Neural Radiance Fields For Lower Dimensions

Jesus Crespo
MIT
jrch@mit.edu

Abstract

A simplified implementation of NeRF was shown to synthesize novel views of 2D scenes. A procedure for generating 2D scenes on which to train models is presented.

1. Introduction

The problem of novel view synthesis lies at the intersection of computer vision and computer graphics. Generating images of scenes from different perspectives starting only with previously taken images is of interest to both the theoretical and applied community, as it requires an actually functional approach to scene understanding and it opens a whole new field of possiblities for creative endeavours like animated graphics in video games.

In 2020, Mildenhall *et al*. [4] introduced a solution to address part of this problem: NeRF. It provides a procedure for novel view synthesis which relies on the representation of 3D scenes as neural radiance fields learned from 2D images of the target scene along with their camera parameters. Their approach showed improved results over prior work [2, 3, 6] both quantitatively and qualitatively.

In this paper, a simplified TensorFlow [1] implementation of NeRF, TinyNeRF [5], was adapted to work with 2D scenes, with guidance from a PyTorch implementation of NeRF by Lin [8]. The ways in which the implementation here presented differs from a faithful implementation of NeRF are described in Sec. 3. Due to a lack of readily available 2D scene data, a procedure for generating synthetic 2D scene data is also introduced and subsequently used to train a TinyNeRF2D model. A Google Colab notebook and animated GIFs can be found here: https://drive.google.com/drive/folders/1Re8uj_723rhkp7XxJS4NVFg9C27vjyze?usp=sharing.

2. Related Work

The author is not aware of other work on NeRF for novel view synthesis of 2D scenes. See Xian *et al*. [7] for an extension of NeRF to space-time (4D) scenes.

3. Method

3.1. Scene Generation

Square 2D scenes 100 pixels wide were generated. Starting with a black background, pixels in the scene were colored white either according to a mathematical formula, like a disk with a radius of 31 pixels shown in Fig. 16, or by hand using the GIMP image editor, like that shown in Fig. 17. Any white pixels were then colored either with a full color wheel or a graded color wheel, shown in Fig. 15. The full color wheel was generated using HSV values, letting saturation and value be their maximum and letting hue uniformly increase from its minimum to its maximum with a full clockwise rotation starting from the left. The graded color wheel has maximal saturation at the center and uniformly decreases to its minimum to the edges of the scene. For the results presented in Sec. 4, two scenes were used: the fully colored disk scene shown in Fig. 16 and the graded colored complex scene shown in Fig. 17.

3.2. Image Generation

To generate ground truth 1D images of a 2D scene, rays were projected from a camera on the scene through each image pixel onto the scene, and the first colored pixel on the scene to intersect a ray was assigned as the color for that image pixel. See Figs. 18 through 25 for visualizations. This differs from NeRF in that viewing direction does not affect the emitted color from a pixel on the scene.

Rays started from near bounds (10 for both scenes) and ended at far bounds (50 for the disk scene and 70 for the complex scene), with pixels sampled uniformly along the ray (45 for the disk scene and 100 for the complex scene). Intrinsic camera parameters consisted of 1D image size (32 pixels for both scenes) and focal length (20 for the disk scene and 30 for the complex scene). Extrinsic camera parameters consisted of distance from center (45 pixels for both scenes) and counterclockwise angle from the bottom of the scene (360 angles uniformly distributed starting from the bottom of the scene going counterclockwise), with the camera always pointing toward the center of the scene.

By varying the camera angle, 360 1D images were collected for each scene. These were split into training and testing images by selecting every fifth image starting with the first image for training and the rest for testing, leading to 72 training images and 288 testing images. Note that both sets still have images uniformly distributed around the scene. See Figs. 11 through 14 for visualizations.

3.3. Model

In this paper, Mildenhall *et al.*'s hierarchical volume sampling procedure is not used, so the model presented here is that of their coarse network. The architecture for this model follows that of NeRF with a few exceptions: the input to the model is a concatenation of the 2D position and its positional encoding, and viewing direction is not used at all, so the two last layers are instead replaced with a final output layer of 4 channels with no activation. Additionally, the positional encoding used did not include the multiplicative factor of π. The color and volume density are then extracted from the feature vector by passing it through sigmoid or ReLU activations, respectively, when needed. Given that the ground truth images of scenes were invariant to viewing direction (see Sec. 3.2), this architectural change aligns with the dataset. See Mildenhall *et al*. [4] for a full description of their architecture and positional encoding procedure. Both scenes were tested using a positional encoding maximum frequency $L = 4$, and the complex scene was additionally tested using $L = 6$.

3.4. Training

Rather than randomly sampling pixels in the training set as done in NeRF, an image and its camera pose from the training set are randomly sampled at each training step. The pose is used to generate the rays for each pixel in the image. Points are sampled uniformly along each ray (beginning at a bin) and random uniform noise is then added so each point along the ray is randomly sampled from its bin. These points are reshaped into batches before being fed to the model to lower memory usage at a given moment. The color and volume density are then extracted for each point along each ray, which are then used to calculate the estimated color for each pixel in the image using their volume rendering procedure. See Mildenhall *et al*. [4] for a full description of their stratified and hierarchical volume sampling procedures.

The optimizer used with this model was Adam using a learning rate of 5×10^{-4} with otherwise default TensorFlow settings. Unlike done in NeRF, this implementation did not use learning rate decay. Additionally the loss function used was Mean Square Error (MSE) as opposed to NeRF's Sum of Squared Errors. The average loss across the test set was computed every 1/20th of the total optimization steps, though it was not used in training in any way other than for collection of results. Various different GPUs were

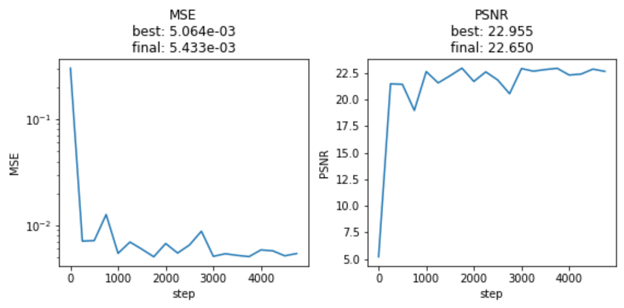

Figure 1. Test results on 2D scene of fully colored disk using 4 positional encoding frequencies after 5K optimization steps.

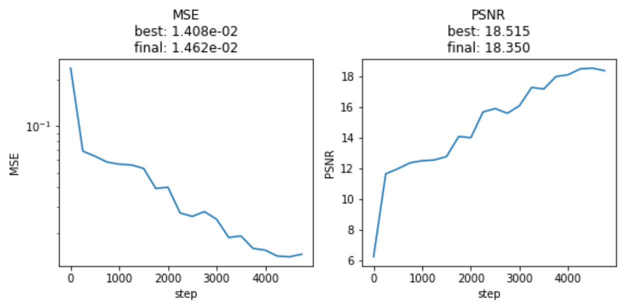

Figure 2. Test results on graded colored complex 2D scene using 4 positional encoding frequencies after 5K optimization steps.

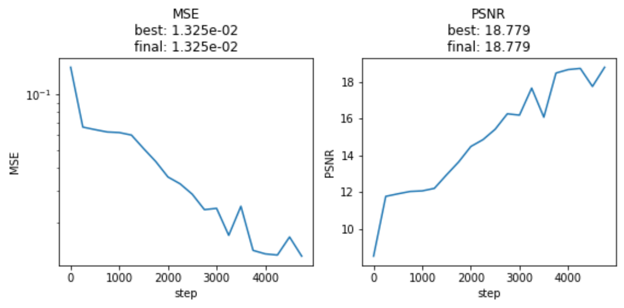

Figure 3. Test results on graded colored complex 2D scene using 6 positional encoding frequencies after 5K optimization steps.

used, though training one scene (while additionally calculating the test loss) always took less than 5 minutes for 5K optimization steps on a single Google Colab GPU.

4. Results

Four different configurations were tested: the disk scene using positional encoding maximum frequency $L = 4$ with 5K optimization steps (Fig. 1), the complex scene using $L = 4$ with 5K steps (Fig. 2), and the complex scene using $L = 6$ after 5K steps (Fig. 3) and after 30K steps (Fig. 4). Mean Square Error (MSE, lower is better) and

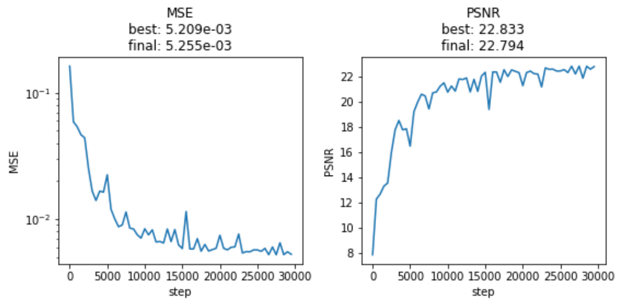

Figure 4. Test results on graded colored complex 2D scene using 6 positional encoding frequencies after 30K optimization steps.

Peak Signal-to-Noise Ratio[1] (PSNR, higher is better) values were computed for each model tested and plotted across training steps. For the following discussion we shall only consider PSNR given that one can be computed from the other and vice-versa.

For the disk scene, after 5K steps using $L = 4$, the model achieved a best PSNR score of 22.955 at some point during training. For the complex scene, after 5K steps using $L = 4$, the model achieved a best PSNR score of 18.515. After 5K steps using $L = 6$, the model achieved a best PSNR score of 18.779. After 30K steps using $L = 6$, the model achieved a best PSNR score of 22.833.

The model's PSNR score on the disk scene was greater than that on the complex scene by 4.44, suggesting that scene complexity (loosely approximated here by the repeating, regular nature of the disk as opposed to the arbitrary nature of the complex scene) negatively affects the learning of the model, as one might expect.

Increasing L from 4 to 6 showed some improvement (increased PSNR by 0.264), though this increase was over 16 times smaller than that across scenes, suggesting that this increase in positional encoding frequencies was not very helpful. In contrast, simply running the model for 6 times as long (from 5K to 30K optimization steps) showed remarkably better improvements (increased PSNR by 4.054), over 15 times larger than the $L = 4$ to $L = 6$ jump.

Both quantitative and qualitative results suggest these TinyNeRF2D models can synthesize novel views of 2D scenes to a decent extent. Training (as opposed to test) results can be seen in Figs. 5 through 7. We encourage the reader to see Figs. 8 through 10 for a qualitative comparison of the models' comprehensive view of scenes against ground truth, and animated GIF versions of these can be found at https://drive.google.com/drive/folders/1Re8uj_723rhkp7XxJS4NVFg9C27vjyze?usp=sharing.

[1]Actually computed as $-10\log_{10}(MSE)$, so shifted negatively by $10\log_{10}(255^2) \approx 48.131$ from proper definition.

5. Conclusion

This paper presents a simplified adaptation of NeRF which synthezises novel views of 2D scenes. A procedure for generating toy 2D scenes is described, and two such scenes were then tested with TinyNeRF2D models and these learned quantitatively and qualitatively decent representations of the scenes. Limitations to this study include: a small sample of results; the models being trained on uniformly distributed images around the scene as opposed to different and arguably more difficult image distributions; not as faithful a replica of NeRF for 2D scenes as could be imagined.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 1

[2] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4), July 2019. 1

[3] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), July 2019. 1

[4] 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 *ECCV*, 2020. 1, 2

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Tiny nerf. https://colab.research.google.com/github/bmild/nerf/blob/master/tiny_nerf.ipynb, 2020. 1

[6] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1

[7] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020. 1

[8] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 1

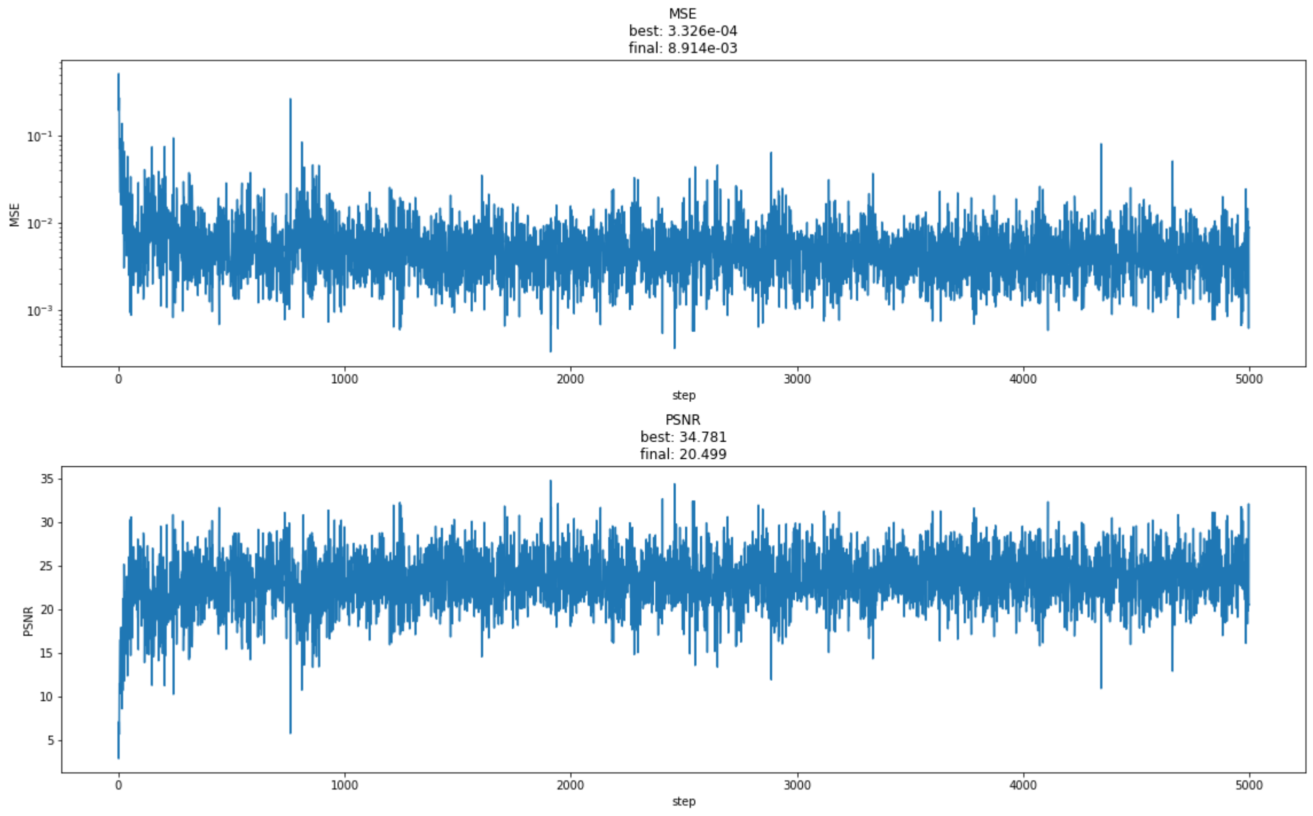

Figure 5. Training results on 2D scene of fully colored disk using 4 positional encoding frequencies after 5K optimization steps.

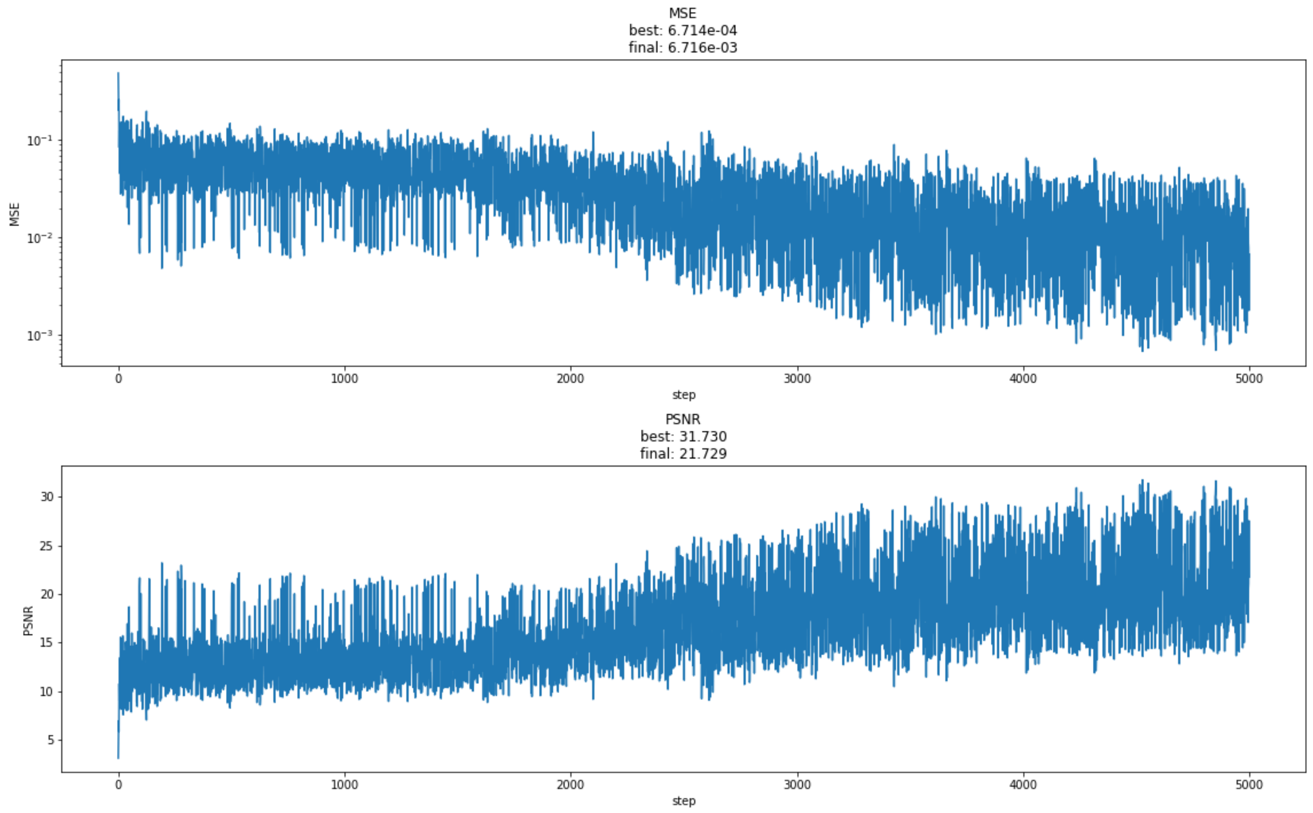

Figure 6. Training results on graded colored complex 2D scene using 4 positional encoding frequencies after 5K optimization steps.

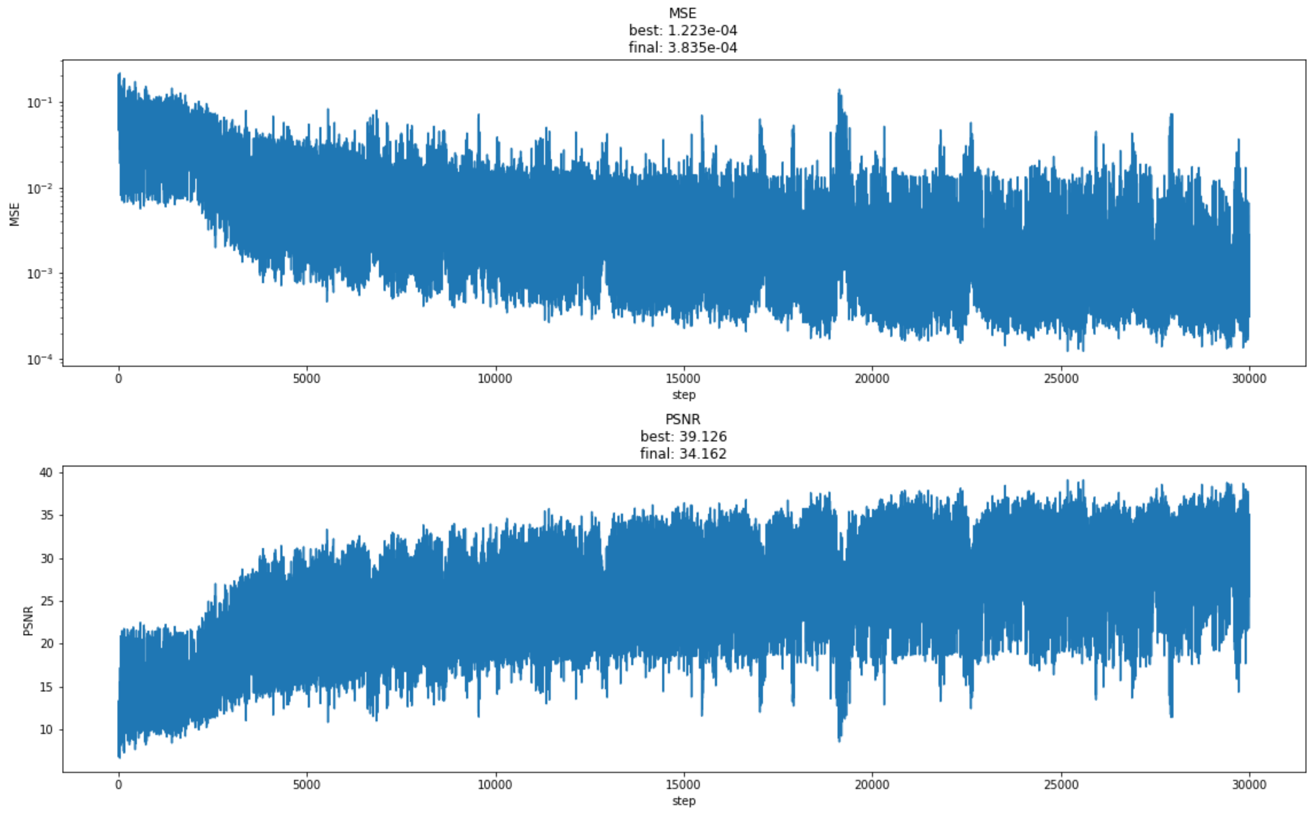

Figure 7. Training results on graded colored complex 2D scene using 6 positional encoding frequencies after 30K optimization steps.

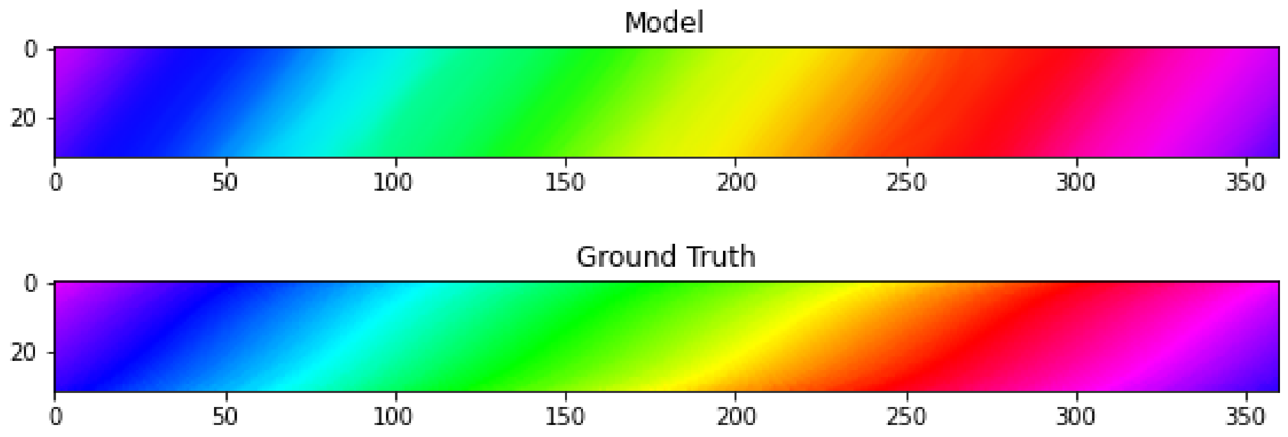

Figure 8. Qualitative comparison between model and ground truth scene views of fully colored disk using 4 positional encoding frequencies after 5K optimization steps. Y axis is image coordinate, while X axis is image number. There are 360 images uniformly distributed around the scene, going counterclockwise from the bottom of the scene. Note how the model seems to have a greater slant to its colors, which reflects the greater slant as seen in its training set shown in Fig. 11.

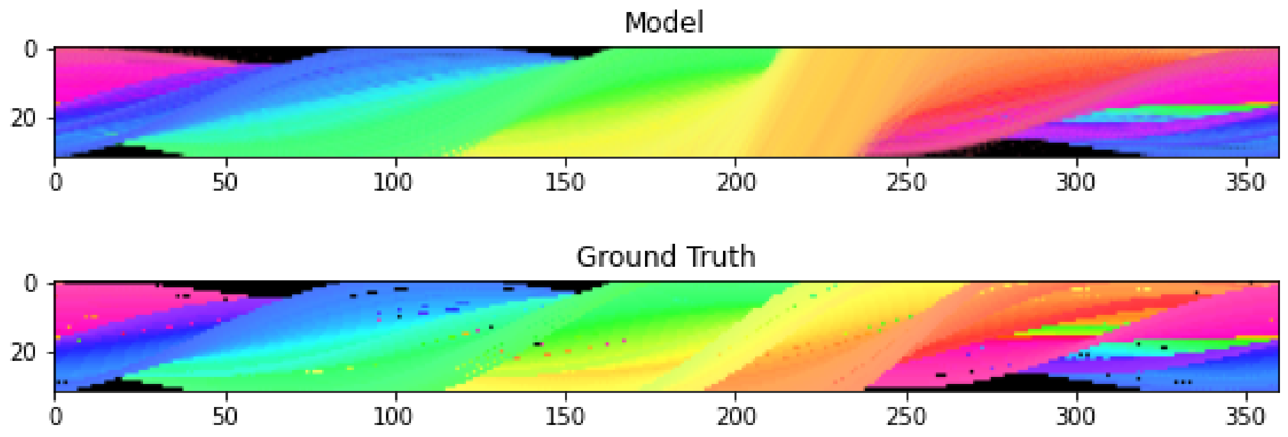

Figure 9. Qualitative comparison between model and ground truth scene views of graded colored complex scene using 4 positional encoding frequencies after 5K optimization steps. Y axis is image coordinate, while X axis is image number. There are 360 images uniformly distributed around the scene, going counterclockwise from the bottom of the scene. Note that the model has very smooth transitions that don't include the ground truth disturbances. In particular, it doesn't even contain the spots seen in its training set shown in Fig. 12.

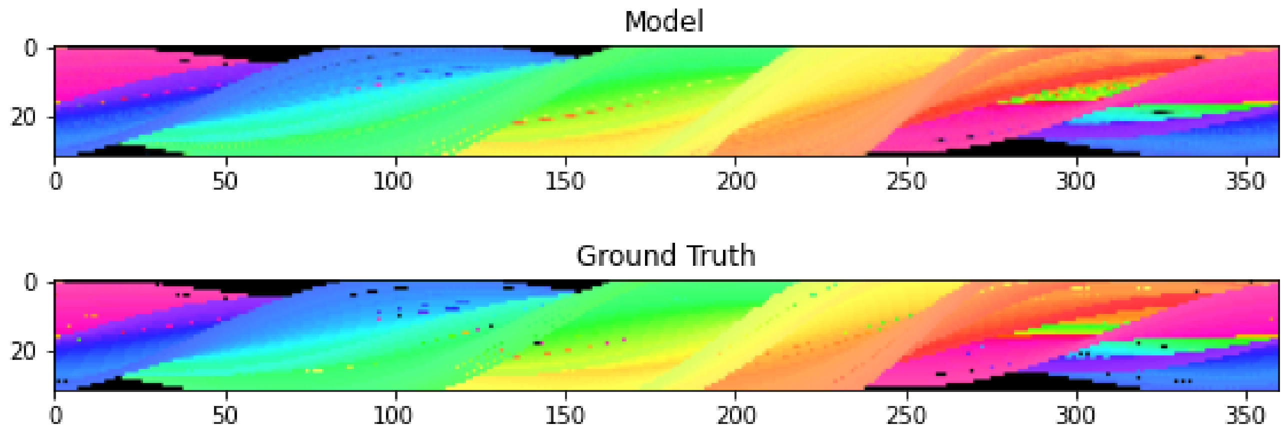

Figure 10. Qualitative comparison between model and ground truth scene views of graded colored complex scene using 6 positional encoding frequencies and 30K steps. Y axis is image coordinate, while X axis is image number. There are 360 images uniformly distributed around the scene, going counterclockwise from the bottom of the scene. Comparing with Fig. 9, the model now has incorporated some of the spots and disturbances noticeable in the ground truth scene and the training set shown in Fig. 12, albeit still to a much smoother extent.

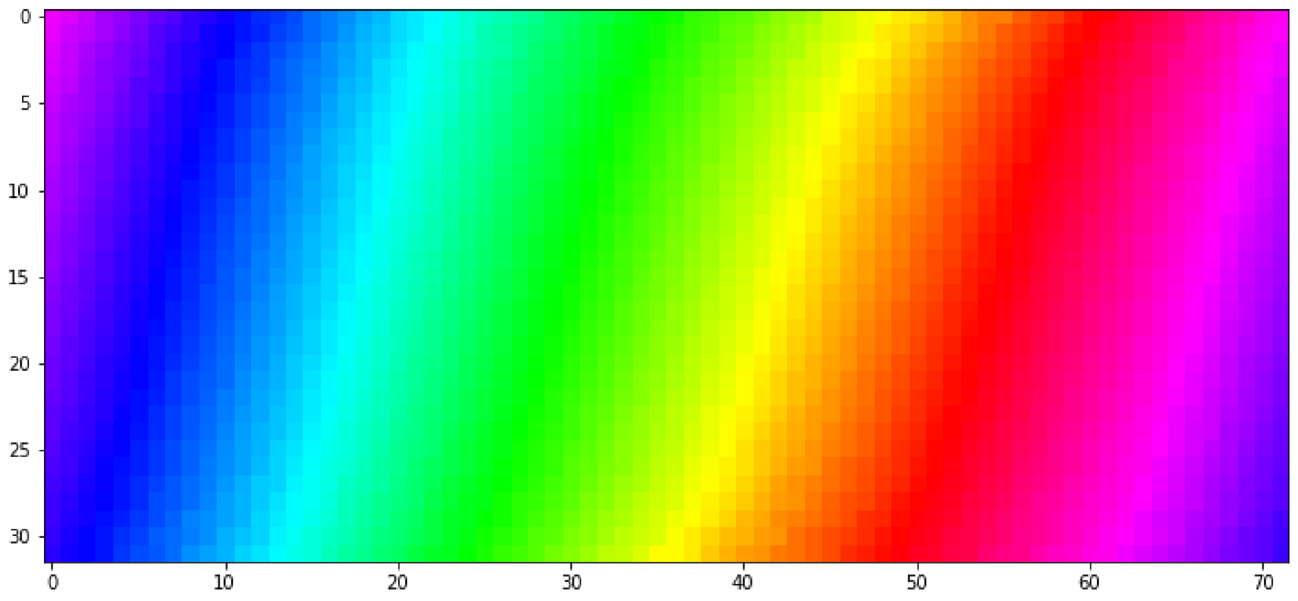

Figure 11. Training images for 2D scene of fully colored disk. Y axis is image coordinate, while X axis is image number.

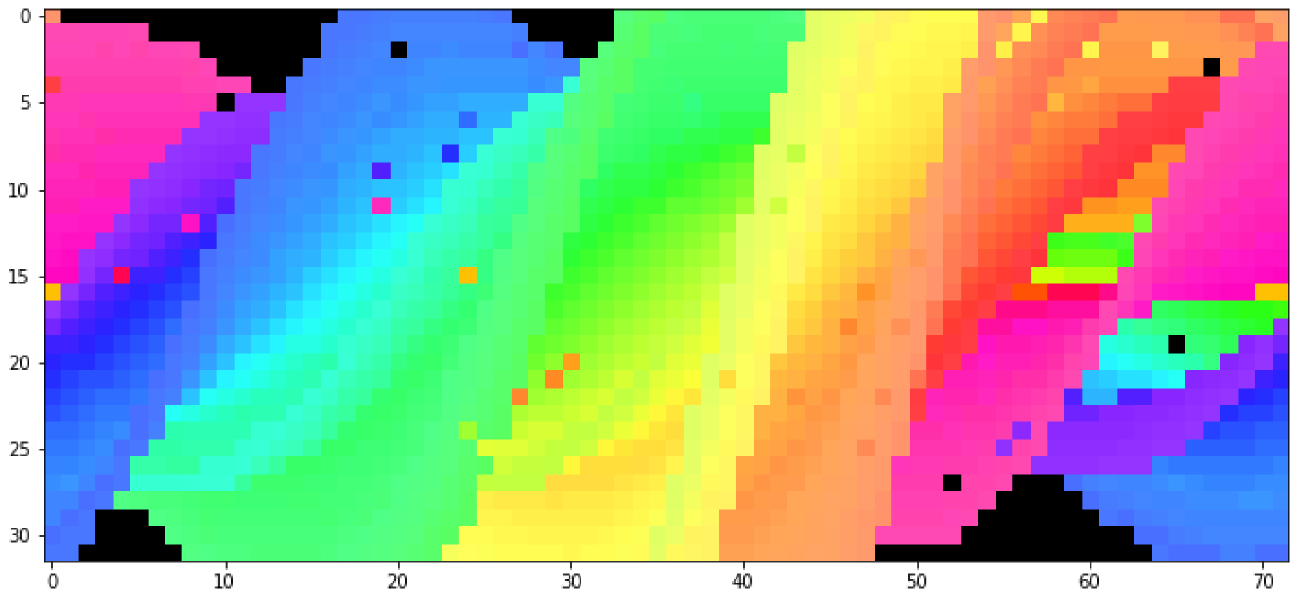

Figure 12. Training images for graded colored complex 2D scene. Y axis is image coordinate, while X axis is image number.

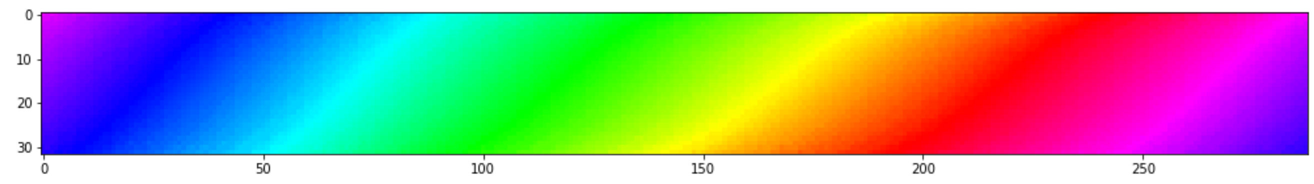

Figure 13. Testing images for 2D scene of fully colored disk. Y axis is image coordinate, while X axis is image number.

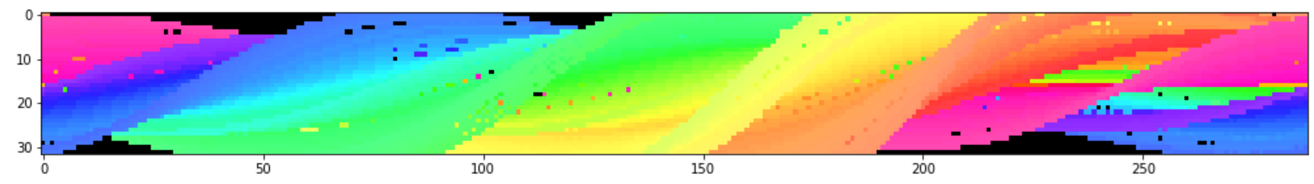

Figure 14. Testing images for graded colored complex 2D scene. Y axis is image coordinate, while X axis is image number.

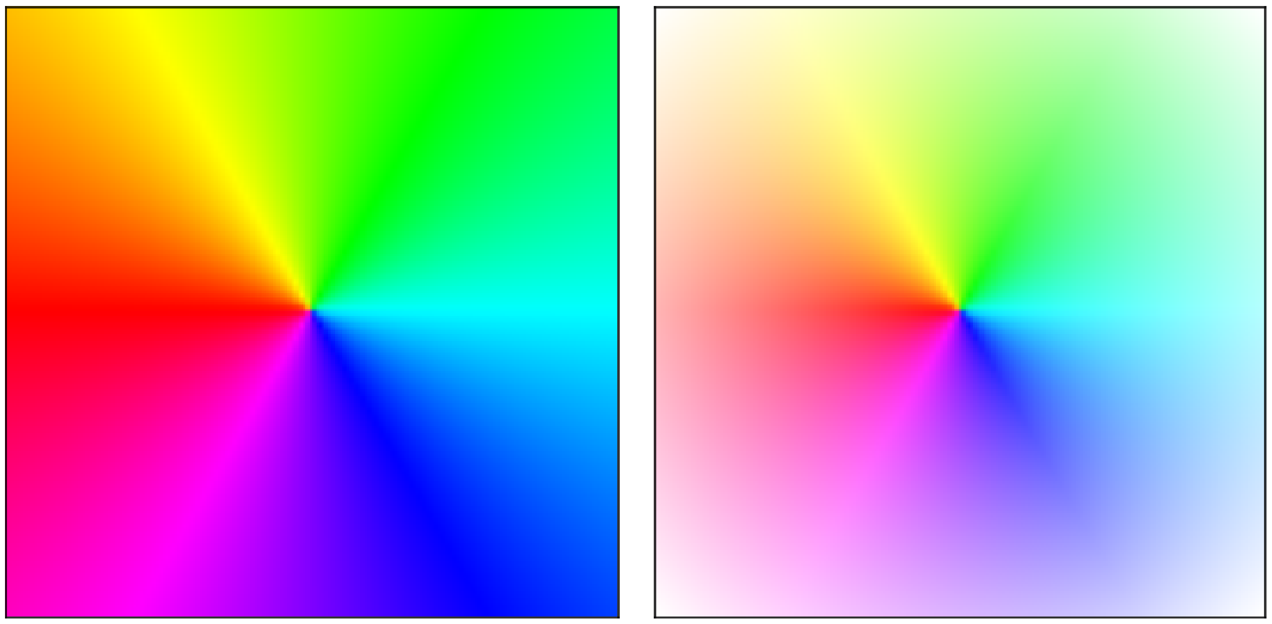

Figure 15. Color wheels applied to white scenes to produce colorful scenes. Left: full color wheel, where hue increases clockwise starting from the left. Right: graded color wheel, where saturation drops off with the normalized distance from the center.

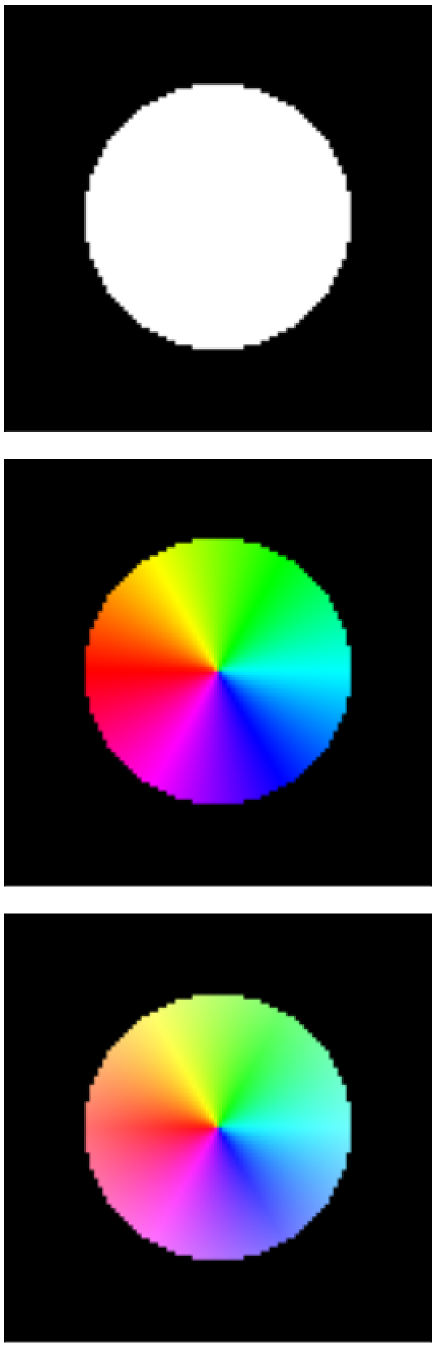

Figure 16. 2D scenes of a disk. From top to bottom: white disk, fully colored disk, graded colored disk. Figure 15 shows these color wheels.

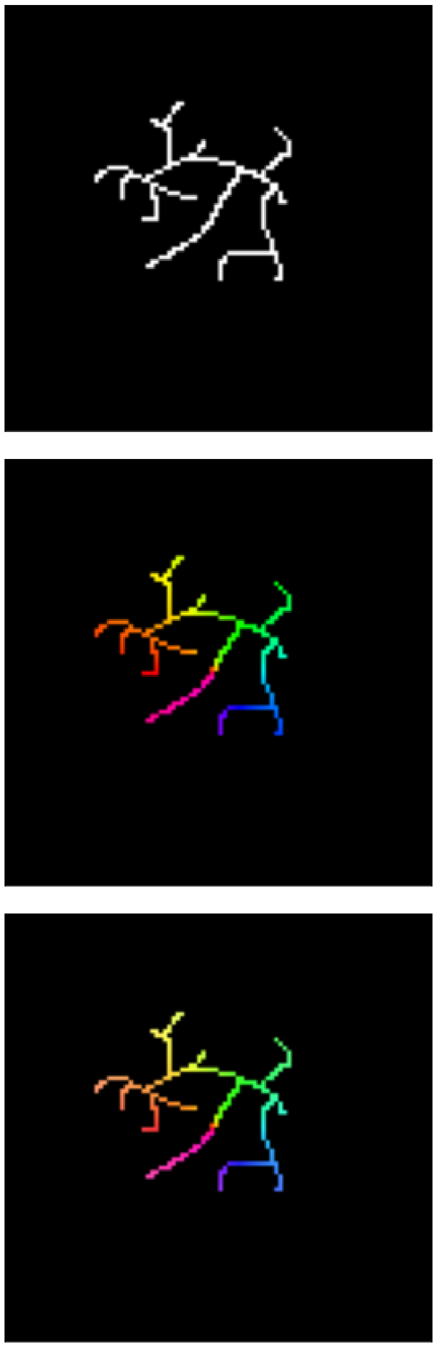

Figure 17. Complex 2D scenes. From top to bottom: white scene, fully colored complex scene, graded colored complex scene. Figure 15 shows these color wheels.

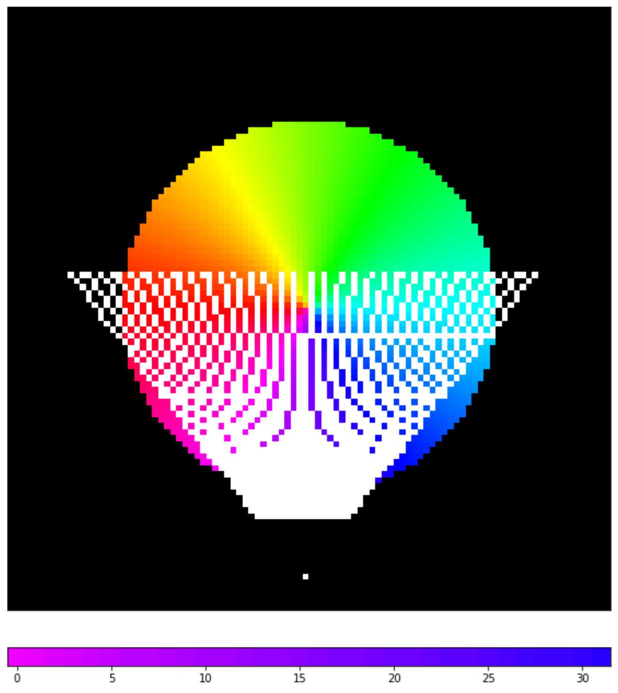

Figure 18. Camera visualization and corresponding 1D image of a 2D scene of a fully colored disk from the bottom.

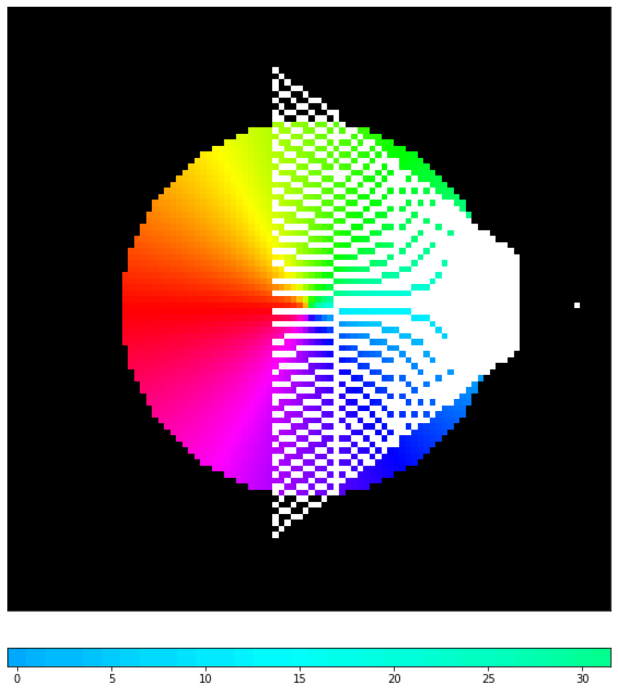

Figure 19. Camera visualization and corresponding 1D image of a 2D scene of a fully colored disk from the right.

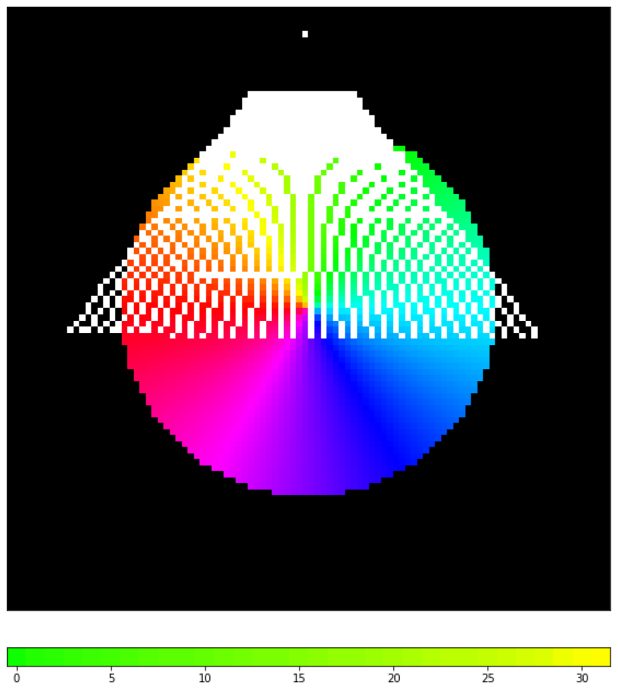

Figure 20. Camera visualization and corresponding 1D image of a 2D scene of a fully colored disk from the top.

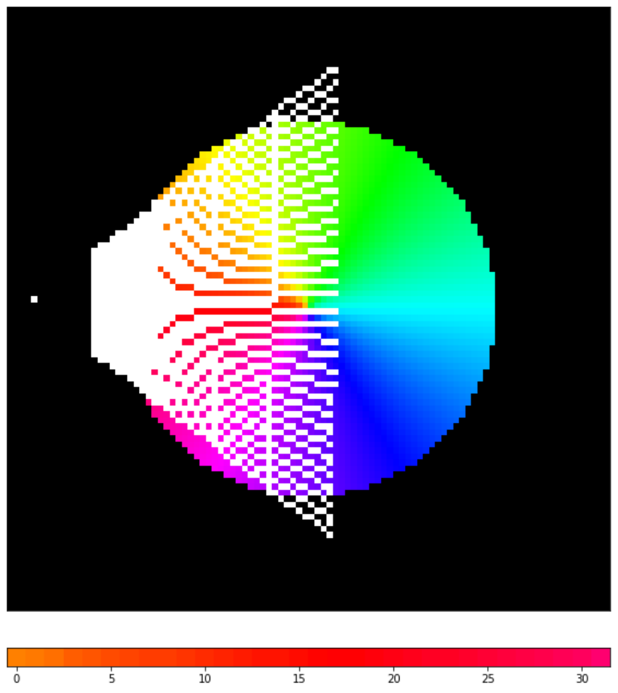

Figure 21. Camera visualization and corresponding 1D image of a 2D scene of a fully colored disk from the left.

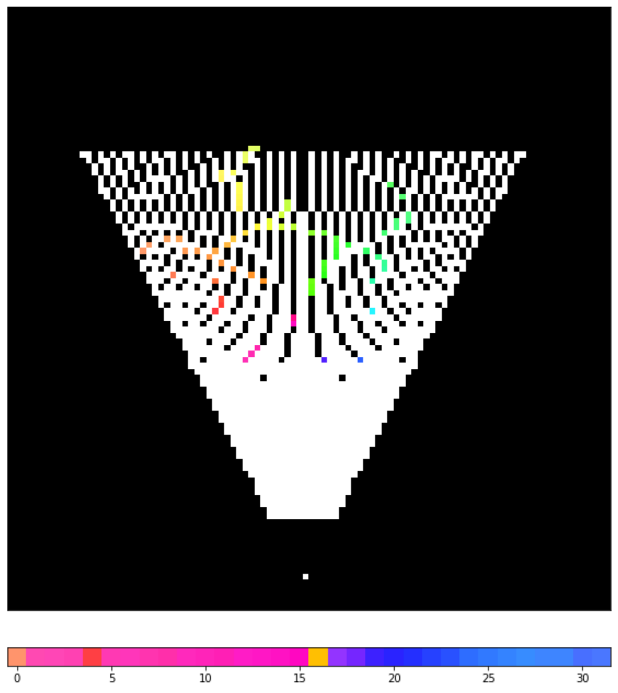

Figure 22. Camera visualization and corresponding 1D image of a graded colored complex 2D scene from the bottom.

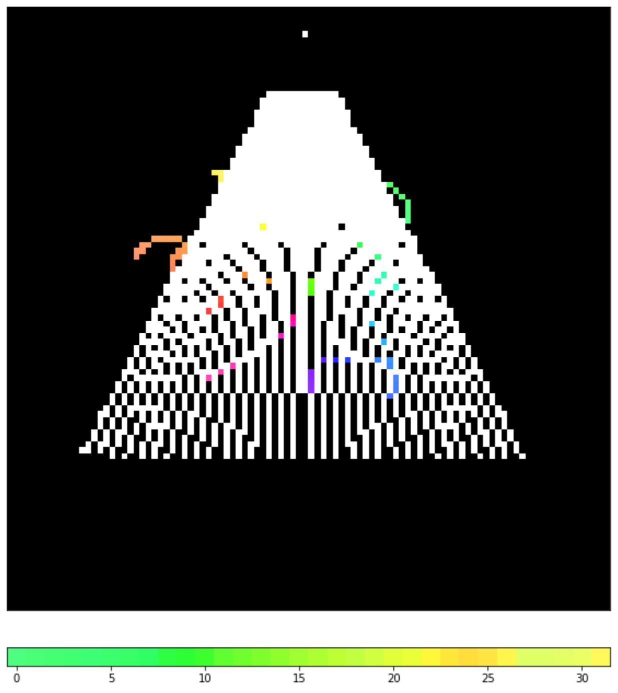

Figure 24. Camera visualization and corresponding 1D image of a graded colored complex 2D scene from the top.

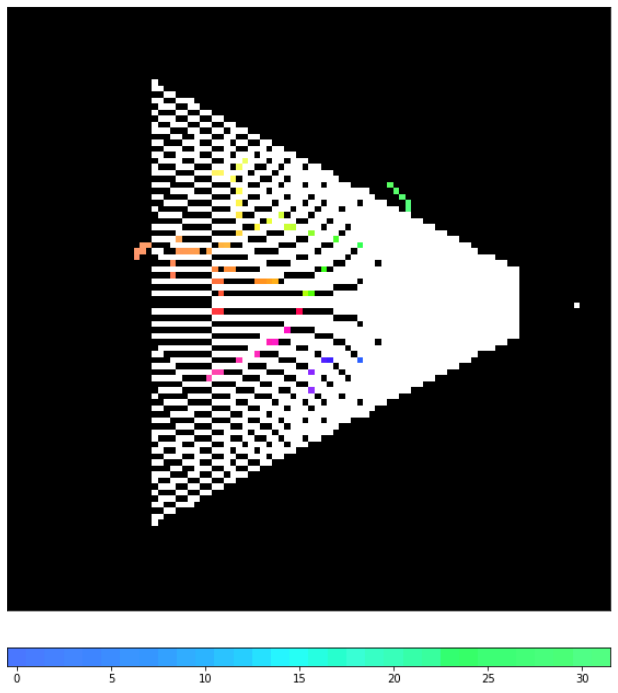

Figure 23. Camera visualization and corresponding 1D image of a graded colored complex 2D scene from the right.

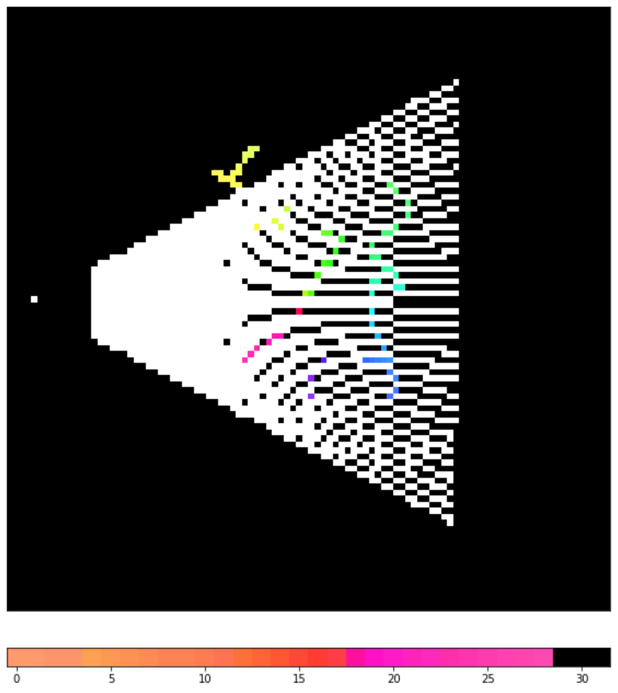

Figure 25. Camera visualization and corresponding 1D image of a graded colored complex 2D scene from the left.