Project 2 - Building Built in Minutes- SfM and NeRF

Karthik Mundanad
Robotics Engineering Department
Worcester Polytechnic Institute
Email: krmundanad@wpi.edu

Kushagra Srivastava
Robotics Engineering Department
Worcester Polytechnic Institute
Email: ksrivastava1@wpi.edu

***Abstract*—This report presents our implementation of a sparse 3D reconstruction framework for a given set of images using multi-view geometry. We present a detailed analysis of all the major steps involved in Structure from Motion qualitatively and quantitatively. For the second phase of this report, we implemented Neural Radiance Fields (NeRF) and conducted thorough testing on synthetic and custom datasets.**

I. Phase 1: Structure from Motion (SfM)

Our task was to construct a sparse 3D point cloud given a finite set of images. Our framework comprises 7 steps: (i) Estimation of the fundamental matrix using a given set of matched features, (ii) Estimation of the essential matrix, (iii) Camera pose estimation from the essential matrix, (iv) Linear and non-linear triangulation subjected to cheriality constraints, (v) Adding n^{th} image using perspective-n-point, (vi) Global optimization using bundle adjustment.

A. Estimation of the fundamental matrix using a given set of matched features

The first step in our framework is to estimate the fundamental matrix given a set of matched features based on the epipolar constraints using RANSAC. We found that using the 7-point algorithm instead of the 8-point algorithm is more robust since the probability of no outliers is exponential in the size of the sample set (see Section 11.6 in [1]).

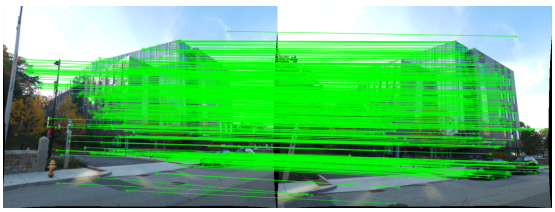
Let $x_j \in R^{n\times3\times1}$ and $x_i \in R^{n\times1\times3}$ be matrices of homogenized matched feature image coordinates for i^{th} and j^{th} image respectively. Our goal is to find a solution for the fundamental matrix, $F \in R^{3\times3}$, such that Equation 1 is satisfied.

$$x_j^T F x_i = 0 \tag{1}$$

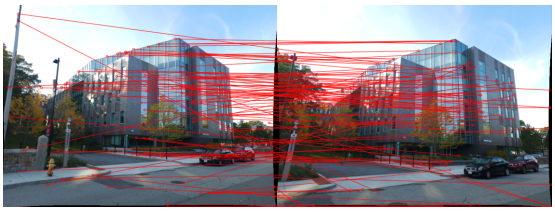
This equation gives rise to a set of equations of the form $Af = 0$, where $f \in R^{9\times1}$ is a vector of all entries in F and A is a function of the matched image coordinates for the image pair. It is possible to solve for f if the rank of A is 7 by making use of the singularity constraint. The solution of $Af = 0$, is in the form

$$\alpha F_1 + (1-\alpha)F_2 \tag{2}$$

where α is a scalar variable. The matrices F_1 and F_2 are obtained by solving for the right null space of A. Using the constraint $det(F) = 0$ which implies $det(\alpha F_1 + (1-\alpha)F_2) = 0$. Since F_1 and F_2 are known, this leads to a cubic equation in α. There will be 1 or 3 real solutions (7 points and 2 camera centers form a quadric, if it is a ruled quadric, there will be 3 real solutions), and substituting it back in Equation 2 will give us the solution for the fundamental matrix. In the case of 3 real solutions, we performed RANSAC on all the candidate fundamental matrices and selected the one with the highest number of inliers obtained. We calculated fundamental matrices for each image pair to estimate inlier feature pairs. These inliers were then used as inputs to all the algorithms discussed below. Please note that we took inputs from online resources to implement our data loader.



(a) Inliers



(b) Outliers

Fig. 1: RANSAC was used to detect inliers and outliers subjected to epipolar constraints for a given set of matched features.

B. Estimation of the essential matrix

Since the camera calibration matrix, K, was given, the essential matrix, E, can be calculated using Equation 4. The essential matrix was calculated using singular value decomposition (SVD) followed by rank reduction.

$$E = K^T F K \tag{3}$$

$$E = U D V^T \tag{4}$$

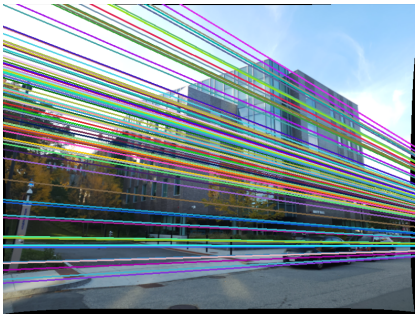


Fig. 2: Epilines for the first image features plotted on the second image.

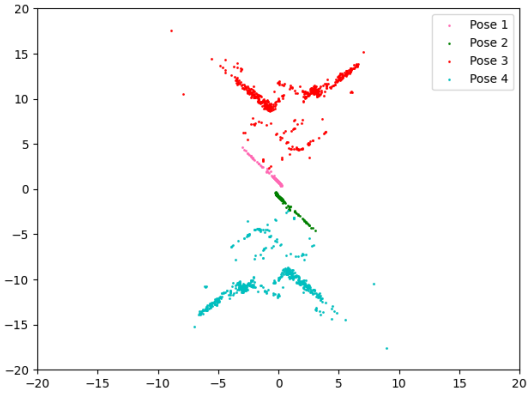


Fig. 3: Triangulated world points for all camera poses obtained after SVD of E.

C. Camera pose estimation from the essential matrix

After calculating the essential matrix, there will be 4 possible solutions for the camera pose (2 solutions for R and C).

$$C = \pm U_3 \tag{5}$$

$$R = U \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\tag{7}$$

where U_3 is the third column of U. It is important to ensure that $R \in SO(3)$. Hence, if $det(R) < 0$, then $R = -R$ and $C = -C$.

D. Linear and non-linear triangulation subjected to cheriality constraints

Since there are 4 camera poses, the next step is to linearly triangulate all the feature points and evaluate the cheriality constraint which states that the world points must be in front of the image plane for both camera poses (depth > 0). Figure 3 shows a 2D plot (Y=0) of the triangulated points using all the candidate camera poses. To rule out three camera poses, we selected the camera pose that had the most number of features satisfying Equation 8.

Fig. 4: Improvement after non-linear triangulation

Fig. 5: Improvement in feature projection after non-linear PnP

$$r_3^T (X - C) > 0 \tag{8}$$

where r_3 is the third column of the candidate rotation matrix, C is the candidate translation vector and the world point X was calculated using linear triangulation. Since linear triangulation has no geometric meaning, we used non-linear optimization to minimize the reprojection error which improved the world point estimates. Figure 4 shows the refinement in feature locations after non-linear optimization.

E. Adding n^{th} image using perspective-n-point

To estimate n^{th} camera pose, we found feature points present in the first two images and the n^{th} image, and the corresponding world points obtained after non-linear triangulation. Since the intrinsic parameters, K, are known, we solved the following system of equations using SVD to obtain R and C.

$$\lambda x = K \begin{bmatrix} R & C \end{bmatrix} X \tag{9}$$

where x are the feature points, X are the corresponding world points and λ is the scaling factor. The value of R was corrected to ensure that it was a valid rotation matrix. This linear estimate acted as an initial guess for the non-linear optimization framework using reprojection error. Figure

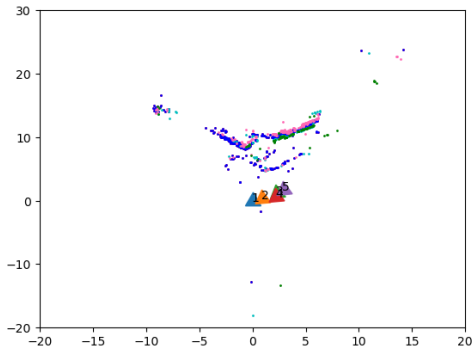


Fig. 6: Point cloud obtained after estimating world points using the first two images and registering all the other images using PnP.

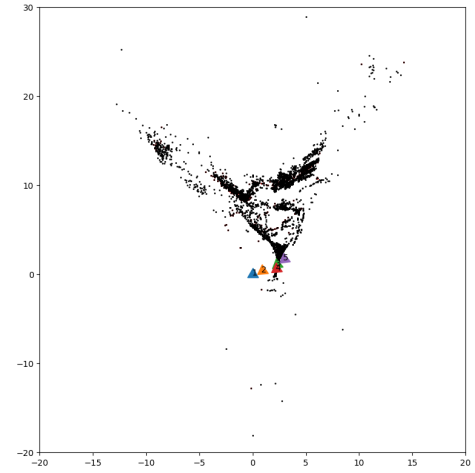


Fig. 7: Point Cloud including all the points from the matches file using PnP poses.

5 shows the improvement in feature projection after non-linear optimization. We perform this registration step for all the 3 images. The resulting point cloud and the estimated camera poses are illustrated in Figure 6.

	Mean Reprojection Error
Linear Triangulation	87.402
Non-Linear Triangulation	66.731
Linear Perspective-n-Point	88.609
Non-Linear Perspective-n-Point	9.997

TABLE I: We report the reprojection error averaged over all the images.

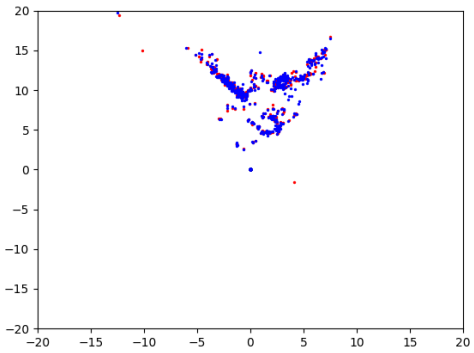


Fig. 8: Point cloud correction before and after bundle Adjustment. Red indicates before bundle adjustment and blue indicates after bundle adjustment for the 1st matching file.

F. Global optimization using bundle adjustment.

Bundle Adjustment was performed to optimize both the triangulated world points and camera poses. To facilitate this, we constructed a visibility matrix and generated a sparse matrix for **scipy.optimize.least_squares**. The results of point cloud triangulation using the first matching file were presented, revealing limited optimization potential due to the small number of points. However, a notable total shift of 23.30 in the L2 norm between the points was observed before and after bundle adjustment. This is illustrated by Figure 8.

(a) Lego Dataset (b) Ship Dataset

Fig. 9: Ground Truth Samples for Lego and Ship Dataset

II. Phase 2: Neural Radiance Fields (NeRF)

In this section, we present our implementation of NeRF [2] and the subsequent results obtained on synthetic and custom datasets.

A. Implementation Details

For the scope of this project, we implemented two versions of NeRFs: (i) Tiny NeRF and (ii) Original NeRF as in [2]. The network architectures are illustrated in Figures 11 and 12. Hyperparameters for both networks are reported in Table II

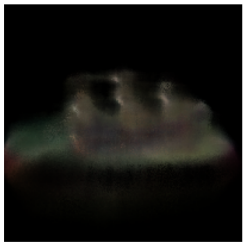


Fig. 10: Case in where our model does not perform well

Network	Batch Size	Learning Rate	Iterations
Tiny NeRF	4096	e^{-4}	4000
NeRF	128000	e^{-4}	4000

TABLE II: Hyperparameters

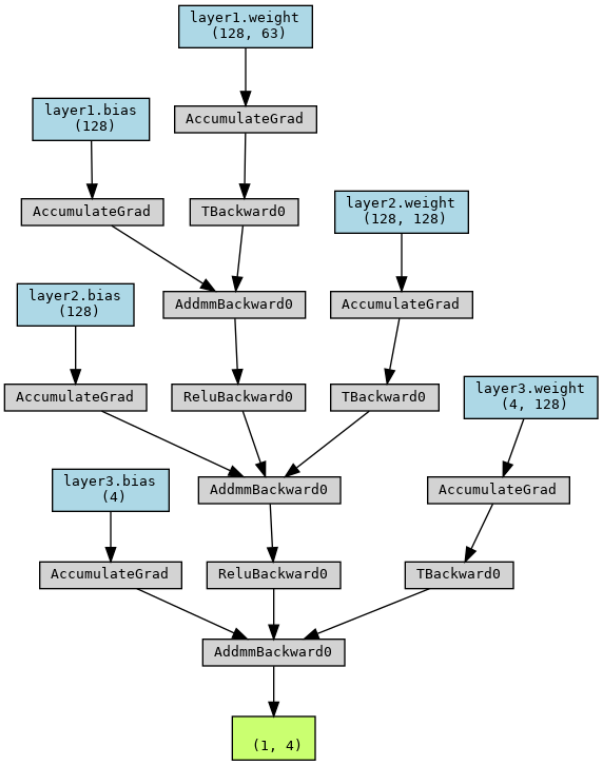


Fig. 11: Tiny NeRF Architecture

	Ship 100	Ship 200	Lego 100	Lego 200	Custom Dataset
PSNR	18.562	19.214	16.542	17.854	12.325
SSIM	0.751	0.721	0.612	0.627	0.316

TABLE III: PSNR and SSIM values for Tiny NeRF. Here Ship/Lego N represents $N \times N$ image dimension

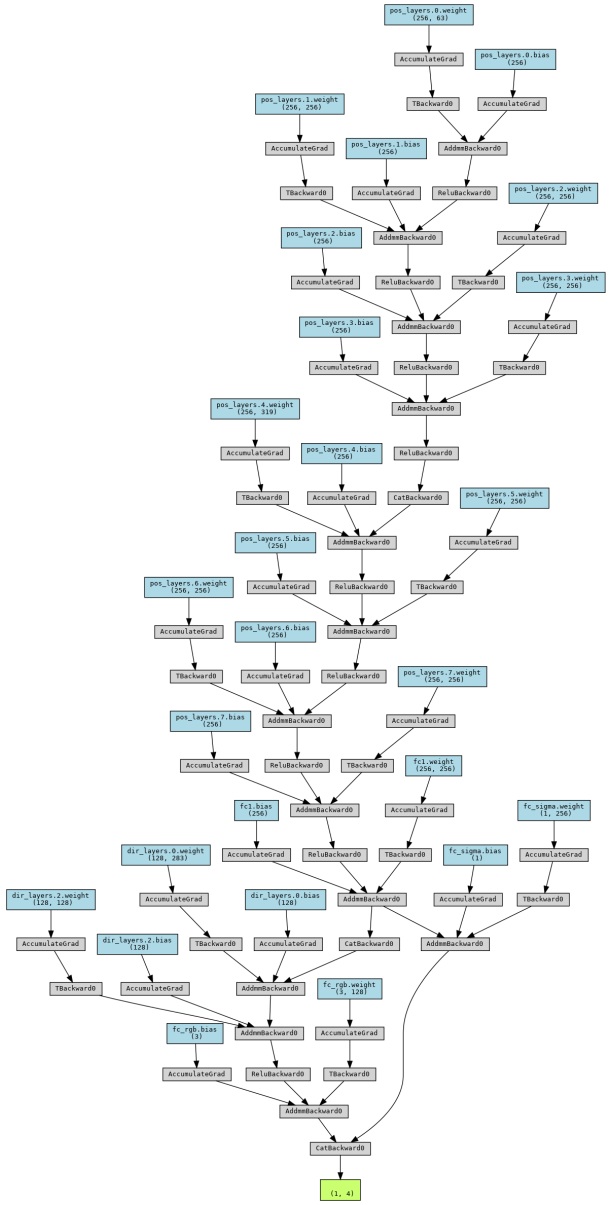


Fig. 12: NeRF Architecture

B. Results

- **Synthetic Datasets** We tested the performance of our network on the Ship and Lego Datasets for Tiny NeRF and the Lego Dataset for NeRF. We were unable to generate good results using NeRF for the Ship dataset. The results can be visualized in Figures 14 and 16. The ground truth for the corresponding images can be found in Figure 9. The PSNR and SSIM values for the corresponding Tiny NeRF experiments can be found in Table III. In particular, we experimented with two sets of image dimensions. The NeRF model as given in the paper was implemented for lego and ship datasets but it was heavily dependent on weight initialization. The result for 4000 iterations is shown in Figure **??**. Note that the

Fig. 13: Custom Dataset

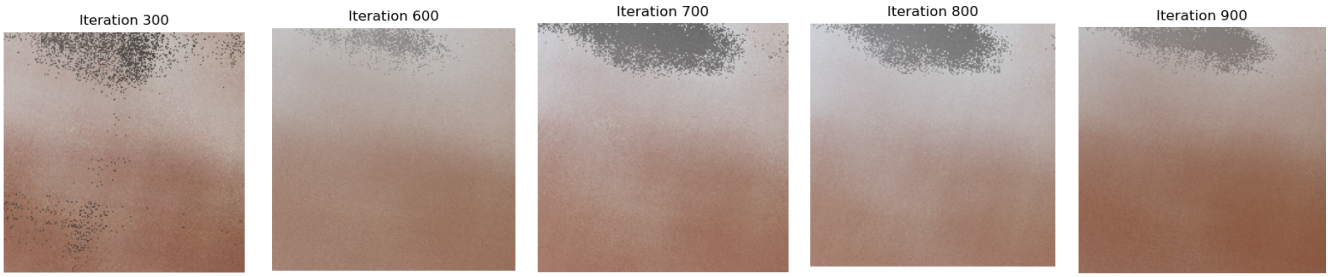


Fig. 14: Rendered Image Sample generated using Tiny NeRF for Custom Dataset

results are not drastically different from the Tiny NeRF model but running it on a higher number of iterations will result in better results. The results for the ship dataset is not included as they were not satisfactory

- **Custom Dataset** Our Custom Dataset samples, illustrated in Figure 13, were sampled from a video captured using a OnePlus 9R. A sparse 3D point cloud was constructed using COLMAP and the obtained poses were converted into the format acceptable by our model using NeRFStudio. We were unable to get good outputs on our custom dataset using Tiny NeRF. The reason for that could be: (i) Non-uniform background, (ii) Small area occupied by the subject red car in the image, (iii) Reflection from wood surface. The corresponding rendering outputs can be found in Figure 14. Our dataset is available here

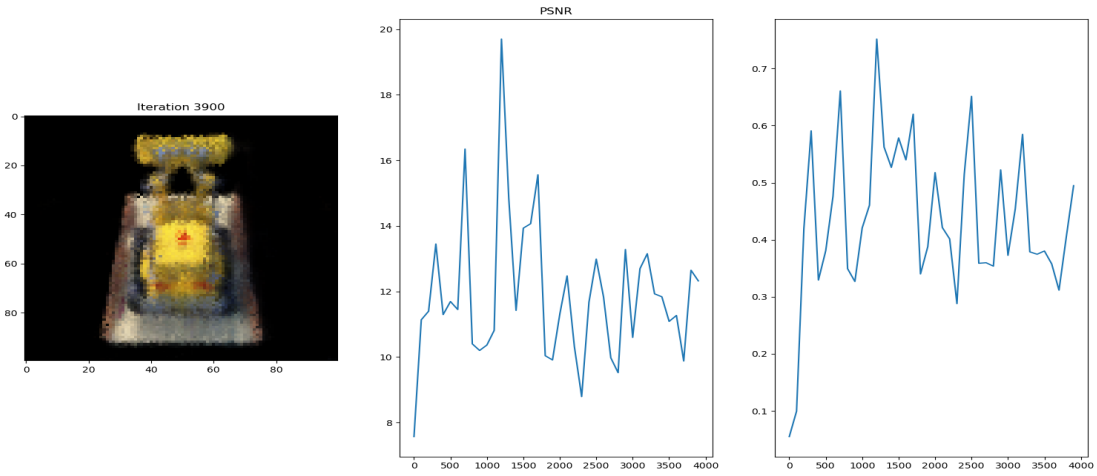
C. Challenges

The NeRF model as shown in the paper was heavily affected by weight initialization in our case. This was because of to large number of 0 valued pixels being passed in the model as rays. The solution to this is to stack all the rays from all the images and randomly pass the rays to the model. The network also took a lot of time to train and the batch size for the model had to be fine-tuned depending upon the resolution of the image. We intend to run this model for a higher number of iterations to check the results. Even for the Tiny NeRF model, the image needed to be highly synthetic for the model to learn satisfactorily. This is indicated by our tests on the custom dataset where a large number of iterations are required. The model is not robust to illumination or reflection as indicated by Figure 10. In this view, the ship was not well illuminated, and hence the output is not up to the mark.

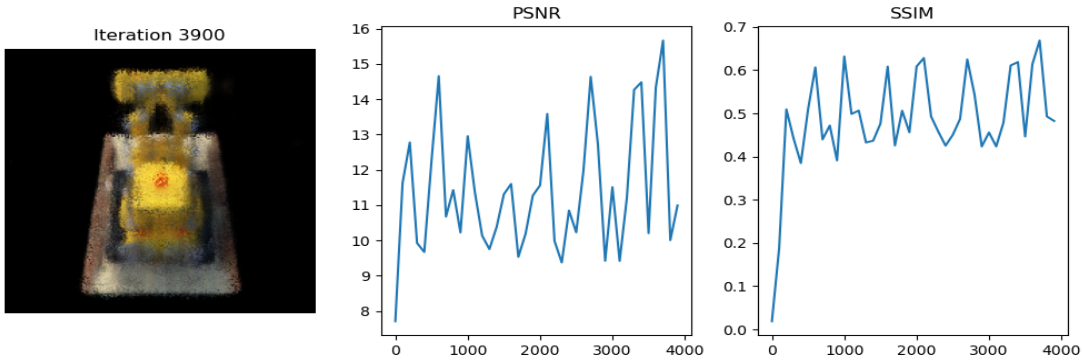
References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

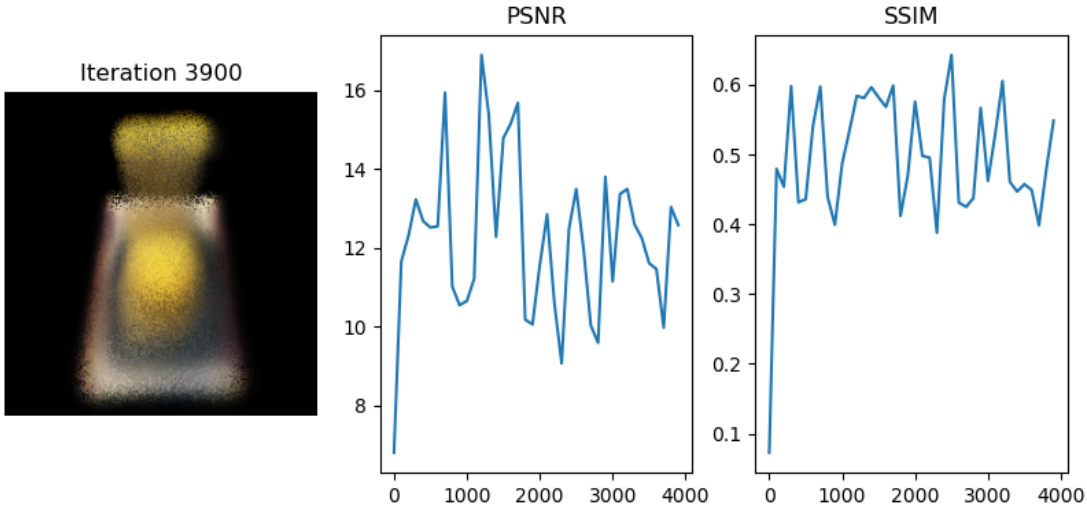
[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934



(a) Tiny NeRF Lego Output for 100×100 image



(b) Tiny NeRF Lego Output for 200×200 image



(c) Tiny NeRF Lego Output for 200×200 image without positional encoding

Fig. 14: Results for Lego Dataset

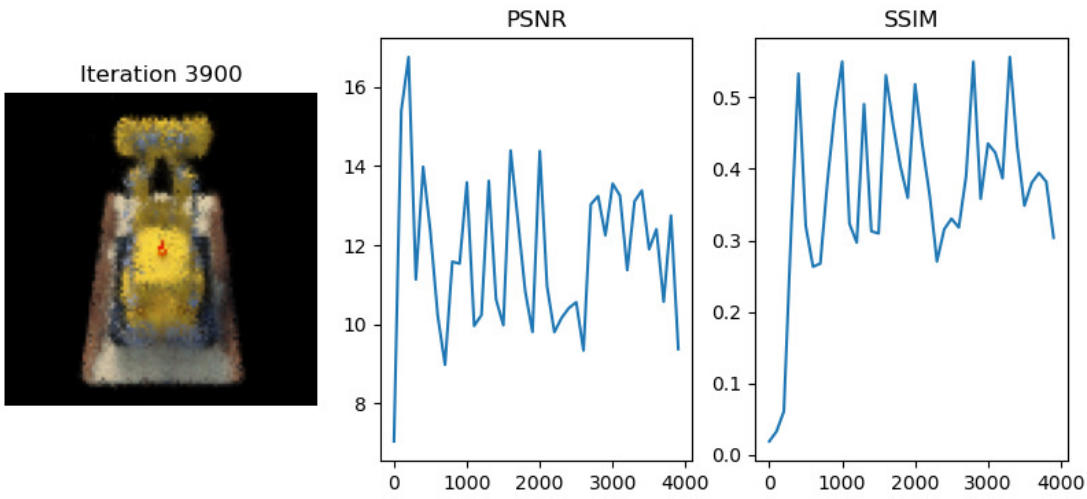
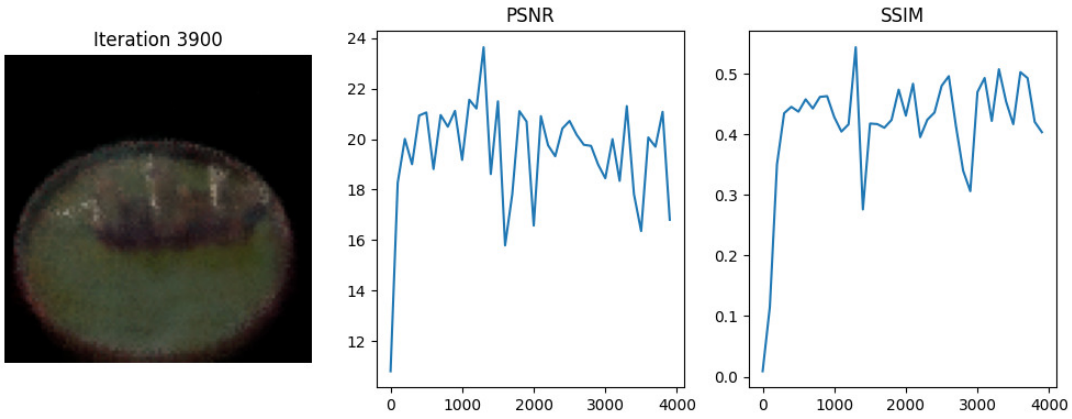
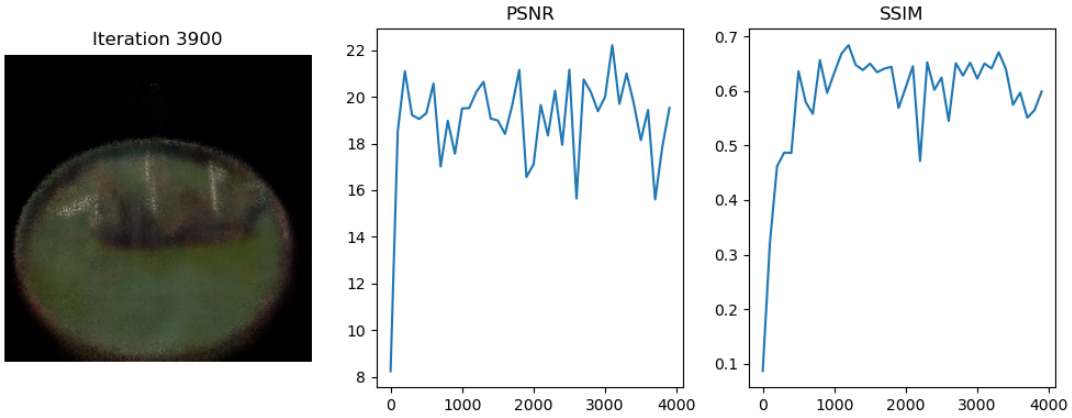


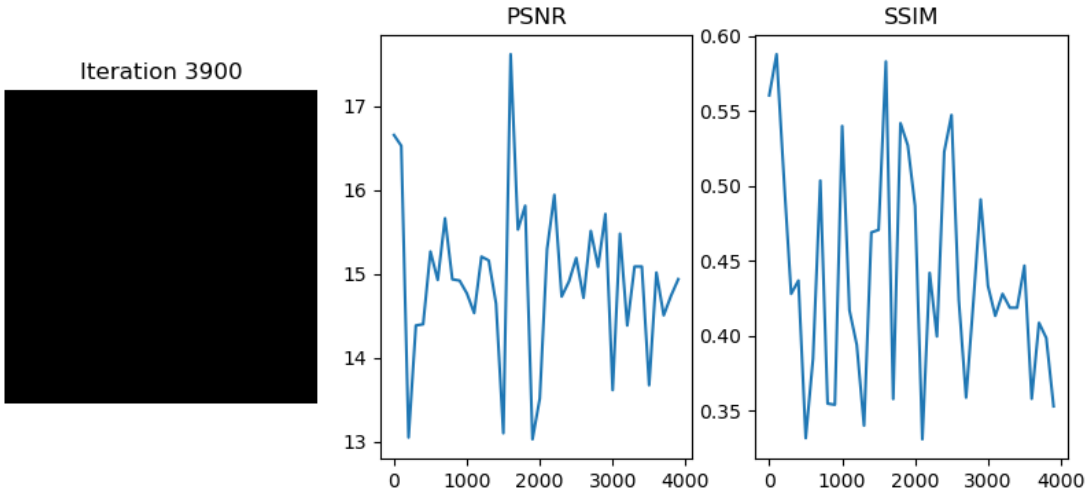
Fig. 15: NeRF Lego Output for 200×200 image



(a) Tiny NeRF Ship Output for 100×100 image



(b) Tiny NeRF Ship Output for 200×200 image



(c) Tiny NeRF Ship Output for 200×200 image without positional encoding

Fig. 16: Results for Ship Dataset