images. This reconstruction is sparse and fails to represent reflective surfaces and the sky.

https://docs.mesa3d.org/osmesa.html

Method	PSNR* (`train`) ↑	PSNR* (`test`) ↑
COLMAP (point)	13.019	11.933
COLMAP (surfel)	13.291	12.343

Table 2. Quantitative results for COLMAP. We discard invisible pixels (e.g., sky pixels that COLMAP fails to reconstruct) when computing the PSNR.

Figure 1. Qualitative results for COLMAP. We demonstrate the two rendering options using the fused pointcloud computed by COLMAP.

6. Examples from our Datasets

In Figure 3, we show the camera images from our Mission Bay dataset. In Figure 4, we show both camera images and corresponding segmentation masks from our Alamo Square dataset.

7. Societal Impact

7.1. Methodological

Our method inherits the heavy compute footprint of NeRF models and we propose to apply them at an unprecedented scale. Our method also unlocks new use-cases for neural rendering, such as building detailed maps of the environment (mapping), which could cause more wide-spread use in favor of less computationally involved alternatives. Depending on the scale this work is being applied at, its compute demands can lead to or worsen environmental damage if the energy used for compute leads to increased carbon emissions. As mentioned in the paper, we foresee further work, such as caching methods, that could reduce the compute demands and thus mitigate the environmental damage.

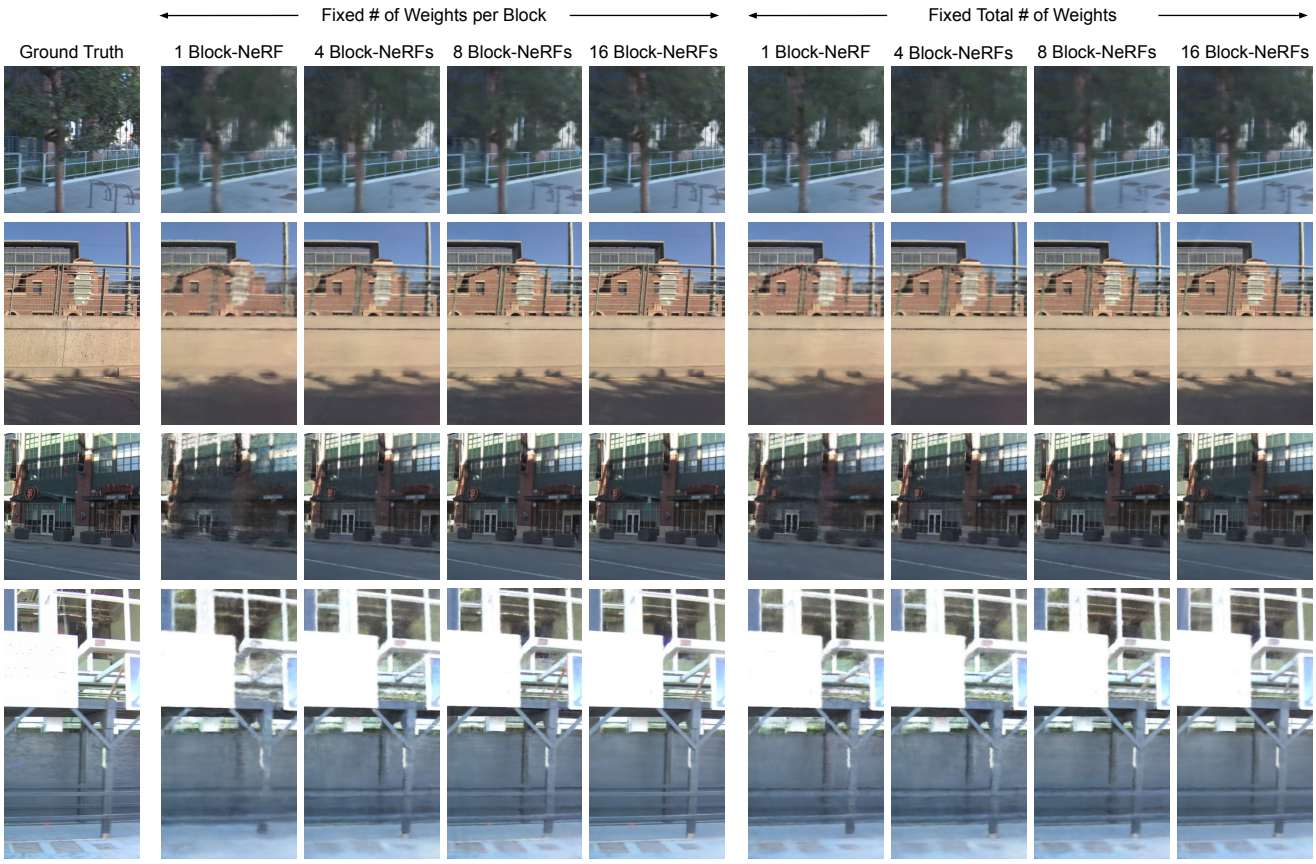

Figure 2. Qualitative results on Block-NeRF size and placement. We show results on the Mission Bay dataset using different options discussed in § 5.3 of the main paper.

Figure 3. Selection of images from our Mission Bay Dataset.

7.2. Application

We apply our method to real city environments. During our own data collection efforts for this paper, we were careful to blur faces and sensitive information, such as license plates, and limited our driving to public roads. Future applications of this work might entail even larger data collection efforts, which raises further privacy concerns. While detailed imagery of public roads can already be found on services like Google Street View, our methodology could promote repeated and more regular scans of the environment. Several companies in the autonomous vehicle space are also known to perform regular area scans using their fleet of vehicles; however some might only utilize LiDAR scans which can be less sensitive than collecting camera imagery.

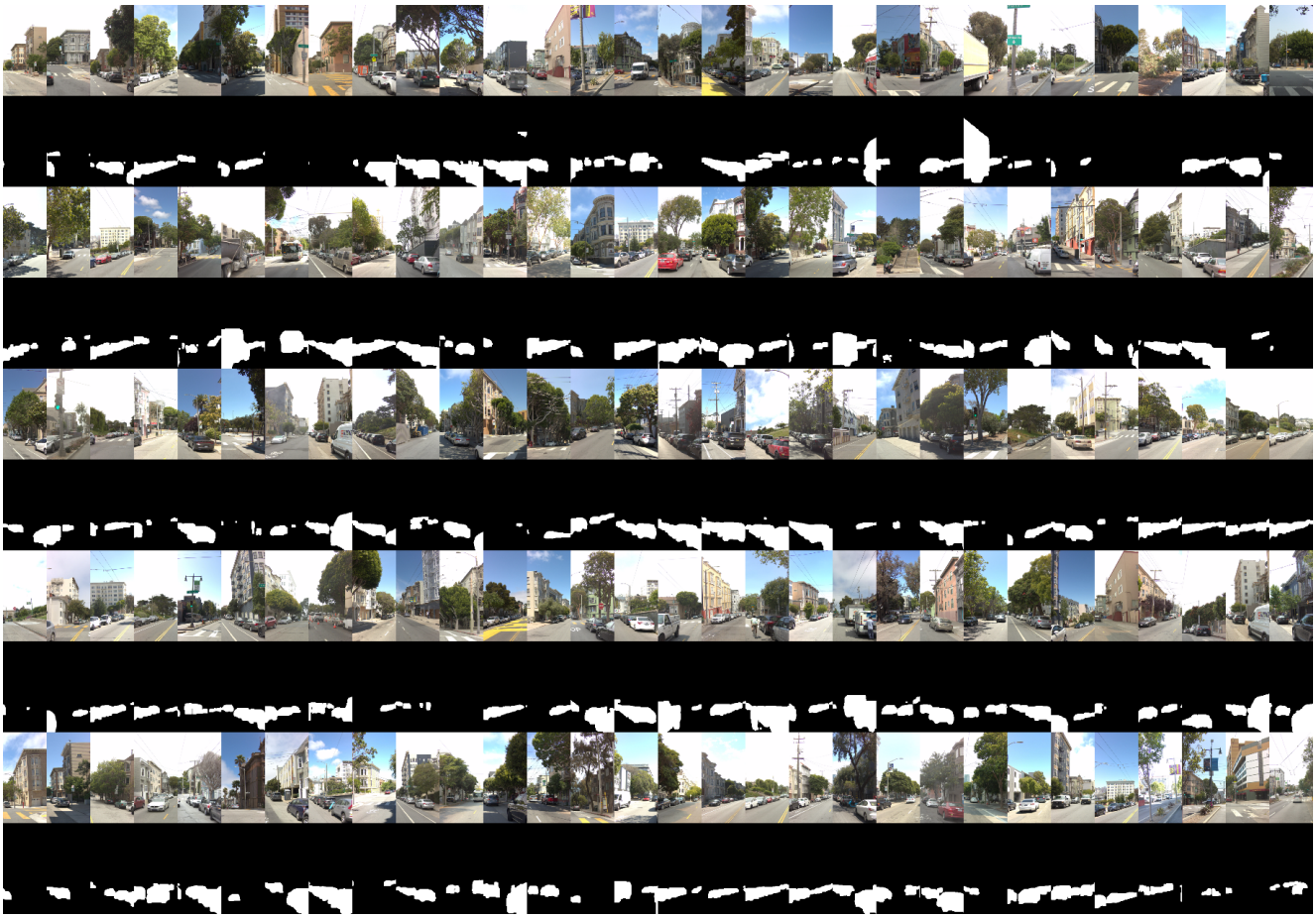

Figure 4. Selection of front-facing images from our Alamo Square Dataset, alongside their transient object mask predicted by a pretrained semantic segmentation model.

References

[1] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 2013. 2

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 1

[3] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. *CVPR*, 2016. 2