Filtering Data to Improve NeRF for Driving Scenes

Shiva Sreeram
MIT
sasreera@mit.edu

David Baek
MIT
dbaek@mit.edu

Abstract

Autonomous driving is a complex domain with a goal in achieving robustness in a myriad of possible scenarios. One limitation, though not unique to this field, is the availability of data to be utilized in the development of rigorous self-driving processes. Our work aims to address the issue of data generation time such that the ability to construct a NeRF of a scene is less expensive in terms of compute time and memory. However, with a reduction in compute time and memory usage comes a reduction in performance. As such, we aim to pre-process the data through appropriate filters such that the key components of the scene that are vital to driving can be synthesized by NeRF and reduce the performance losses. Though small, we were able to achieve an improvement in average PSNR of the original data with the reduction in compute time and memory from 19.74 to 19.92 through our filtering pipeline.

1. Introduction

Neural Radiance Field (NeRF) [2] is a groundbreaking method of novel view synthesis, which learns the mapping from 3D location to 2D view direction, radiance, and volume density via a neural network. Using NeRF as a scene representation, the authors were able to significantly improve the rendering quality compared to the previously-dominant Convolutional Neural Network-based approach.

However, the original NeRF comes with several limitations [3], namely the slow training and rendering times. In particular, developing an efficient NeRF training pipeline that can be done in a timely manner, while not degrading performance completely, is vital to dataset generation in key fields such as driving.

In this paper, we consider an application of NeRF to autonomous driving, and present a novel filter-based data preprocessing method for NeRF with driving scenes. The filters we consider are a manual sharpening filter, Gaussian blur, Gaussian sharpening, and Laplacian sharpening.

This paper is organized as follows: In Section 2, we describe the related works. Section 3 explains our methods for processing scenes with NeRF, our rendering pipeline, and evaluation process. In Section 4, we summarize the experiments we have conducted and analyze the results of said experimentation. Section 5 concludes this progress report by providing an overview of our work and contributions.

2. Related Works

2.1. Neural Radiance Fields

NeRF has been utilized extensively in scene reconstruction, particularly in driving. NeRF for street views (S-NeRF) [1] has shown remarkable success in synthesizing of large-scale backgrounds and moving vehicles in the foreground. S-NeRF also utilizes noisy and sparse LiDAR points to improve the training and learning processes of the pipeline. Experiments were run using existing datasets for driving, such as nuScenes [4], that provide RGB image data alongside the LiDAR feed in order to complete the process.

S-NeRF builds upon the aforementioned work in NeRF [2] as well as work in large-scale NeRF and depth supervised NeRF. Large-scale NeRFs, such as mip-NeRF 360 [5], aim to address the synthesis of large-scale outdoor scenes through handling "unbounded" scenes in which the camera can point at a direction where objects can exist far into the horizon (as would be the case when looking down a road). Depth supervised NeRF [6] methods utilize depth information to supervise the training process of NeRF. Urban-NeRF [7] introduces the idea of using high quality LiDAR sweeps for the depth information.

Our work aims to build upon past implementations through pre-processing not only in depth information, but also in RGB space, to improve model performance.

2.2. Image Deblurring

With NeRF, there has been work to handle blurry images in Deblur-NeRF [8], which utilizes a Deformable Sparse Kernel (DSK) that is jointly optimized with the NeRF to model spatially-varying blur kernels. This process was motivated by image deblurring work aiming to recover a sharp

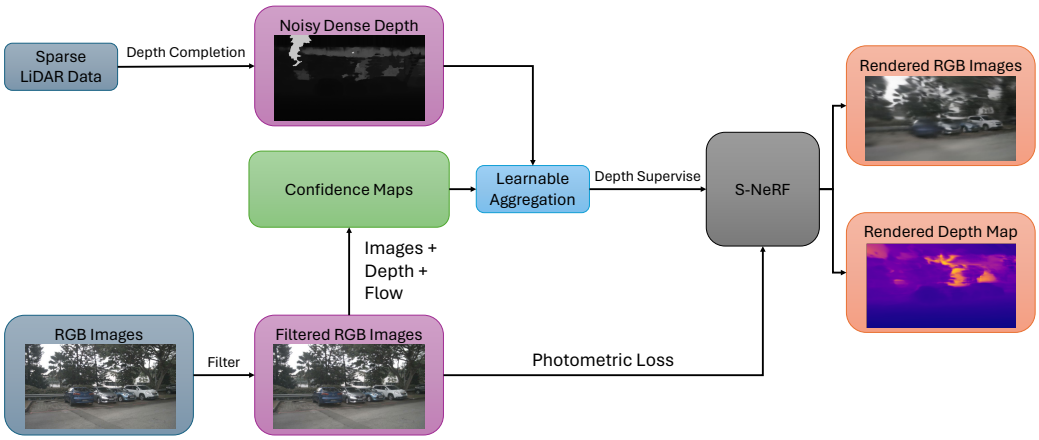

Figure 1. Pipeline overview. Following a similar structure to the S-NeRF framework [1], we provide filtered images to train the S-NeRF model to yield the reconstructed images and depth. This process involved a significantly lower training time by reducing the number of training iterations and dimension of hidden layers as detailed in Section 4.

image from a blurry one. Blur kernels and various deblurring techniques have been extensively studied [9].

Our aim in this work is to utilize a pre-processing stage to apply blurring and sharpening filters while applied to the autonomous driving domain, in order to improve the reconstruction quality.

3. Methods

We use S-NeRF [1] implementation and nuScenes [4], a public autonomous driving scene dataset, for the evaluation. We use a variety of filters to preprocess images from the nuScenes dataset before feeding them through S-NeRF and generating the reconstruction. In this section, we detail the stages of our pipeline shown in Figure 1.

3.1. Depth Completion

To work with S-NeRF, the following stages are needed: a data loading stage to appropriately store the data, depth completion to clean up the sparse LiDAR data, and finally the training process. Let us expand on the depth completion stage, providing an explanation on the details in the paper and what the code is performing. This involves using the pre-trained SeparableFlow [10] and Non-Local Spatial Propagation Network (NLSPN) [11] models. Depth completion in itself is necessary to take depth information captured from a LiDAR and convert it to pixel space in the frame of the camera view. The LiDAR data provided by nuScenes is too sparse for NLSPN as it is 32-channel when NLSPN performs best with 64-channel data. As such, 5 to 10 frames of LiDAR data are accumulated together to give a denser representation. However, this accumulation contains errors from a variety of sources such as dynamic objects

Figure 2. (Left) Example RGB image from a scene from nuScenes as well as the (right) resultant "depth completion" to clean up the sparse LiDAR information. Data of this format is fed to S-NeRF.

(i.e. pedestrians), occlusions, etc. SeparableFlow is utilized to remove outliers by computing the optical flow from one RGB image to the next frame. Then the LiDAR flow is obtained by reprojecting LiDAR points to the neighbor image plane. These flows are compared to locate the outliers and they are removed appropriately to generate the final noisy dense depth map with NLSPN. The confidence level in the process of removing outliers is included in the confidence maps (one of the aspects shown in Figure 1). An example of depth completion is shown in Figure 2.

3.2. Filtering

Before we feed the image to NeRF training, we employ the following four different types of filters to preprocess the image: For an image f and filter g, each filter computes

- Manual: $[f * g_M]$, $g_M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$,
- Laplacian sharpening: $[f * g_L]$, $g_L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$,

- Gaussian sharpening: $[1.5f - 0.5(f * g_G)]$,
- Gaussian Blurring: $[f * g_G],\ g_G = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$,

where $*$ denotes convolution, k denotes the size of the Gaussian filter g_G, and σ denotes its standard deviation. The default value we used was $\sigma = 3$. In each experiment, we only control one of k or σ – the other was implicitly computed by OpenCV's built-in function $\sigma = 0.3[0.5(k-1)-1]+0.8$.

These filters were chosen for a few reasons. The Laplacian was chosen not for its ability to obtain an accurate color, but in an attempt to focus on the edges of objects in the scene such that the shapes could hopefully be better reconstructed. The other types of sharpening filters were chosen with the aim of improving clarity of the elements of the scene to achieve a higher-quality reconstruction.

The blurring, or smoothing, was chosen to test if removing some of the finer detail would assist with the reconstruction process, especially since the source of the LiDAR data used was more sparse and has been made more dense in a "noisy" fashion.

3.3. S-NeRF Pipeline

We closely follow the S-NeRF [1] pipeline given in the original work. We will showcase our understanding of the process in this subsection. Note that in general, a NeRF will represent a scene as a neural radiance field where it learns the mapping of a 3D position and viewing direction to a color with its differential density. Then, following training, the rendering is conducted by determining the color through the camera ray via the origin of the ray, the accumulated transmittance along the ray, and the corresponding color and density of a sampled point.

S-NeRF itself involves a few nuances. Given the framework of driving scenes, there is very little overlap in camera views, requiring camera pose processing. The camera parameters are obtained via simultaneous localization and mapping (SLAM) and the IMU sensor of the autonomous cars and are refined with a pose refinement network. Note the process for moving vehicles in the foreground is different, but we focused on background reconstruction for this project. Also, given the nature of the scenes in the dataset, the range is bounded before position encoding.

S-NeRF also uses depth supervision as discussed in Section 2, where the process of depth completion and obtaining confidence maps was detailed in the Depth Completion subsection. The loss functions used in S-NeRF utilize the standard RGB loss of NeRF [2], confidence-conditioned depth loss from depth supervision, and a smoothness constraint to handle large depth variance. However, as stated in the repository, the smoothness constraint is bugged and unused.

Our project employs this pipeline's strengths in reconstructing driving scenes, with a key change in dropping the number of iterations and hidden layers to massively reduce training time (as will be discussed in Section 4), in combination with our filters.

3.4. Evaluation Metrics

Following S-NeRF [1], we report the following two image similarity metrics to evaluate the quality of reconstructions: PSNR (Peak signal-to-noise ratio) and SSIM (Structural similarity index measure). PSNR is defined as follows:

$$\text{PSNR} = -10\log_{10}(\text{MSE}), \tag{1}$$

where MSE is the mean squared error between the original image and the reconstructed image, where pixels are normalized to be within the range $[0, 1]$. Higher PSNR means better reconstruction.

Another useful image similarity metric is SSIM (Structural similarity index measure) [12]. *SSIM index* is a combination (multiplication) of three comparative measures: luminance (l), contrast (c), and structure (s) comparisons between two images x and y. Each measure is defined as

$$l(x,y) = \frac{2\mu_x\mu_y + \alpha_1}{\mu_x^2 + \mu_y^2 + \alpha_1},\ c(x,y) = \frac{2\sigma_x\sigma_y + \alpha_2}{\sigma_x^2 + \sigma_y^2 + \alpha_2}, \tag{2}$$

$$s(x,y) = \frac{\sigma_{xy} + \alpha_2/2}{\sigma_x\sigma_y + \alpha_2/2}, \tag{3}$$

where μ_i and σ_i is the average and standard deviation of pixel values in image i, $\alpha_1 = (0.01L)^2$ and $\alpha_2 = (0.03L)^2$ are constants, where $L = 1$ is the range of normalized pixel values. SSIM is computed by taking the mean of SSIM indices within each local window of size 11×11 with Gaussian weighted pixel values. Multi–scale SSIM (MSSSIM) is computed by multiplying SSIM values of images at different scales, which are obtained by consecutive average pooling of the image. SSIM value is in the range $[-1, 1]$, and higher SSIM indicates higher similarity between images.

SSIM is a better indicator of *perceptual* reconstruction quality than PSNR, as it incorporates the local structural similarity information as opposed to PSNR.

In this paper, we report both PSNR and MSSSIM of the reconstructions. Note that these metrics are computed by comparing reconstructions with the *original* images, not the preprocessed images.

4. Experimental Results

Following the integration of the nuScenes dataset with the S-NeRF repository and compilation of depth completion to clean up the sparse LiDAR data, we perform experiments using a variety of filters. In this section, we discuss our experimental setup and the types of experiments conducted, including analysis of the results.

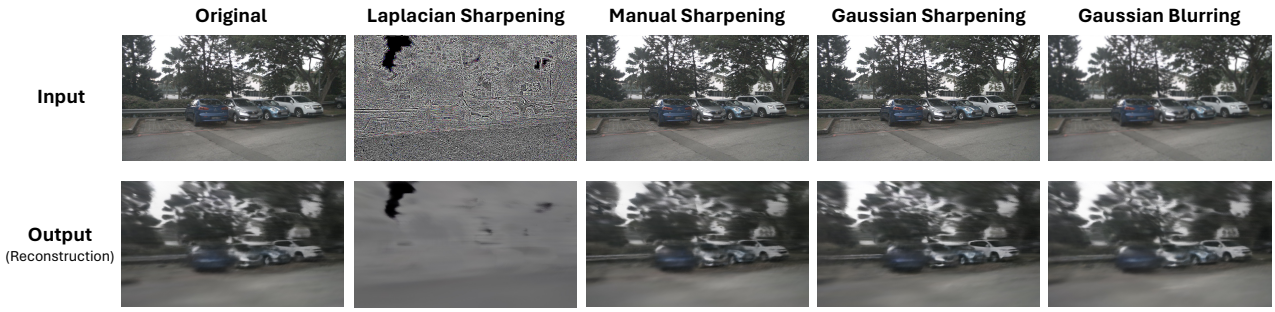

Figure 3. Comparison between input and reconstructed images for different types of input pre-processing strategies.

Method	PSNR (mean)	SSIM (mean)
Original	19.74	0.617
Laplacian Sharpening	11.86	0.475
Manual Sharpening	19.46	0.604
Gaussian Sharpening	19.54	0.611
Gaussian Blurring (σ=3)	19.63	0.614
Gaussian Blurring (k=3)	19.79	0.618
Gaussian Blurring (k=5)	19.81	0.622
Gaussian Blurring (k=7)	19.90	0.624
Gaussian Blurring (k=9)	**19.92**	**0.626**

Table 1. Image similarity metrics for the reconstruction with reduced hyperparameters. Varying the filter size of the Gaussian blur filter led to marginally better results than the reconstruction using the original, unfiltered images. Note that these metrics are computed by comparing reconstructions with the *original* images, not the preprocessed images.

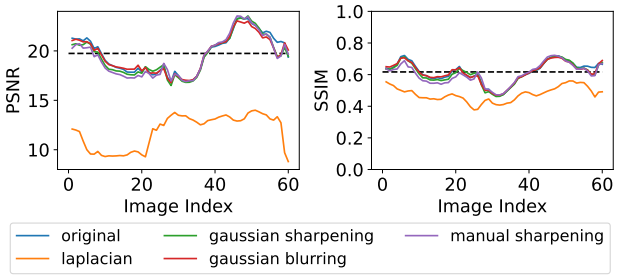

Figure 4. (Left) PSNR, and (right) SSIM as a function of image index in the scene for different types of filters. Black dashed line indicates the average value of the original reconstruction, without any input preprocessing.

4.1. Experimental Setup

We trained locally with a RTX 3090 and on a cluster with Tesla V100, with 1600x900 images for 100K iterations and 128D hidden layers, which took $\approx$ 15 hours to train, and 1 hour to evaluate a scene of 60 images, using the same seed as the original work. In said original work, the hidden layers were 1024D and ran for 200K iterations. The former had to be reduced for the code to even run with the memory restrictions we had, and the latter was reduced in order to run in a somewhat reasonable amount of time for multiple experiments. There were some errors in the evaluation script regarding loading camera intrinsics from the dataset and issues with semantic logits when running the function for rendering the image that needed to be fixed. We have also written an additional script for computing SSIM, which was not readily available in the public repository.

4.2. Applying Different Filters

We have performed filtering for a given driving scene in the dataset. Based on the comment of the author of the original S-NeRF work on issue 13 on the GitHub repository (which also discusses some of the issues in the evaluation script mentioned in the prior subsection), this particular scene is quite complex for the model, making it an ideal candidate for our experimentation.

The first row of Figure 3 shows the filtered images using four different types of filters described in Section 3. The sharpened images are quite difficult to tell apart from the original but note that when sufficiently zoomed into the images, the road texture is less blurry. Gaussian blurred image has a noticeable difference compared to the original image.

The second row of Figure 3 shows the resultant reconstructions of utilizing these filters. The average image similarity metrics is given in Table 1, where these averages were computed across the scene where the PSNR and SSIM are given in Figure 4. Note that with the change of hyperparameters, the reconstruction using the original images is not high quality. Let us now analyze the reconstructions after training with the filtered images. While we did not expect high performance from the reconstructions with Laplacian sharpening due to the color, it was unfortunate to see that the edges highlighted in the filtered image yielded no assistance in the clarity of the shapes in the scene. In Figure 4, it was curious to see that the PSNR was peaking at image

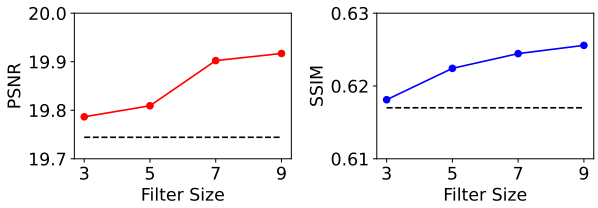

Figure 5. Average (left) PSNR, and (right) SSIM as a function of filter size for Gaussian blurred inputs. Black dashed line indicates the average value of the original constructed image, without any input preprocessing.

indices where the other filters were at their worst, but this PSNR was still quite low.

We were surprised to see that the other types of sharpening did not boost performance as we expected the additional "clarity" to reduce how "fuzzy" the reconstruction was. However, closely inspecting the reconstructed Gaussian sharpening image may provide the source of the issue. Looking at the reconstructed silver car (second car from the left), we see that the front bumper has awkwardly merged with the tire of the car. It seems by trying to extract clarity from the image, the filter has inadvertently filled in nonsensical details into the image that led to poorer reconstruction.

The Gaussian blur had the best performance of the filter types but was still slightly worse than the original. Note though, that in Figure 4, the metrics for Gaussian blurring very closely followed the original until about the last quarter of image indices, more so than the other filter types. As such, we felt this required further experimentation, as we will detail in the next subsection.

4.3. Gaussian Blur Filter Exploration

Given the aforementioned results, a deeper exploration into the Gaussian blur was necessary. We felt that one of the limiting factors of the filters we used prior was the smaller kernel size given the high resolution of the images and as such, we conducted an experiment in varying the filter size. The plots in Figure 5 (as well the details in Table 1) showcase the average performance after applying these filters and we can see that while marginal, they are actually higher than the original reconstruction! Not only that, but the performance was monotonically increasing with filter size. Looking to Figure 6, we can see features such as the arrow marking on the road having more clarity with our approach. As to the potential source of this improvement, a likely candidate is one we touched on earlier, that the depth information was obtained by accumulating data across 5 to 10 frames and as such, a blurring or smoothing process including more surrounding pixels may play better with the depth completion process. Regardless, this notable result may have potential for further exploration as to the source

Figure 6. (Left) Original reconstruction (right) reconstruction after training with data filtered by a k=9 Gaussian blur kernel.

and whether it can improve further.

Note that with a PSNR of 19.92, we may have improved upon the original reconstruction with the same model hyperparameters, but compared to rendering of background scenes in the original work [1], where more iterations were used and higher dimensional hidden layers, they achieved a PSNR of 25.06. Given the sheer difference in compute time and memory however, the result from this project was not expected to compete, but perhaps the methods used here can be applied to the original work to achieve greater results.

5. Conclusion

We have explored various input pre-processing strategies to improve the reconstruction quality of S-NeRF. It was quite surprising to discover that Gaussian blurring input images could improve the quality of reconstruction: the larger the filter's size, the larger the improvement. While we modified some hyperparameters to make training time manageable, we believe that combining our image pre-processing strategies with the authors' original set of hyperparameters could lead to an even better reconstruction. Moreover, we believe that the delta of PSNR/SSIM improvement will plateau at some point as the filter size becomes larger, since too much blurring, at some point, will lead to losing important information about the input scene. This is a direction that is worth exploring for future works. Overall, we believe this is an exciting area for image reconstruction that can assist autonomous driving which involved techniques that we learned throughout this course.

Contributions

Our contributions were split accordingly: Shiva gathered the nuScenes dataset to run the depth completion stage and cleaned up the evaluation script in the repository. David set up the pipeline for filtering images and the SSIM evaluation. We worked together to get the S-NeRF framework running on our respective computing sources. In terms of experimentation, David ran the experiments for the variety of filter types and Shiva ran the experiments for the Gaussian blur filter experimentation. The analysis of the experimental results and writing the report were done together.

References

[1] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. In *ICLR 2023*, 2023. 1, 2, 3, 5

[2] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 3

[3] Frank Dellaert and Lin Yen-Chen. Neural volume rendering: Nerf and beyond. *arXiv preprint arXiv:2101.05204*, 2020. 1

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 1

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 1

[7] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022. 1

[8] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-nerf: Neural radiance fields from blurry images. *arXiv preprint arXiv:2111.14292*, 2021. 1

[9] Ruxin Wang and Dacheng Tao. Recent progress in image deblurring. *ArXiv*, abs/1409.6838, 2014. 2

[10] Feihu Zhang, Oliver J. Woodford, Victor Adrian Prisacariu, and Philip H.S. Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10807–10817, 2021. 2

[11] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *Proc. of European Conference on Computer Vision (ECCV)*, 2020. 2

[12] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 3