Feasibility study of Representing Scenes as Neural Radiance Fields (NeRF) for Fruit Representation on Citrus Trees

Uriel A. Rosa
Stanford University
uarosa@stanford.edu

ZhuoYi Cai
Stanford University
zycai@stanford.edu

Abstract

An accurate and fast fruit detection system is a key element of autonomous agricultural robotic platforms. The citrus harvesting task is particularly challenging due to illumination, noisy background and occlusions created by citrus canopies. In this study, we explore the possibilities and limitations in deploying the Neural Radiance Fields technique for fruit representation on citrus trees. A simplified model representing scenes as Neural Radiance Fields for view synthesis was implemented in a series of experiments focusing on how object type, neural network depth, image resolution, dataset size and ray sample resolution affected final 3D object rendering. This simplified model only requires RGB images, Camera-to-World matrices and focal length as inputs and outputs 3D rendered objects. PSNR was used as the sole metric for evaluating the quality of the final rendering output. All experiments were set up and run on publicly available Colab platforms, which imposed severe limitations on computing resources. Analysis of the experimental results show that computing power and training dataset quality, among other factors, put up strong challenges in the path for deploying NeRF techniques in fruit representation for citrus harvesting tasks.

** The aliases "citrus" and "orange" are used to distinguish between the two citrus trees.*

1. Introduction

Machine vision is expected to play a large role in robotic fruit harvesting. Fruit detection is essential for the implementation of tree fruit yield estimation, and automated fruit harvesting. An accurate and fast fruit detection system is a key element of autonomous agricultural robotic platforms [1].

Illumination, background and occlusions create challenges for fruit detection and harvesting. Occlusion is intense in citrus canopies. It creates obstruction for the camera views, but also for the end effector designed for harvesting the fruit [2].

Deep neural networks have been used to improve fruit detection. However, precise representation and geometric mapping of the fruits is an outstanding issue due to the presence of dense leave canopies, i.e. exhibiting low porosity. The implementation of the NeRF technique in this study, might help representing and rendering the fruits in the presence of sparse fruit exposures to the cameras.

The NeRF method has been proposed to synthesize novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views [3][4]. This novel approach may potentially help circumvent the described occlusion issues.

The goal of this study was to evaluate the benefits of a proposed vision system that combines a set of cameras, arranged surrounding a dense tree canopy, and uses a NeRF model to synthesize the geometry of the citrus tree scene. The particular interest is to accurately render and localize the fruits.

2. Background

Automated citrus harvesting has been a topic in research community for a few decades. A basic end-effector prototype using a canopy volume reduction technique has been proposed and evaluated [2]. Their method provides a temporary reduction in the tree canopy volume, creating space for the action of simple and potentially fast three-axis Cartesian manipulators. The action of the device creates fruit alignment that can simplify fruit quality assessment and picking. This simple prototype end-effector successfully harvested an average of 84% of the fruit and caused no significant damage to harvested fruits during the harvest operations.

A Faster Region-based CNN object detector was developed in a recent deep neural network study. In order to improve the efficiency and accuracy in the fruit detection system in 2016, a detector model was adapted through a transfer function for the fruit detection task using images computed from color (RGB) and Near-Infrared (NIR) data [1]. However, this method requires bounding box annotations. This process can be expensive and time consuming. For example, in one experiment, the model was trained to perform the detection of seven fruits, and the

entire process took four hours to annotate and train the new model per fruit.

A recently proposed NeRF-based view synthesis method [3][4] may shed a new light to the path of deploying neural network in fruit harvesting. This method takes in a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, φ)) and outputs volume density and view-dependent emitted radiance at that spatial location. For the remaining of this report, a series of experiments were performed on a simplified NeRF-based model for investigating the possibilities and limitations of incorporating this technique into autonomous robotic fruit harvesting applications.

3. Approach

3.1. Full NeRF View Synthesis Model

The original NeRF-based synthesis model method proposed in [3] optimizes an underlying 5D continuous volumetric scene function using a sparse set of input views to minimize the rendering error. It takes a 5D input (spatial location (x, y, z) and viewing direction (θ, φ)) and optimizes a deep fully-connected neural network (MLP) which represents the underlying continuous function. This fully-connected neural network consists of a double layer structure. The first structure is 8 fully-connected layers, with 256 channels per layer and using ReLU activation functions. The second structure is also a fully-connected layer with 128 channels per layer and uses ReLU activation functions. As a result, 5D coordinates (x, y, z, θ, φ) are developed into the radiance (RGB) emitted in each direction (θ, φ) at each point (x, y, z) in space. A density function at each point represents the amount of radiance accumulated by a ray passing through (x, y, z). The key idea here takes advantage of differentiability of the volume rendering process. Thus, by training and optimizing the underlying network with gradient descent, the rendering error across multiple views can be minimized, and the trained network is able to render a coherent model of the scene by assigning respective volume densities and accurate colors at each location in space.

However, the basic implementation of the optimizing NeRF representation has convergence issues. A positional encoding and hierarchical sampling procedure are used in the full NeRF model. These techniques enable the multilayer perceptron (MLP) to represent higher frequency functions and reduce the required number of samples per camera ray.

3.2. Mini NeRF Simplified View Synthesis Model

The model used in this study is a simplified version of the full NeRF view synthesis model. Due to computing resource limitations from the publicly available Colab platform, this simplified model was built based on the sample model tiny_nerf [5]. In this model, the network does not consider directional information (θ, φ) as the input and hierarchical sampling optimization procedure is not included. Thus, the same rendering quality as the original paper is not expected, but the data collected from the experiments conducted in this study should be sufficiently informative for the purpose of this study.

Figure 1 shows the modified version of the NeRF pipeline. The first step in the pipeline is to extract rays from the images using the image size, focal length and camera-to-world matrices (C2W). Then, based on the number of samples per ray setting, a sampled set of 3D points along with their spatial coordinates are generated by marching through each camera ray. The positional encoding step is also adopted from the original paper to map the images into high dimensional input for the network. As suggested in [6], mapping inputs to a higher dimensional space using high frequency functions helps better fitting data with high frequency variation contents. This is the case as in the tree images which have high frequency variation properties due to the constitution of leaves and fruits. Following the mapping function used in [3], the encoding function used is:

$$r(x) = (\sin(2^0\pi x), \cos(2^0\pi x), \dots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)) \quad (1)$$

where L is the encoding dimensional parameter.

After the sampled set of 3D coordinates are mapped to a higher dimensional space in *PosEnc*. this set of points and coordinates are then fed to the MLP networks. Since this model does not include directional information, instead of using the double "stack" structure like in the original paper, this MLP network only comprises of 8 fully-connected layers with 256 channels per layer and using ReLU activation functions. In one of the following experiments, network structure was modified to examine the effects it has on the rendered object output. In the following step, using the output emitted RGB color of each point in space from the network and the target images to calculate the training loss and PSNR. After N iterations, the RGB output from the network is then used to render the final 3D images.

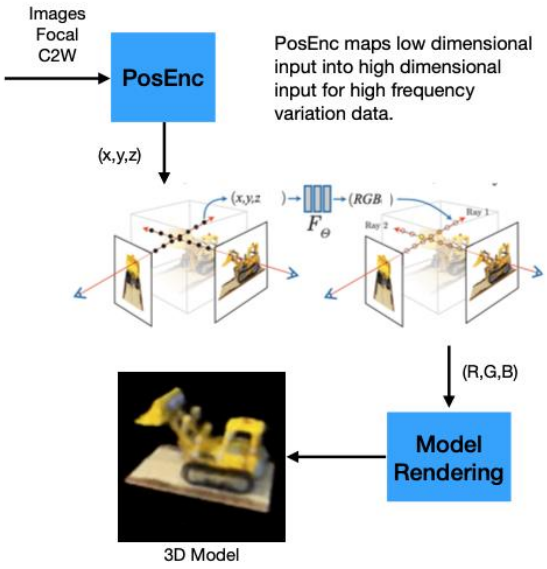

Figure 1 The mini NeRF pipeline diagram

3.3. Volume Rendering with Radiance Fields

The color of any ray passing through the scene is rendered using principles from ray tracing volume rendering techniques [7]. As described in [6], the expected color $C(r)$ of a camera ray could be estimated using a discrete set of stratified samples with the following equation:

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i \boldsymbol{c}_i \qquad (2)$$

where,

$$T_i = exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \quad (3)$$

$$\alpha_i = 1 - exp\left(-\sigma_j \delta_j\right) \quad (4)$$

The σ represents the volumetric density, which can be interpreted as the differential probability of a ray terminating at a point along the ray. The δ is the distance between adjacent ray samples and $\boldsymbol{c}_i$ is the RGB color estimated by the model at the sample locations.

4. Experiment Infrastructure Setup

4.1. Data Acquisition

In this study, synthetic images of an orange and a citrus tree along with ground truth camera poses and bounds were extracted from the Blender graphics package [8] by using Python3 scripts. Figure 2 shows the Citrus and the Orange tree models.

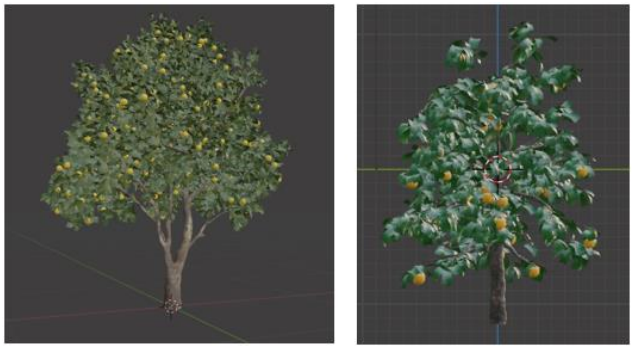

Figure 2 Citrus and Orange Tree Models

The Eleven cameras with focal distance f = 14mm were placed around the tree sector from -45° to 45° as illustrated in Figure 3. 11. The images were captured and available for training. As a result, it was learned that this small quantity of images was far from sufficient from obtaining a basic NeRF model to rendering any meaningful images.

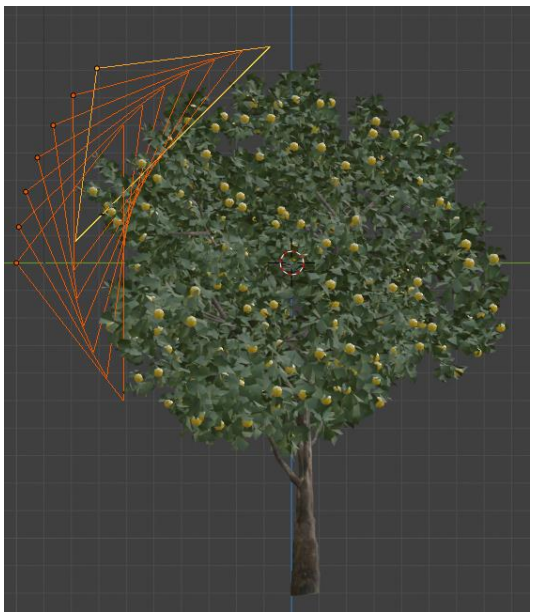

Figure 3 Initial Camera Layout Setup

This issue was corrected by capturing larger number of images using an automated script. A camera is set up such that it can capture images 360° surrounding the tree at every 3.6°. This setup is demonstrated in Figure 4. Images were captured with different focal lengths and resolutions. In addition, the angle between two adjacent images was determined as 360°/VIEWs. VIEWs represent the total number of images captured for the dataset. During the capturing process, the corresponding camera poses were extracted directly from Blender and packed along with the images. All images for each dataset shared the same focal length.

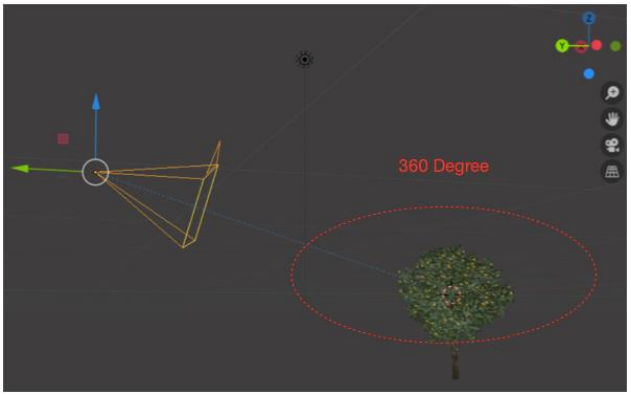

Figure 4 Camera Setup in Blender

In an image dataset of the orange tree, a spot light with 10 KW illumination power was introduced to enhance images captured from reflected tree leaves. This approach conditioned the model for the experiments later conducted where the effect of different object types on the NeRF network performance was studied. Figure 5 shows the effect of the lighting enhancement and its positioning with respect to the tree.

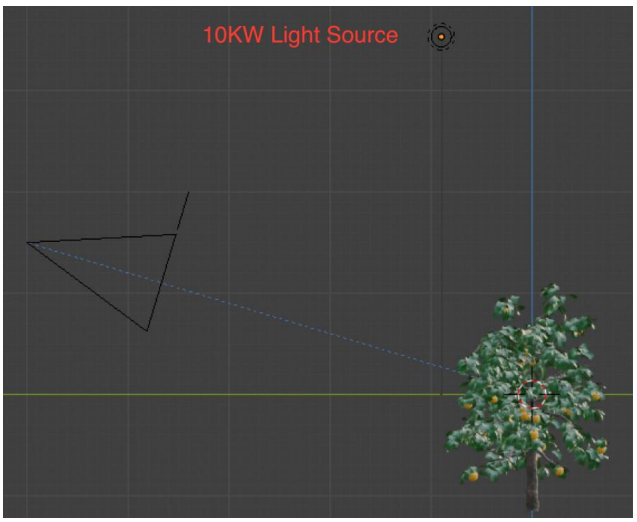

Figure 5 Lighting for Orange Tree

Table 1 lists all the datasets used for the following evaluations.

Table 1 Dataset List

Object	Resolution	# of images	Focal (mm)
Orange	100x100	200	50
Citrus	100x100	200	50
Citrus	200x200	200	50
Citrus	200x200	200	120
Citrus	100x100	2000	50

4.2. Computational Environment

The miniNeRF pipeline is implemented in the Colab Environment. This environment imposes several limitations and poses some challenges to the experiments conducted in this study.

First limitation is the resource allocation by Colab. This resulted in an unstable computational resource for each conducted experiment. Different runs set with the same settings showed different run times so that it was not possible using execution times to comparing computing performances.

Second, there were 12-hour hard-stop limits imposed for each run. Colab stops the execution without saving the experimental model. Also, there was a limit in the GPU resource usage controlled by the Colab Environment. Two different accounts were required to continue the experiments, or upgrade to the Pro version.

The most critical limitation encountered was the OOM (memory) issue. This prevented the runs from using higher resolution images and increasing the ray sample beyond 65.

However, besides all these limitations, the data collected through the experiments were informative for the purpose of this study.

5. Results

A series of experiments we conducted in this study to find out the limitations and possibilities in deploying Neural Radiance Fields technique for fruit representation on a citrus tree.

5.1. Experimental Constraints

As discussed in section 4.2, the Colab environment imposed some limitations on the flexibility of the delineated experiments. Here are a few constraints:

- Resolution: not higher than 200x200
- Ray Samples: not higher than 65
- PosEnc Dimension: not higher than 24
- Iterations: not higher than 20000
- Evaluation Metric: PSNR

Only the PSNR was used in this evaluation due to the instability of the Colab environment. The PSNR is calculated using equation (5):

$$PSNR = -10 * \log_{10} Loss \quad (5)$$

where $Loss$ is the mean square of the difference between the rendering RGB values and the ground truth images.

5.2. Experiment 1: Object Type

Two tree models, orange and citrus trees, were used to examining if object geometry, color and lighting condition have any effects on the training and rendering performance of the model.

Figure 6 Sample Citrus (left) and Orange (right) Tree Image

Figure 6 shows a pair of sample images. A citrus tree image under normal condition is shown on the left. An orange tree image with a 10KW light source added is shown on the right. Due to the additional light source, there were more light intensity reflected from the leaves. Also, the color of fruit and leaves shows more saturation on the orange tree.

Two datasets with 200x200 and 100x100 resolution images with focal lengths of 50mm and 150mm were used in the experiment runs. Ray samples were set to 65. Positional encoding dimension was set to 6. After 100 iterations, the citrus tree images seemed to give a better rendering result with the PSNR around 18.5 while the orange tree images result in a PSNR around 14.5. Figure 7 shows this comparison results.

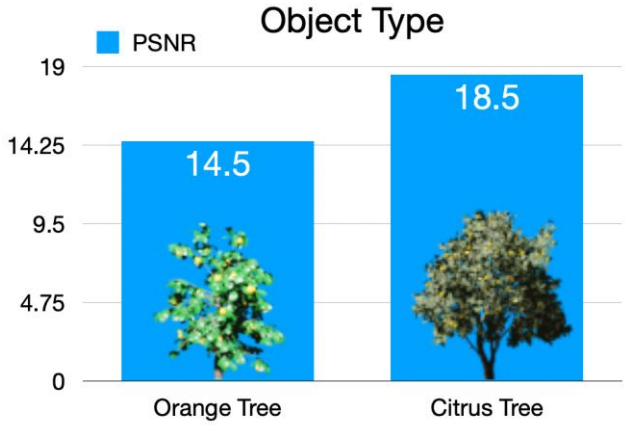

Figure 7 PSNR Rating Comparison on Object Type

The citrus tree images result in relatively higher PSNR may be due to the reflection gloss on the orange tree leaves. Also, the citrus tree shows denser green color at the center of the images. Both runs had similar run times around 3 hours. Assuming there were the same computing resources on both runs, the object type did not affect the computing performance, but the color and glossy surface of the object seemed to reduce the rendering quality of the object. This is critical because, in the context of fruit harvesting, the images taken by the robot are mostly under sunny weather conditions. Exposure may need to be adjusted accordingly.

5.3. Experiment 2: Focal Length

Images with focal length 50mm and 150mm were examined in this experiment. Figure 8 shows a pair of sample images. On the left there is the image with f=50mm and the image with f=150mm is on the right.

Figure 8 Citrus Tree images with 50mm and 150mm focal length

There is a contrast separation between the tree and the background on the image with 50mm while the image with 150mm is filled with leaves and fruit due to the zooming effect of the lens.

Two datasets with 200-200x200 resolution images with focal length 50mm and 150mm respectively were used. Ray sample was set to 65. Positional encoding dimension was 6. After 250 iterations, the images with 50mm resulted in significantly better rendering performance with a PSNR of 17.0 PSNR while images with 150mm resulted in a PSNR of only 11.6.

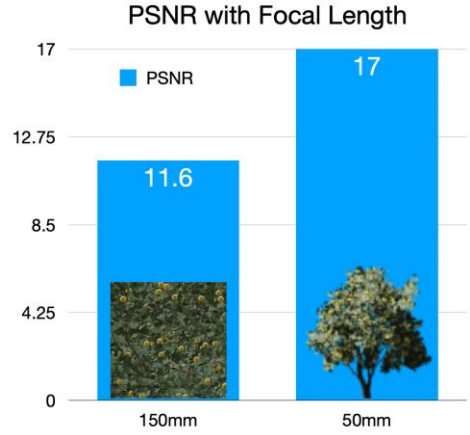

Figure 9 PSNR Rating Comparison on Camera Focal Length

The images with camera focal length 50mm resulted in apparent better rendering results. This is most likely because the separation between the tree and the background helped the network learning the 3D object in the scene. Meanwhile, the 150mm images filled with leaves and fruits did not have the resolution for the network to identify the actual tree object. For the fruit object, the leaves create a

very noisy background, which highly reduces the network learning ability. In the context of fruit harvesting, the robot is likely in close proximity to the fruit. This will give a challenge to the system for an accurate object detection.

5.4. Experiment 3: Training Iterations

The previously described experiments limited the training iterations around 100. In this experiment the training iteration is evaluated.

The same 200x200 image set with 150mm focal length as the Experiment #2 was used, except for the number of iterations. It consisted of 200 images. Ray sample setting remained 65. Positional encoding dimension was 6. The learning rate of the network was set to 5e-4. However, he iterations were set to 250 and 1000 in two different runs.

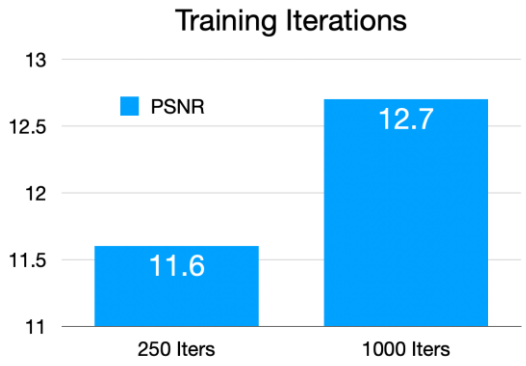

Figure 10 PSNR Comparison on Training Iterations

Figure 10 shows that the network trained with 1000 iterations resulted in only a slightly better rendering performance. This was partly because of the close-up images used in this experiment. As discussed previously, close-up images pulled down the rendering performance. It was expected the performance gap would significantly increase between 250 and 1000 iterations if the training dataset had 50mm images. Also note that the run time for 250 iterations here was about 3 hours while the 1000 iterations took about 8 hours. These run time numbers may be affected by resource allocation in the Colab environment.

5.5. Experiment 4: Input Dimension

As discussed in 3.2, mapping inputs to a higher dimensional space using high frequency functions helped better fitting data with high frequency variation contents.
In all the experiments discussed previously, the input positional encoding dimensions were set to 6. This number may have been sufficient in most cases. However, in this study, tree images were used as the training datasets. They tend to have high frequency variation contents due to leaves and fruit placements, and their shadows. In this experiment, 1950-100x100 images with 50mm focal length were used as the training dataset. The Ray sample setting was 55. The positional encoding dimension was increased to 24. The network run through 20,000 iterations. The result was significantly different as shown in Figure 11.

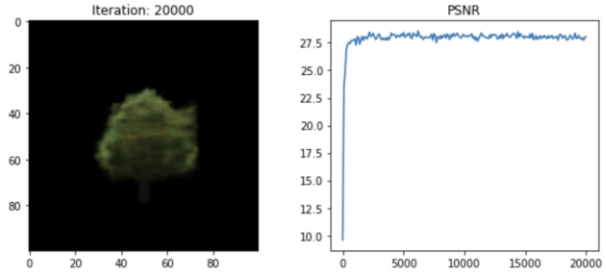

Figure 11 PSNR graph with PosEnc = 24 at 20000 iterations

This experiment was very informative in several observations. First, the PSNR was significantly higher comparing to the experiments with lower positional encoding dimensions (24 vs 6). The resulted PSNR for the run shown in Figure 11 was up to 27.5. This seems to be consistent with what was expected since the images of the tree object contains high frequency variation contents. Second, surprisingly, the training speed was also significantly faster. The 20,000 iterations could be completed in about 5 hours. This may have been due to the better computing resources allocated for the task by Colab. However, consistent observations were made in other runs with higher positional encoding dimensions. So, it is thought the increase of positional encoding dimension contributes to the training time reduction. Lastly, the PSNR graph presented on the right in Figure 11 shows that the PSNR saturates at around 4,000 to 5,000 iterations. This requires further study. One possible reason is the low image resolution of the training dataset which does not contain enough details in the image contents compared to the higher resolution images. It was not possible to verify this assumption since the Colab environment prevented the runs from allowing image resolutions beyond 200x200.

5.6. Experiment 5: Network Structure

The NeRF optimizes a deep fully-connected neural network (MLP) which represents the underlying continuous function. In [6], the full version of NeRF includes 8 fully-connected layers with ReLU activations and 256 channels per layer. In addition to that, the output of a 256-dimensional feature vector is concatenated with the camera ray's viewing direction and then passed to an additional fully-connected layer with a ReLU activation and 128 channels before outputting the view-dependent RGB color.

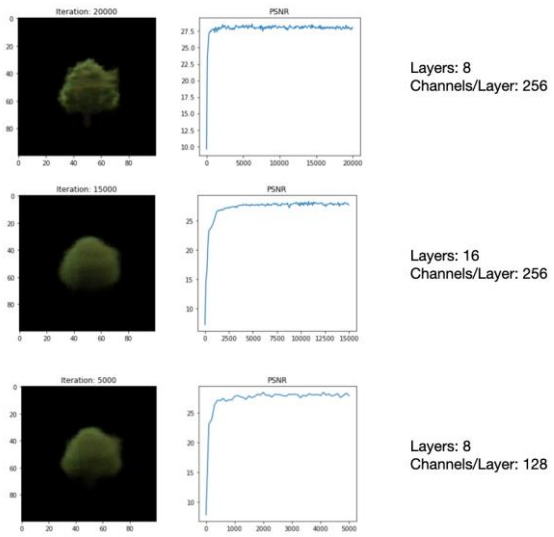

Figure 12 PSNR performance on Network Structure

In this study, since directional information were not included as the input to the network, the additional output layer was omitted and the output RGB color was calculated with a sigmoid function. This experiment attempted to determine how differently network structures perform. As in Experiment #4, 1950-100x100 images with 50mm focal length were used as the training dataset. The Ray sample setting was 55. The positional encoding dimension was increased to 24.

Figure 12 shows the results (rendered images and PSNR vs iterations graph) from three different network structures. The network for the left bar graph has 8 layers and 256 channels per layer. It has the best performance of the three. The PSNR saturated at around 4,000 iterations and reached the value of 27.5. The middle bar graph has the network with 16 layers and 256 channels per layer. It takes 5,000 to 6,000 iterations to reach PSNR saturation of 26.5. The right bar graph has an 8-layer network with 256 channels per layer. The final PSNR is about 25.8. Note that this value was not the peak value during the training process. Instead, it was obtained when the training stopped at 5,000 iterations. So, this was probably no the final PSNR value of the network. But it can be seen that it definitely reaches more than 5,000 iterations to converge. These results are presented in Figure 13.

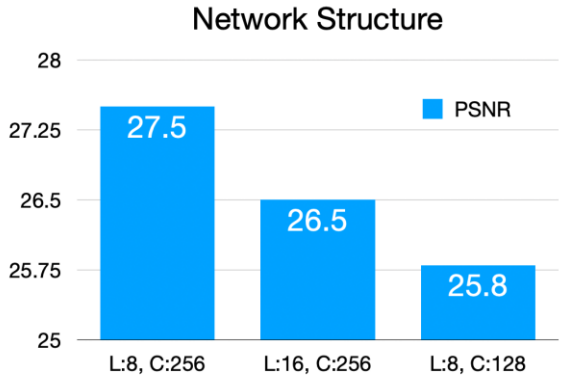

Figure 13 PSNR Comparison on Different Network Structure

6. Conclusions

From these series of the experiments in this study about the NeRF representation technique, it has been concluded that this technique has a huge potential with a few to-be improved problems. The NeRF can be used to accurately represent high resolution scenes. However, one constraint is that it can only be used to represent static scenes. This is an issue to be addressed in future research if the goal is to apply this technique in fruit harvesting robotic contexts. In addition, it requires significant computing power to train and process the source images. This may require extended hardware acceleration support. Training image acquisition is another issue to be addressed, since it requires a large number of images as the training dataset in order to have the expected rendering results. Close-up images may pose additional challenges to the adoption of this technique. However, further studies are required to substantiate this statement.

However, the NeRF technique is reach of potential applications. If computing resources allow, it is desirable to seeing how higher resolution images with fewer image quantities, higher ray sample numbers and different input dimensions affect the network performance. In addition, it can be looked even further to seeing if different implementations can help improving training performances, such as run time reduction.

The results of this study did not stablish that the current implementation of Neural Radiance Fields technique is ready for applications like fruit harvesting. It can be concluded that further studies and improvements are required before this technique can be deployed for the representation of a citrus tree in the real-world application context.

7. References

[1] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez and C. McCool. DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors (Basel). 2016 Aug

3;16(8):1222. doi: 10.3390/s16081222. PMID: 27527168; PMCID: PMC5017387.
[2] S. Lee and U. A. Rosa. Development of a Canopy Volume Reduction Technique for Easy Assessment and Harvesting of Valencia Citrus Fruits. Transactions of the ASABE. 49(6)1695-1703.
[3] NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng. ECCV. 2020.
[4] NeRF youtube video. [ECCV 2020] NeRF: Neural Radiance Fields (10 min talk) - Bing video
[5] https://github.com/bmild/nerf
[6] N. Rahaman, A. Baratin, D. Arpit, F. Dra¨xler, M. Lin, F.A. Hamprecht, Y. Ben-gio and A.C. Courville: On the spectral bias of neural networks. In: ICML (2018)
[7] J.T. Kajiya and B.P.V. Herzen: Ray tracing volume densities. Computer Graphics (SIGGRAPH) (1984)
[8] Blender Graphic Package. Version 2.81.

8. Supplement Material

Source code and dataset for the experiments can be downloaded from ***here***. The following table lists the files included in the package.

Table 2 Submission Package File List

File/Folder Names	Description
miniNeRF.ipynb	Colab notebook for miniNeRF implementation
view360.py	Image acquisition code
navelTree.blend	Blender file for the Tree model
cs231a_final_presentation	Presentation keynote
videos	Some experiment rendering videos
tree_2000_100x100	2000 100x100 images dataset
tree_orange	Orange tree images and model
tree_citrus_200x200	200 200x200 images of citrus tree dataset
tree_1000_100x100	1000 100x100 citrus tree image dataset
tree_200_10KW	200 100x100 citrus tree images with 10KW light
citrus_tree_200x200_f150mm	100 200x200 citrus tree close-up images
tree_100_100x100	100 100x100 citrus tree images (We messed up the fruit in this dataset.)
tree3_11_cams_failed_test	11 images from 11 cams (a failed experiment; included here anyway)