We thank reviewers for the constructive comments. We will release code and data. We now address main concerns.

Synthetic results are better than real scene results (R2, R3, R4): The camera pose errors of real scene data, albeit small, caused this difference, since there are no camera pose errors for synthetic data.

Storage usage for network weights (R1, R2, R4): The storage usage for the network weights of NSVF varies from $3.2 \sim 16$MB (including around 2MB for MLPs), depending on the number of used voxels ($10 \sim 100$K). NeRF has two (coarse and fine) slightly deeper MLPs with a total storage usage around 5MB. We will add this to the revision.

Scene edtiting is not clear regarding MLPs (R1): We used the learned multi-object model which is trained with different voxel embeddings for each object but sharing the same MLPs (L224-225). We will make it clear.

Training with RGB or RGBD images? (R2): All the scenes except the ScanNet scenes are recovered from RGB images. Our method is also applicable to RGBD data, e.g. ScanNet, for which depth is used for voxel initialization and training. Fig. 16 shows a comparison of w/ and w/o voxel initialization. We will make the type of training data clear.

How voxel size affects rendering speed (R2): Large voxels used to bound a scene are likely to contain more empty space, thus leading to longer rendering time spent on evaluations in empty space.

Performance with different voxel resolutions (R3): The ablations of different voxel resolutions w/ and w/o fixing step size at different training rounds are shown in Table 2 and Table 4, respectively.

Comparion with one round of training at the final resolution (R3): Our test on *Wineholder* shows that compared with one-round training, our progressive training is faster and easier to train, uses less space and achieves better quality. The metrics (PSNR↑, SSIM↑, LPIPS↓) are: 29.77, 0.946, 0.033 (One-round) v.s. 32.04, 0.965, 0.020 (Progressive).

Experiments on large-scale scenes and dynamic scenes (R3): Table 1 (below) shows that NSVF achieves the best performance on these two tasks. For the *ScanNet* results, better represented geometry results in better rendering quality.

Table 1: Results for *Maria Sequence* (Left) and *ScanNet (one scene)* (Right). Here geometry accuracy is measured by RMSE of ground-truth depths and depths of rendered geometry. No result for NV is reported for *ScanNet* because training failed to converge.

	PSNR↑	SSIM↑	LPIPS↓
SRN	29.12	0.969	0.036
NV	33.86	0.979	0.027
NeRF	34.19	0.980	0.026
Ours	**38.92**	**0.991**	**0.010**

	RMSE↓	PSNR↑	SSIM↑	LPIPS↓
SRN	14.764	18.25	0.592	0.586
NeRF	0.681	22.99	0.620	0.369
Ours (w/o depth)	0.210	25.07	0.668	0.315
Ours (w/ depth)	**0.079**	**25.48**	**0.688**	**0.301**

Effect of early termination (R3): The quantitative metrics (PSNR↑, SSIM↑, LPIPS↓) and average rendering speed (sec/frame) on *Wineholder* of NSVF with early termination with different thresholds are shown as follows: 31.93, 0.965, 0.021, 4.0 ($\epsilon = 0.0$) v.s. 32.03, 0.965, 0.020, 2.1 ($\epsilon = 0.001$) v.s. 32.04, 0.965, 0.020, 2.0 ($\epsilon = 0.01$) v.s. 29.99, 0.947, 0.029, 1.7 ($\epsilon = 0.1$). The selection of $\epsilon = 0.01$ gives the best trade-off between quality and rendering speed.

How the initial grid resolution affects the performance (R3): Our tests show that the initial grid resolution does not affect the quality of results. We will include the experiments in the revision.

Comparison with DeepVoxels (R3): As stated in the SRN paper, SRN outperforms DeepVoxels (by the same authors). So, as treated in NeRF, we see no need to compare with DeepVoxels because our method outperforms SRN.

Eq. 2 seems incorrect (summation v.s. product) (R3): Eq. 2 is correct, because it is equivalent to the one in Mildenhall et al., based on the elementary identity $\exp(\sum_i x_i) = \prod_i \exp(x_i)$.

Describe the benefits of sparse voxel grids (R3): The benefits are described in detail throughout the paper, e.g., Line 48-51 on the benefits, Sec. 2 and Sec. 3 on the motivation and advantages of sparse voxel grids, etc.

Why NeRF is better in SSIM (R3): In fact, NeRF has worse SSIM scores than ours. In the submission we cited the SSIM scores reported in the NeRF paper for the eight NeRF's synthetic objects. After submission, we were informed by the NeRF's authors that their SSIM metric was calculated incorrectly. Now the corrected SSIM scores reported in their updated version are $\sim$0.03 lower than the original wrong scores. Thus, our method is better than NeRF in SSIM now.

NeRF results for *Steamtrain* (R4): Thanks for pointing this out. We realized after submission that we forgot the preprocessing step of scaling the two models, *Steamtrain* and *Ignatius*, into the cube of side length 2 centered as required for running NeRF code. We subsequently retrained NeRF for these two models with this preprocessing. The results have improved but still are worse than our results. The corrected quantitative metrics (PSNR↑, SSIM↑, LPIPS↓) for these two models are as follows: *Steamtrain*: 30.84, 0.966, 0.031 (NeRF) v.s. 35.13, 0.986, 0.010 (Ours); *Ignatius*: 25.43, 0.920, 0.111 (NeRF) v.s. 27.91, 0.930, 0.106 (Ours). We will correct the results of these two models in revision.

The threshold for self pruning (R4): We clarify that for ALL the experiments we set the threshold as 0.5, which works stably. Self pruning may prune incorrectly for very thin structures. We will discuss failure cases in the revision.

Other clarifications: (i) the ratio of foreground to background is based on image pixel and it can reach 1 (R2); (ii) we train all the methods with the same views (R3); (iii) "overfitting" refers to overfitting to training views (R3); (iv) rendering time is related to not only the foreground ratio but also the complexity of the object itself (R4).

We will also: (i) add missing references (R2, R3); (ii) rephrase the statement "the proposed method is 10 times faster than the state-of-the-art" (R3); (iii) summarize the experimental settings described in Appendix in the main paper (R3); (iv) improve grammar and word usage, fix typos, and rewrite unclear parts (R3).