Supplementary for CityNeRF: Building NeRF at City Scale

1. Spatial Prior for Ray Sampling

Camera Poses Acqusition. Since city-scale scenes are generally large, it is difficult to use COLMAP [11] to estimate camera poses. We thus utilize the actual world coordinate system in Google Earth Studio [1] to obtain the ground truth camera poses. We scale the world coordinates proportionally to make the camera poses arranged in a reasonable range (not necessary to be within $[-1, 1]$) and maintain enough precision at the same time. Accordingly, the Fourier feature mappings $\sin(kx)$ in position encoding are allowed to take $k = 2^i$, $i < 0$ to accommodate these coordinate values outside a single sine/cosine period.

Near-far Bounds. Due to the large range of camera positions and the limited number of point samples per ray, NeRF severely suffers from a near-field ambiguity [4, 13] as shown in Fig. 1, where rarely observed scenes such as long-shot ones tend to be packed close to cameras to overfit each input view. Based on the fact that city scenes are always on earth surface and will not exceed a certain height, we impose a spatial prior to guide the network to sample within an effective range. The near-far planes of each ray are determined by computing their intersection of each ray to the *earth surface* and the *earth hull* with a predefined maximum altitude above the ground. An illustration of the near and far bounds is provided in Fig. 2(a), denoted with t_{near} and t_{far}. Specifically, we set the far bound of a ray to its intersection with earth surface by:

$$d_{far} = \frac{-2[\hat{\mathbf{u}} \cdot (\mathbf{o} - \mathbf{c})] \pm \sqrt{\nabla}}{2\|\hat{\mathbf{u}}\|^2}, \quad \text{and}$$
$$\nabla = 2[\hat{\mathbf{u}} \cdot (\mathbf{o} - \mathbf{c})])^2 - 4\|\hat{\mathbf{u}}\|^2(\|\mathbf{o} - \mathbf{c}\|^2 - r^2),$$

where $\hat{\mathbf{u}}$ is the ray direction, $\mathbf{o}$ is the ray origin, $\mathbf{c}$ is earth center and r is earth radius. Similarly, the near bound is derived by substituting r with $r + h$, where h denotes the hull of earth we set and the height of the tallest building in the scene should be no more than h.

2. Miscellaneous

2.1. Weighted Position Encoding

The fidelity of NeRF depends critically on the use of positional encoding. For the k-th component in the log-linear

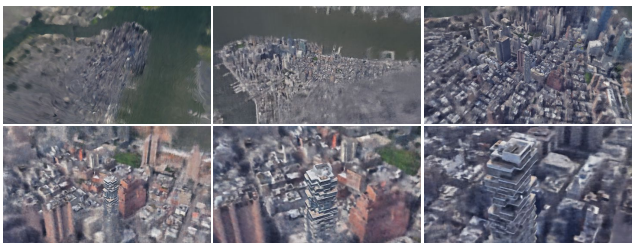

Figure 1. Near-field artifacts w/o using spatial prior.

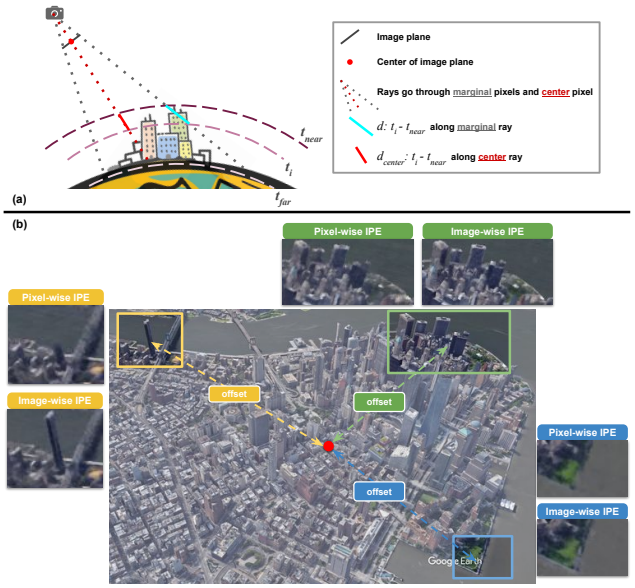

Figure 2. Qualitative comparison between using IPE calculated per pixel and the unified image-wise IPE.

spaced frequencies, the default PE [7] uses a Fourier feature mapping with $\gamma_k(x) = \sin(kx)$[1]. Instead of performing point-sampling along each ray, Mip-NeRF [2] divide the casted cone into a series of conical frustum and constructs an integrated positional encoding (IPE) representation of the volume covered by each conical frustum. They approximate the conical frustum with a multivariate Gaussian, obtaining the IPE as:

$$E_{x\sim\mathcal{N}(\mu,\sigma^2)}\left[\gamma_k(x)\right] = \sin(k\mu)\exp\left(-(k\sigma)^2/2\right).$$

[1]We eliminate $\gamma_k(x) = \cos(kx)$, and eliminate the case for y, z axes and viewing directions here for notation brevity.

These changes allow the MLP to reason about the size and shape of each conical frustum, instead of just its centroid. We experimentally found that this could boost its learning in modeling multi-scale scenes, while default PE may suffer from recovering the right geometries. Specifically, IPE can better capture the overall coarse geometry, though it may perform slightly inferior than default PE while modeling very fine detailed textures due to the dampened high-frequency components.

Note that IPE can be viewed as a special case of a general weighted position encoding, which assigns a smaller weight to high-frequency components conditioned on the input cone volumes to regularize different frequency bands,

$$\begin{aligned} \gamma_k(x) &= w_k(\alpha)\sin(kx), \\ \text{with} \quad w_k(\alpha) &= \exp\left(-(k\sigma)^2/2\right). \end{aligned} \tag{1}$$

The windowed position encoding (WPE) [5, 9, 10] also follows this weighted form with

$$w_k(\alpha) = \frac{1}{2}(1-\cos(\pi\,\mathrm{clamp}(\alpha-k,0,1))), \tag{2}$$

being proportional to the optimization progress. This treatment is suitable for modeling single-scale scenes and can encourage the model to capture the coarse geometry structure quicker at the starting phases, as explained in [10].

Through our design of CityNeRF, we take Mip-NeRF's IPE as the network input, rather than the original point-based PE. As suggested in Mip-NeRF [2], we can manipulate the integrated positional encoding by using a larger or smaller radius than the true pixel footprint, where a smaller radius shrinks the variance in the cross section of the cast cone to reflect our preference for preserving more details.

Image-wise IPE. Below we explain an optional design we integrated in CityNeRF to improve its performance in rendering city-scale scenes. Minor artifacts can be observed when we directly use IPE in rendering grand city views. Recall that Mip-NeRF models the size and shape of each conical frustum by scaling the positional encoding with its covariance. Due to pixel offsets, computing the near-far bound per ray leads to inconsistent sampling intervals within an image, especially in grand-view images. As the results, rays passing through edge pixels have much larger covariance than center ones, causing artifacts on the edge, such as blur and missing geometry as shown in Fig. 2(b). We eliminate this discrepancy by scaling the current frustum's covariance with a factor $t_{ratio} = d_{center}/d$, which is the ratio of the near-far plane distance between the current ray and the center ray. The modifying IPE hence becomes:

$$E_{x\sim\mathcal{N}(\mu,\sigma^2)}\left[\gamma_k(x)\right] = \sin(k\mu)\exp\left(-(k\cdot t_{ratio}\sigma)^2/2\right). \tag{3}$$

As a result, the frustum covariance within an image is unified to yield a more consistent result.

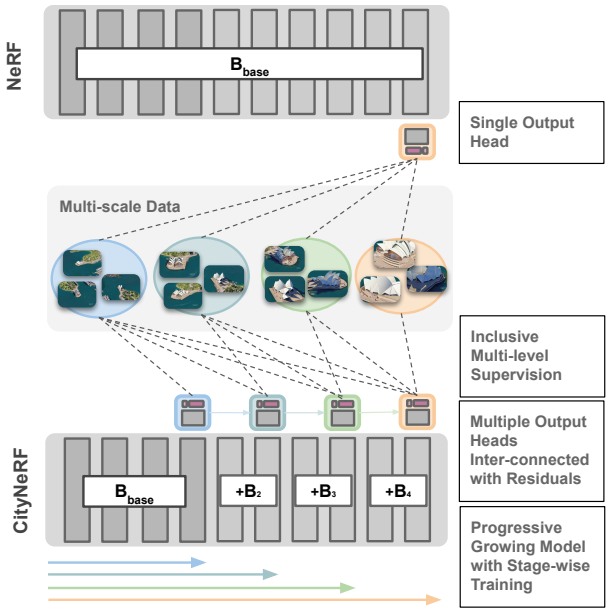

Figure 3. Schematic comparison between NeRF and CityNeRF.

2.2. Sampling around Soft Depth

To bring more fine-detailed geometries, a series of works has proposed to use either ground truth or additionally predicted depth information for more efficient point sampling along the ray [3, 8, 12]. In real cases where no ground-truth depth is available, one could consider loading a trained model to output a rough depth guides and sample around depth to provide high-concentrated samples. However, we found that this hard operation heavily relies on the guided depth and could bring unwilling artifacts when the guides are inaccurate. We propose an alternative to perform a third round sampling given the depth-like information derived from the NeRF's fine-stage sampling. This soft operation performs surprisingly well to boost the texture and geometry details if we want to enhance the high-frequency details. Note that, this additional round of sampling is not necessarily needed in early training stages, but only optional in the later stages when a final "spurt" in performance is desired.

3. More Qualitative Results and Visualizations

Model Comparisons. A schematic comparison between vanilla NeRF and CityNeRF is shown Fig. 3. In a nutshell, CityNeRF differs from NeRF in terms of the allocation of output heads with multi-level supervision, and the stage-wise training scheme with progressive growing model.

Qualitative Results. More qualitative comparisons between CityNeRF and MipNeRF (the best baseline method) are shown in Fig. 4 and Fig. 5, where we showcase scenes with various styles. All these scenarios demonstrate the superiority of CityNeRF compared to the baseline methods

	Earth-scale (Extension)	*Populated Cities* (Main Experiments)		*Places of Interest* (Extra Tests)				
	New York (56 Leonard)	New York (56 Leonard)	San Francisco (Transamerica Pyradmid)	Seattle (Space Needle))	Rome (Colosseum)	Los Angeles (Hollywood Sign)	Bilbao (Guggenheim Museum)	Paris (Centre Pompidou)
Lowest Altitude (m)	290	290	326	271	130	660	163	159
Highest Altitude (m)	6,286,000	3,389	2,962	18,091	8,225	12,642	7,260	2,710
Number of Images	830	463	455	220				

Table 1. Statistics of city scenes captured in Google Earth Studios for this supplementary.

Figure 4. *Left:* Ours CityNeRF. *Right:* Mip-NeRF baseline. Here we show more examples rendered by CityNeRF on five additional scenes across 3 scales, including *Hollywood Sign (Los Angeles)*, *Centre Pompidou (Paris)*, *Guggenheim Museum (Bilbao)*, *Colosseum (Rome), and* Space Needle (Seattle). Statistics can be found in Tab.1. ©2021 Google

across all scales. It is noted that, we can even clearly see the small size fonts on the flag in the Seattle scene.

Evolved Model Weights. With the progressive paradigm, CityNeRF fully unleashes the power of the whole frequency bands provided in the position encoding, while vanilla NeRF fails to activate high-frequency channels in PE. Fig. 6 visualizes the model weights value in these skip layers from shallow to deep layers. A clear shift from low-frequency channels to high-frequency channels in PE can be

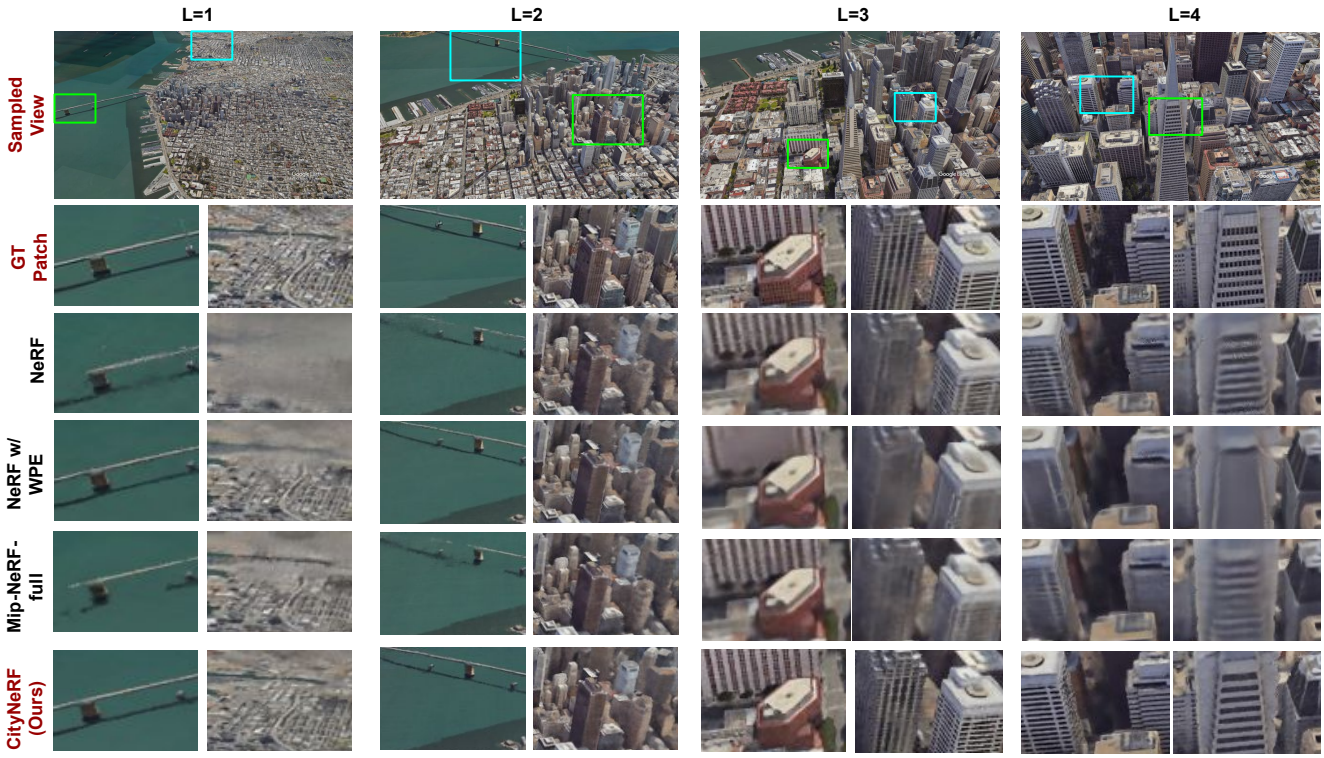

Figure 5. Qualitative comparisons between NeRF [7], NeRF w/ WPE [10], Mip-NeRF-full [2], and CityNeRF. (src: *San Francisco* scene ©2021 Google)

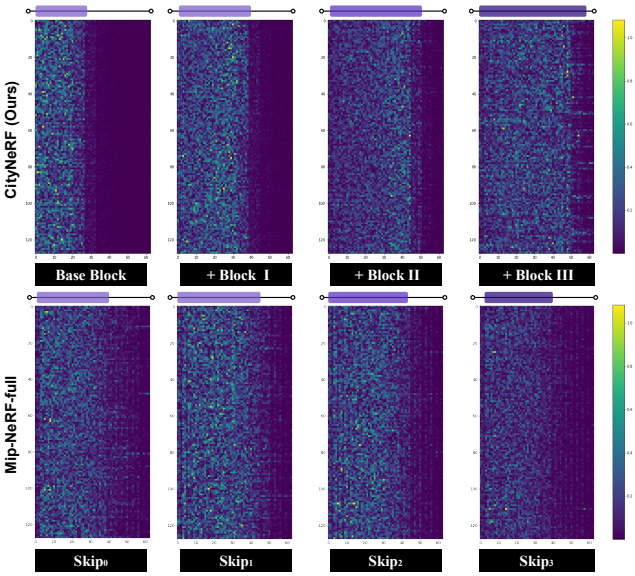

Figure 6. Comparison in model weights between CityNeRF and Mip-NeRF. The model weights are presented in value matrices, where the x-axis represents the frequency components from 2^0 to 2^{10} for (x, y, z) respectively, and the y-axis indicates different feature channels. As the training proceeds, our CityNeRF activates higher frequency Fourier features in PE to construct finer details, while vanilla NeRF fails to access these portion of PE.

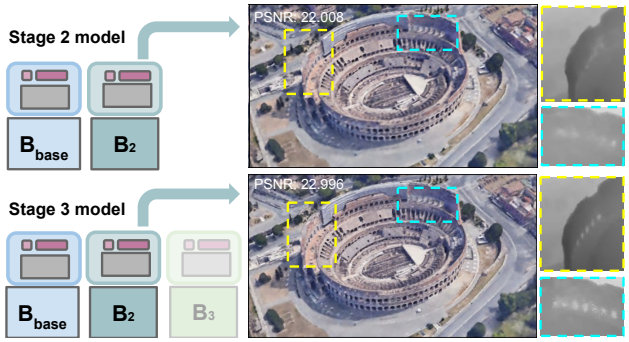

Figure 7. Illustrative samples showing how later-stage training helps the previous blocks form better geometry and textures. We show the results obtained from H_2 with training up to stage 2 and stage 3 separately. The residual connection between blocks allows supervisions from finer-detail views to guide the earlier blocks output correct geometries, with which more complex details emerged in later stages can be built upon. (src: *Rome* scene ©2021 Google)

observed in CityNeRF, while different skip layers in Mip-NeRF equally biased to the low-frequency Fourier features.

Improved Block Heads. Fig. 7 shows the later training stages can help improve or correct the predictions exit from the earlier heads, with the aid of our residual design that encourages the consistency between scales.

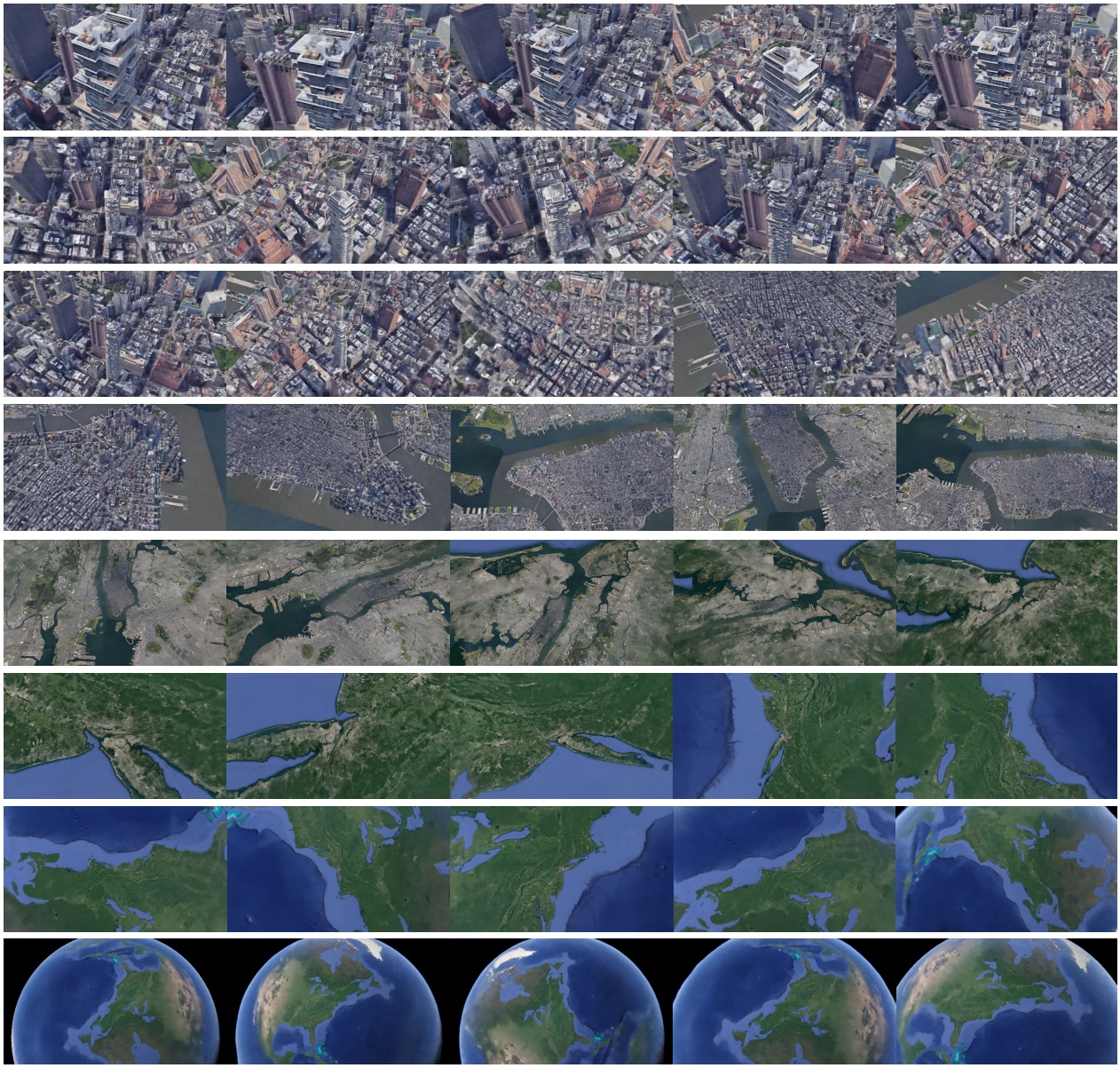

Figure 8. Rendered novel views for Earth-scale scene, trained with 5 stages.

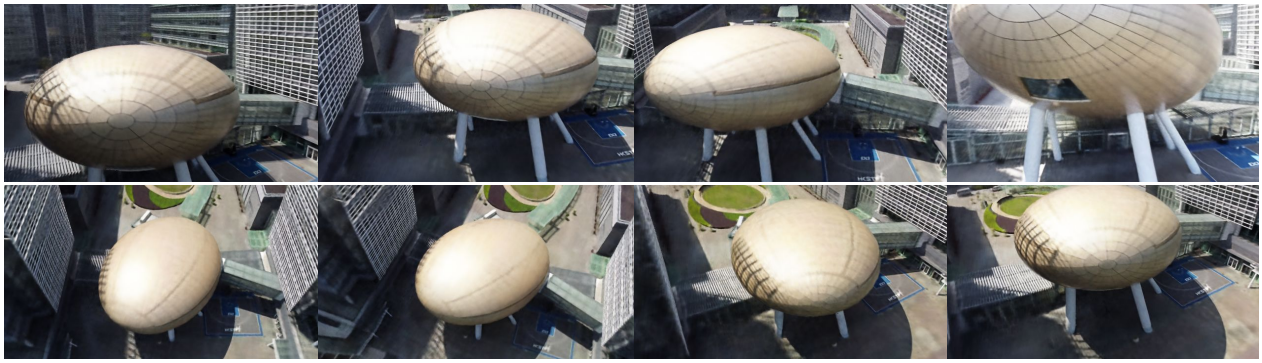

Figure 9. Rendered novel views for drone captured scenes, trained with 2 stages.

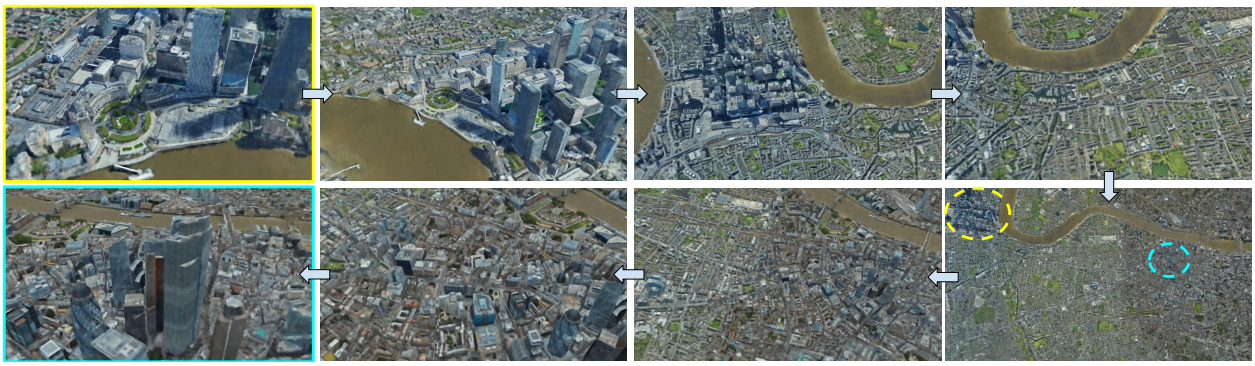

Figure 10. CityNeRF trained on a scene consisting 2 places of interested. The camera trajectory is illustrated by arrows where it starts from capturing one place at ground-level (yellow frame), and moves towards another place aloft, then dive to ground-level again (blue frame).

4. Extensions

Earth-scale NeRF. The progressive training scheme we present to CityNeRF has the potential to tackle data with arbitrary multi-scales. To test the generality of CityNeRF, we run an extreme experiment on the *earth-scale*, where the camera altitude can lift to >6,000 km and capture the entire earth in the view. We set $L_{max} = 5$ to tackle the extreme large space in camera altitude. A selected subset of rendered novel views is shown in Fig. 8, covering scales from ground to satellite. CityNeRF successfully pack such extreme scales into a single model, with high-quality preserved details across all scales. It is noted that vanilla NeRF can hardly handle this scale of scene, resulting in extremely unstable training that produce nan values constantly.

Real-world drone data. One promising application our CityNeRF can bring about is the 3D reconstruction of city scenes captured by a drone. While traditional 3D reconstruction software always requires a dense capture of drone images with enough overlap in order to successfully reconstruct the model based on feature matching, NeRF could provide a new approach to these tasks. We collect a drone footage for an auditorium building, where the frames were taken at different altitudes and angles. The images are processed by COLMAP [11] to obtain the camera poses. As shown in Fig. 9, we test CityNeRF's generality on a drone-captured scene with a simplified 2-stage training, our CityNeRF can successfully recover the accurate geometry and detailed textures both on close-up and distant views.

Multi-dive scenes. Orbit trajectories with ascending camera altitude are used as our main training data. However, our CityNeRF is well suitable for arbitrary camera trajectories where the key is the *multi-scale* characteristic. The above drone captured scenes already demonstrate one scenario where the camera motion is relatively arbitrary. We further deliberately test a *two-dive* scene captured in Google Earth Studio where we require the fine-detailed building models to cover two distant ground building complexes: one for *Canary Wharf* and the other for *Leadenhall*, both in London. The rendering results on novel views are shown in Fig. 10, where CityNeRF can also handle such scenes.

5. Discussions and Limitations

Inconsistency in Training Data. As current CityNeRF is built upon static scenes, it cannot handle inconsistency in the training data. We observed that, in Google Earth Studio [1], objects with slender geometry, such as a lightning rod, flicker as the camera pulls away. Artifacts like flickering moir patterns in the windows of skyscrapers, and differences in detail manifested as distinct square regions on the globe are also observed in the rendered images served as the ground truths[2]. Such defects lead to unstable rendering results around certain regions and bring about inconsistencies. A potential remedy is to treat it as a dynamic scene and associate each view with an appearance code that is jointly optimized as suggested in [6, 10]. Another potential limitation is on computation. The progressive strategy naturally takes longer training time, hence requires more computational resources.

Determining LOD. Each training image (and also the rendered novel view) need to be associated with a level indicator (*i.e.*, LOD), with which the model knows where to early-exit the rendering images to achieve the desired level-of-detail. When rendering a continuous novel view trajectory, though one can choose to render all the images with the highest detail level obtained from the last block, it is suggested to dynamically use the output heads from earlier blocks to alleviate the aliasing issue. Once the camera poses are known, the LOD can be easily determined based on their distances to the ground target.

Potential Misuse. The photorealistic rendering offers the possibility of abusing the content of the scene, such as removing the watermark stamped on the original image for copyright protection. When using this method, the relevant

[2]More details are discussed in Google Earth Studio: Best pratcies.

provisions of copyright protection must be observed[3].

References

[1] Google earth studio. https://earth.google.com/studio/. 1, 6

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021. 1, 2, 4

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. 2

[4] Ajay Jain, Matthew Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. *ArXiv*, abs/2104.00677, 2021. 1

[5] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv preprint arXiv:2104.06405*, 2021. 2

[6] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 6

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 4

[8] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *arXiv preprint arXiv:2103.03231*, 2021. 2

[9] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 2

[10] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2, 4, 6

[11] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 1, 6

[12] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619, 2021. 2

[13] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[3] There are also several ways to minimize these risks with the merit of NeRF. For example, the modeled scene could be rendered with new watermark by manually adding content to the scene.