CS180 • Project 4

NeuralFields

UC Berkeley • CS180 Computational Photography.

Camera Calibration and 3D Scanning

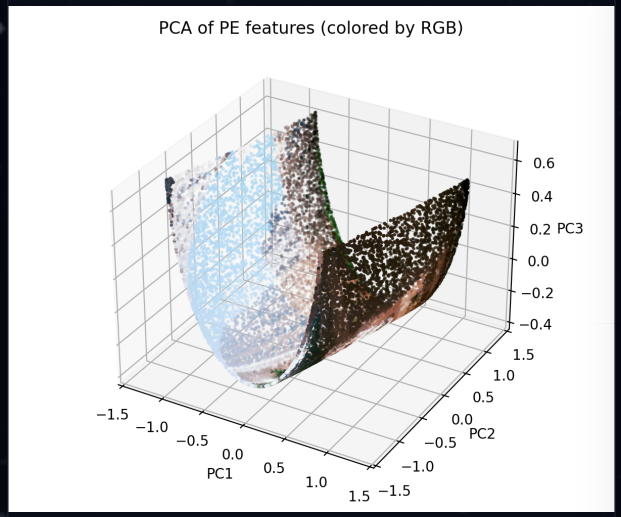
View 1 — global layout of camera poses.

View 2 — closer look at pose cluster and field of view.

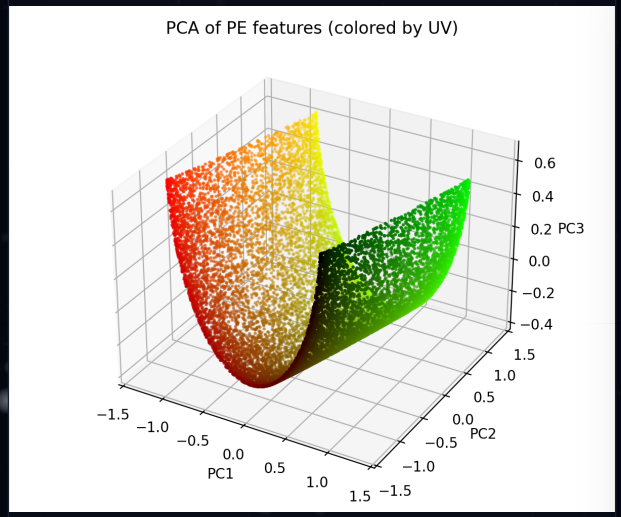
Varying Encoding Frequency

To visualize how positional encoding transforms image coordinates, each PE(u,v) vector is projected onto its top three PCA components while the maximum encoding frequency L is varied. The PCA plots reveal that positional encoding progressively twists the original two-dimensional coordinate grid into a higher-dimensional manifold as L increases. For very low frequency (L = 1), the encoded PE(u,v) vectors occupy a single smooth, bowl-shaped surface in the top three principal components. The gradient of colors from red to green along this surface indicates that the first two principal components are almost linear functions of u and v, so the encoding behaves like a gently curved embedding of the image plane with no folds: nearby pixels in image coordinates remain nearby in feature space. This behavior is typical for low-frequency positional encodings, which tend to produce a smooth two-dimensional sheet in feature space with only mild curvature from which the network can draw samples.

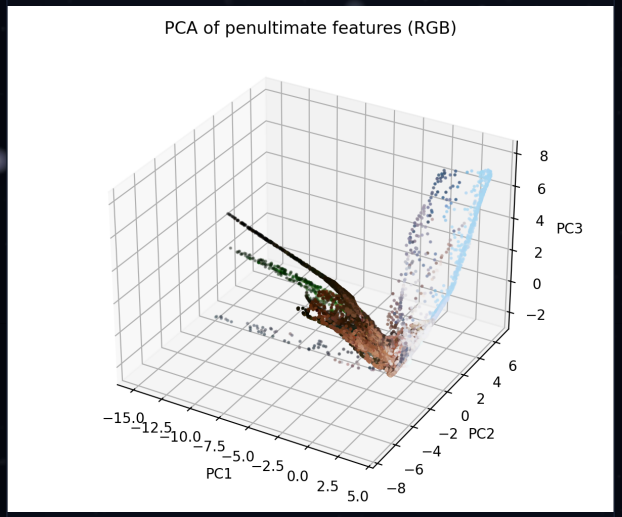
At mid-range frequencies (L = 8 and L = 12), the PCA manifolds begin to exhibit clear multi-layered structure. The surfaces no longer form a single simple bowl but instead resemble sheets that have been folded or partially rolled, with regions of the grid mapped to different heights in PC_3 even when their PC_1 and PC_2 coordinates almost overlap. The color gradients in these plots confirm that the embedding is still globally ordered by u and v, because red and green bands do not intersect randomly; however, pixels that are adjacent along one coordinate can now be separated along the third principal component whenever the corresponding sine and cosine terms switch phase. This behavior shows that mid-frequency encodings preserve global continuity but already introduce local non-linearity: the network receives a representation where local neighborhoods are stretched and sheared, making it easier to represent edges and moderate-frequency texture with a finite number of ReLU layers.



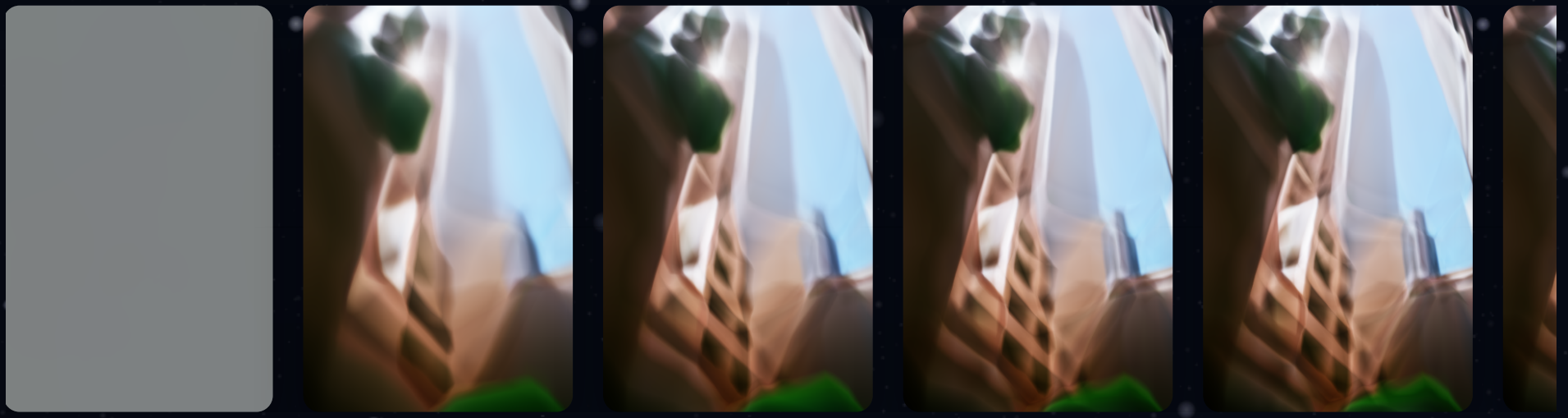
Layers: 256



lr: 1e-2



2500 iterations

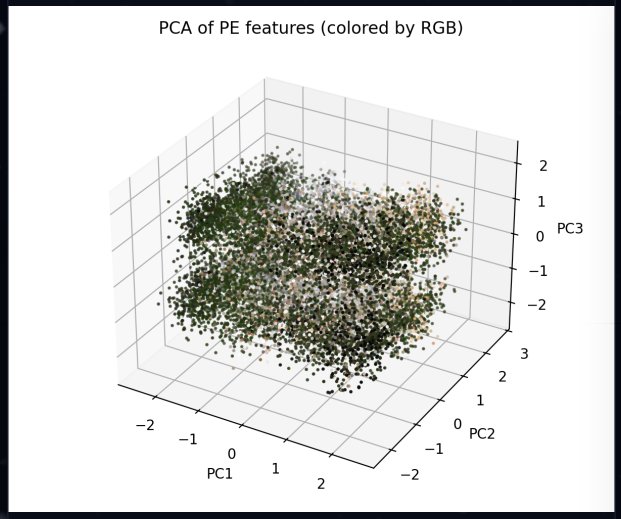


L = 1 L = 4 L = 8 L = 12 L = 16 L = 24

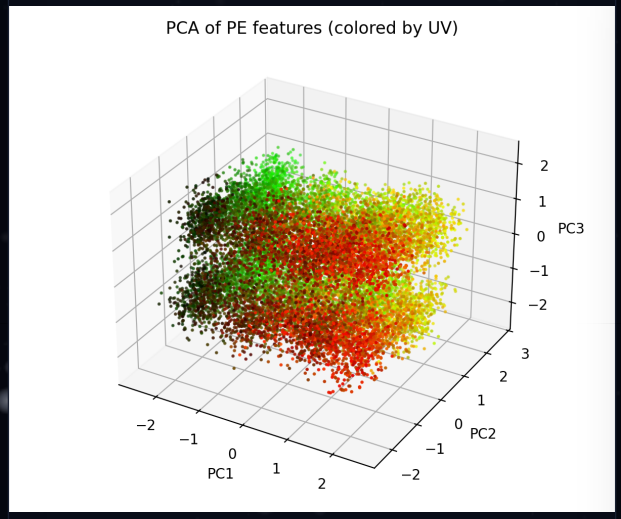
At high frequency (L = 16 and especially L = 24), the PCA geometry becomes highly oscillatory. Instead of a single surface or a gently folded ribbon, the points fill a dense volume made up of overlapping bands and sheets that weave through one another. The color pattern, which encodes the original UV coordinates, makes this structure easier to interpret: blocks of similar color form thin, separated strands in PC space, indicating that many disjoint regions of the original image are now mapped to very different directions in the encoded feature space. In other words, small changes in u or v trigger large swings in the sinusoids at high L, so nearby pixels oscillate through many nearly orthogonal basis vectors. This is precisely the effect positional encoding is designed to achieve: it gives the MLP access to very fine spatial detail by decorrelating local neighborhoods, at the expense of producing a tangled, highly folded manifold that is more challenging to visualize. Across the sweep, the PCA plots therefore illustrate a clear progression: low L preserves a simple, almost planar embedding; mid-range L produces a structured but still readable ribbon; and high L yields a volumetric, multi-sheet structure where the encoding has effectively sprinkled the image plane across a high-dimensional torus in feature space.

Varying Widths

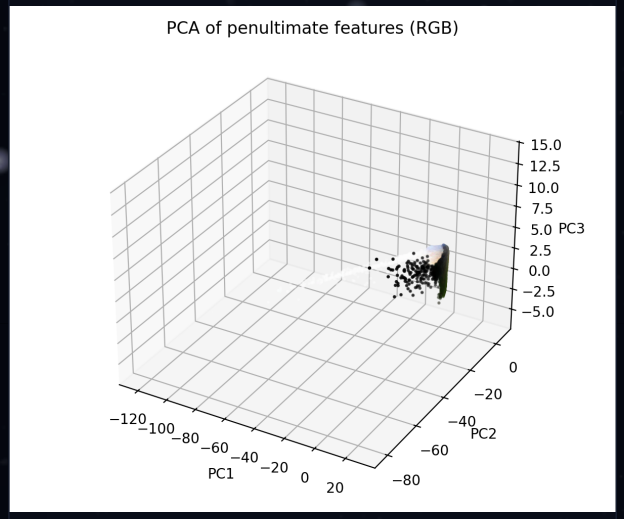
Varying the hidden width changes the geometry of the learned feature space much more than it changes the raw positional-encoding manifold. The PCA plots of PE(u,v) look essentially identical for the narrow and wide models, since both networks receive the same sinusoidal expansion of the input image coordinates; the cloud of PE features remains a dense, roughly cubic volume whose principal components are strongly aligned with u and v and only weakly correlated with color. The difference emerges at the penultimate layer. With a smaller width, the PCA of the penultimate activations forms a relatively thick, fan-shaped cloud: the first principal component correlates strongly with luminance, but a large fraction of variance is still carried by the second component, and colors and spatial regions remain partially entangled. :contentReference[oaicite:1]{index=1} In contrast, increasing the width leads to a much more anisotropic embedding in feature space. The penultimate PCA collapses into a dominant one-dimensional arc that explains the majority of the variance, with most points lying along a smooth trajectory in PC_1 while PC_2 and PC_3 only modulate local deviations.



PE • RGB (lr = 1e-2)



PE • UV (layer depth = 8)



Penultimate • RGB (iterations = 256)

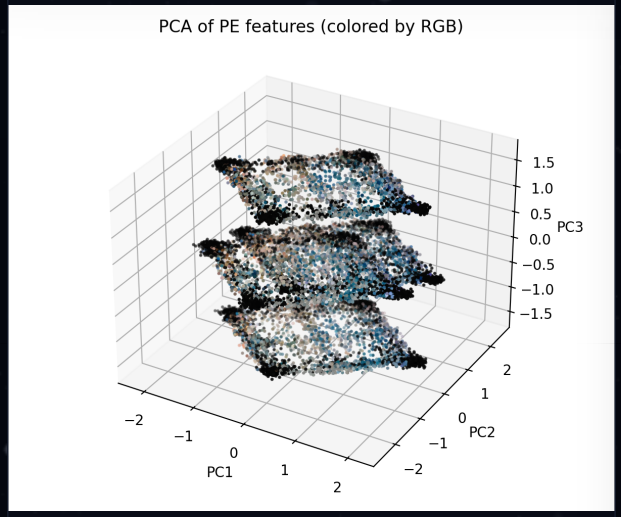
W = 64 W = 256

Visually, this appears as a compact, curved sheet that peels away from the origin and then fades into a long, low-density tail. Pixels on the object and its immediate surroundings cluster tightly along one end of this arc, while background pixels and saturated regions occupy the extended tail, indicating that the wider network has learned a more organized

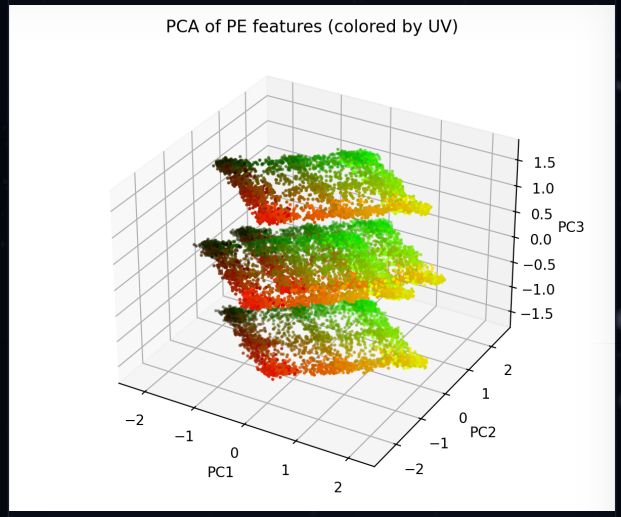
representation where luminance and large-scale structure are encoded by a single dominant direction and finer variations are relegated to orthogonal subspaces. The PSNR curves reflect this difference: both widths reach similar overall reconstruction quality, but the wider model converges to slightly higher PSNR and exhibits a smoother, more monotonic improvement over training, consistent with its more structured internal feature manifold.

Final Reconstruction on Chosen Scene

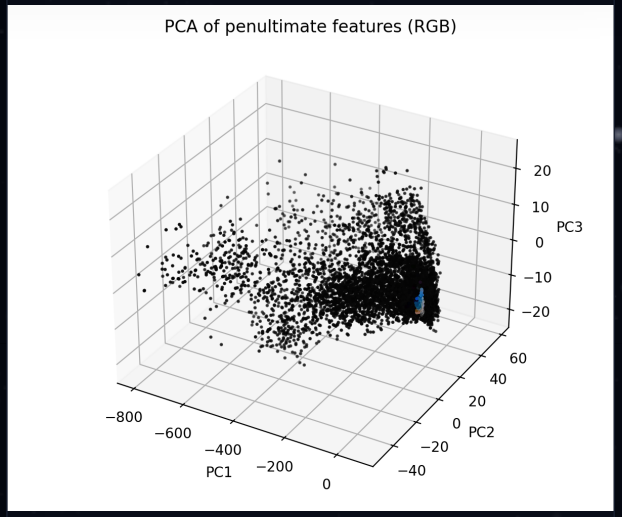
The final Jupiter experiment can be interpreted by reading the PCA plots and PSNR curves together. The PCA of PE features (colored by UV) shows that the raw positional encodings still occupy a fairly regular three-sheet manifold in the top three principal components. The smooth transition from red to green across this structure indicates that the first principal components remain strongly aligned with the underlying latitude and longitude on the sphere, so even at higher frequency the encoding preserves a globally ordered map of spatial position. When the same PE features are colored by RGB instead of UV, the color pattern spreads almost uniformly through this volume, which confirms that the positional encoding itself does not yet reflect any semantic grouping by appearance; the geometry at this stage is dominated by coordinate-based oscillations rather than by the planet's texture.



PE • UV (top three principal components)

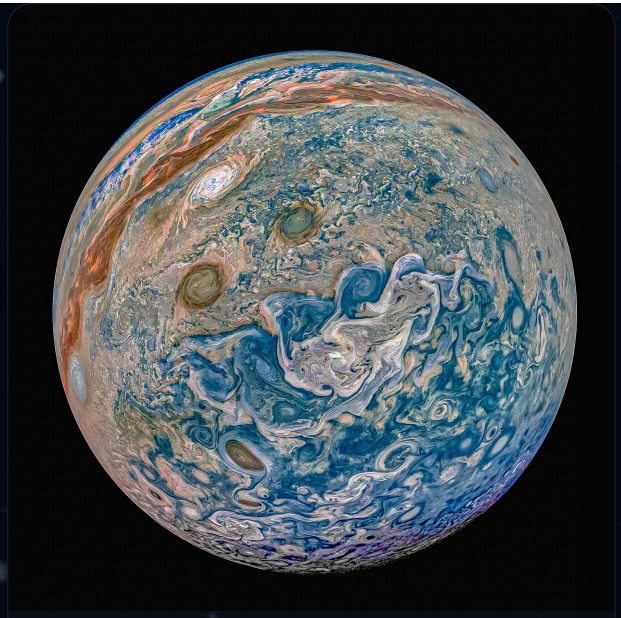


PE • RGB (top three principal components)

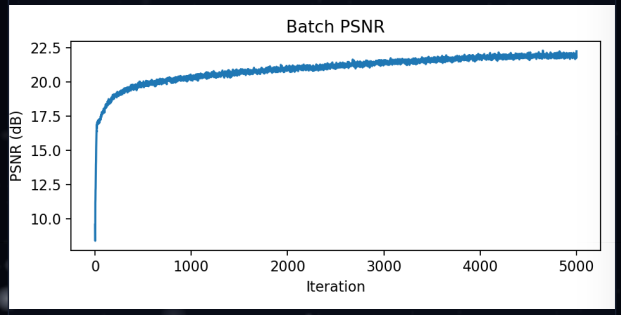


Penultimate • RGB (learned feature geometry)

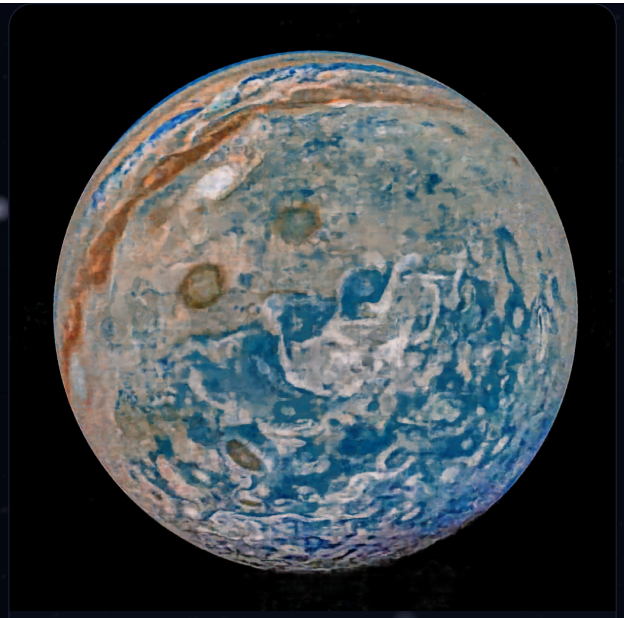
The PCA of the penultimate features tells a very different story. After passing through the MLP, the encoded points are no longer filling a symmetric three-sheet volume; instead, they collapse into a highly elongated plume concentrated near one side of PC space, with most variance lying along a single direction. Points corresponding to bright clouds, blue storms, and the dark background are separated along this main axis, while the secondary axes make only small corrections. This pattern indicates that the network has reorganized the high-dimensional PE basis into a much more compact, appearance-aligned representation in which luminance and large-scale color structure are encoded by one dominant feature dimension, and finer chromatic and textural differences are stored in orthogonal directions. In other words, the penultimate layer behaves like a learned color-space tailored to the Jupiter image rather than a direct reflection of the original (u,v) coordinates.



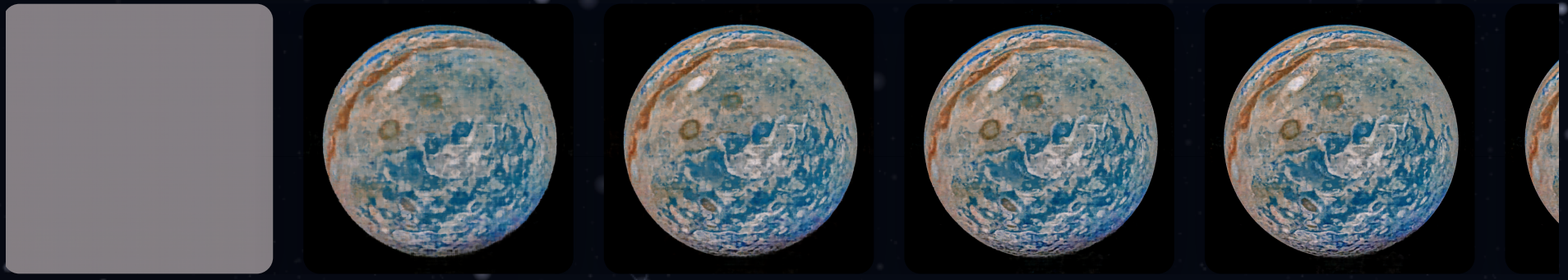
Ground-truth view (input photograph)



Batch PSNR over training iterations



Final NeRF reconstruction (e.g., 5k iters, L = 16, width = 256)



Final Result

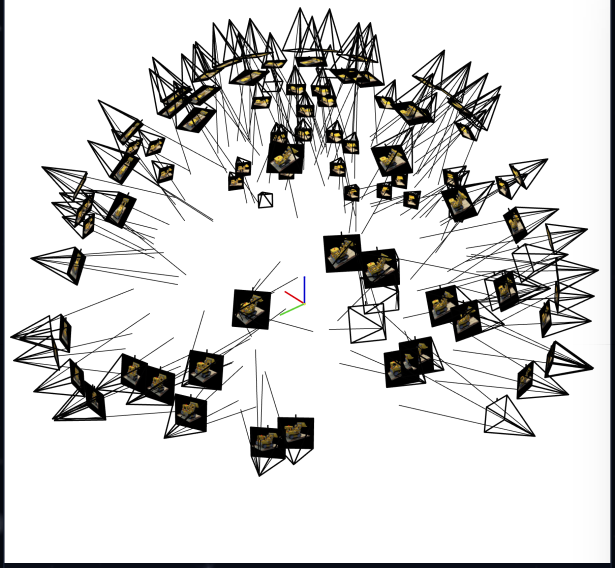
THE PSNR CURVES QUANTIFY HOW QUICKLY THIS REPRESENTATION EMERGES OVER TRAINING. THE BATCH PSNR

rises from single-digit values to around 20 dB within a few hundred iterations, reflecting rapid fitting of the global brightness and broad banding patterns. The full-image PSNR curve shows a large jump between the initial and first checkpoint renders, then a slower, almost linear increase that asymptotically approaches a mid-teens or low-twenties value by the end of training. The corresponding renderings match this trajectory: the very first output is nearly uniform, early snapshots show only faint hints of the dominant bands and storms, and later snapshots progressively sharpen the cloud belts, vortices, and color gradients until the final image closely resembles the ground truth while still smoothing the finest turbulent eddies. Taken together, the PCA plots and PSNR curves demonstrate that the model begins from a purely coordinate-driven encoding, then gradually learns an appearance-driven feature space and uses it to reconstruct both the coarse spherical structure and much of the intricate atmospheric detail of the original Jupiter photograph.

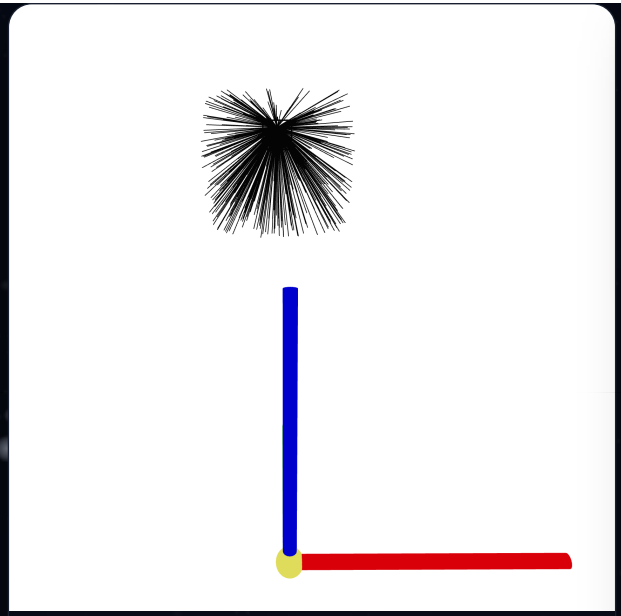
NeRF Pipeline

The first stage of the pipeline translates image pixels into rays in 3D space. For each training step, a batch of pixel locations is drawn from the current set of images. Instead of treating each pixel in isolation, the implementation works entirely in batches so that thousands of rays can be computed in one vectorized pass on the GPU. For every pixel, its horizontal and

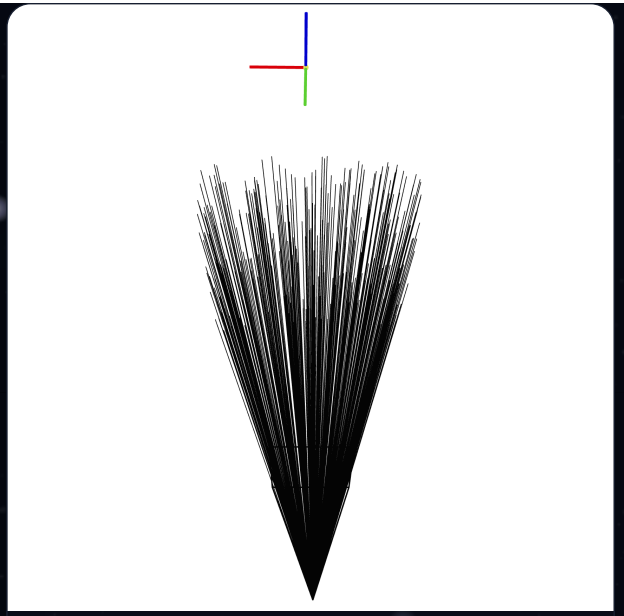
vertical coordinates are first shifted so that they are measured relative to the center of the image; this ensures that a pixel exactly in the middle of the frame produces a ray pointing straight through the optical axis rather than being biased by the top-left origin of image coordinates. These centered coordinates are then divided by the focal length to convert them into directions in the camera's local coordinate system, following the standard pinhole-camera model in which the focal length controls the spread of rays. Each camera in the training set is described by a rigid transform from camera space to world space; for every ray in the batch, the implementation looks up the corresponding camera's rotation and translation and uses them to rotate the local direction into the global reference frame while placing the ray's origin at the camera center. Finally, all directions are normalized to unit length so that subsequent sampling along the ray can be expressed purely in terms of distances from the camera. The result of this stage is a batch of ray origins and matching direction vectors that describe exactly which 3D lines will be probed in the volume.



All calibrated camera frustums around the Lego scene.

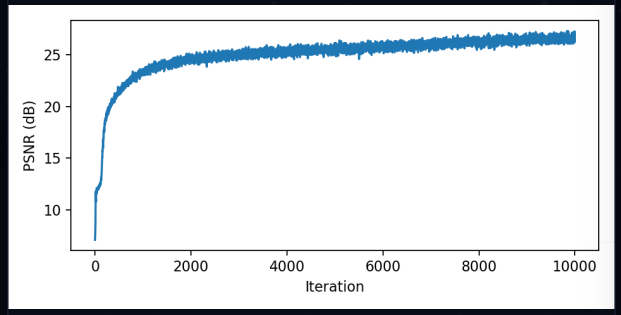


Random training rays emitted from one camera in world space.



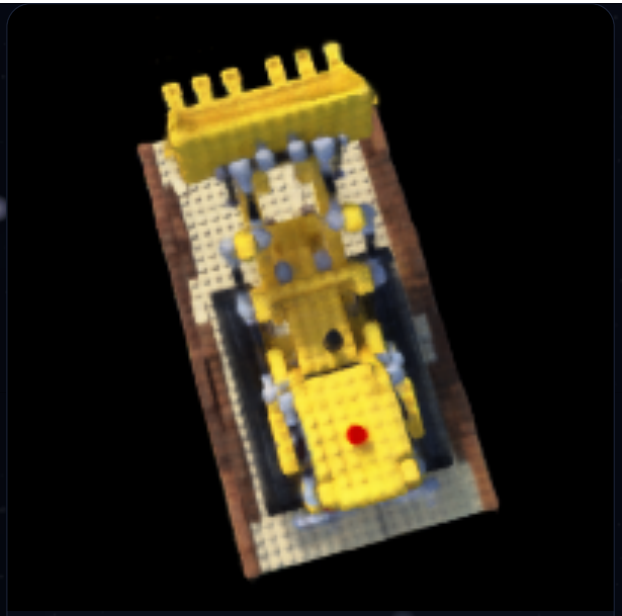
Stratified sample points along each ray, ready for NeRF evaluation.

Once the rays exist in world space, the next stage chooses where along each ray to query the neural field. The implementation uses a stratified scheme: it begins by placing a fixed number of sample depths between a near and a far bound that cover the region where the scene is expected to lie. Instead of always using the same depth values, each interval between two neighboring depths is randomly jittered on every training iteration. This produces slightly different sampling positions from step to step, which reduces aliasing artifacts and encourages the learned density field to be smooth rather than fitting to a rigid grid of points. For each ray, these scalar depths are then converted into 3D sample locations by adding the origin plus the direction vector scaled by each depth value. The outcome of this stage is a dense batch of points in space, all organized per ray, ready to be evaluated by the neural network.



PSNR over training iterations (Lego 200×200)

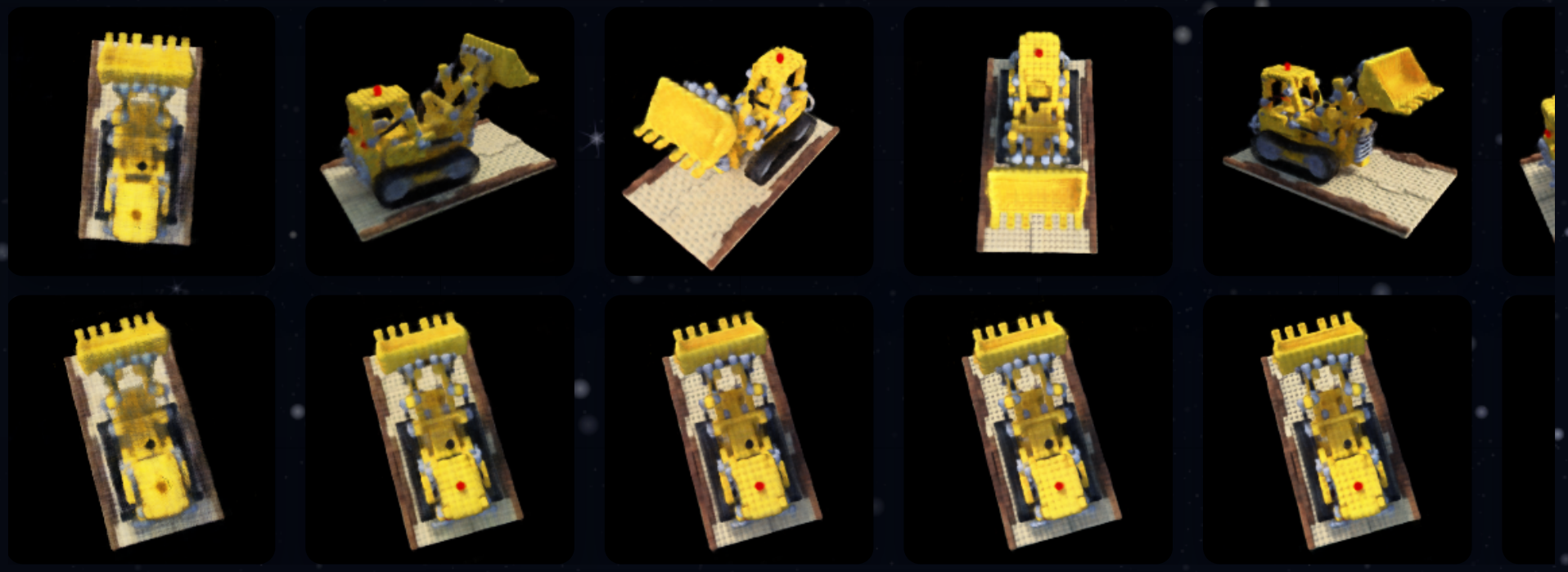
Final NeRF reconstruction of a training view (iteration 10 000)



Final NeRF reconstruction of a held-out validation view

Feeding data into this process is handled by a streaming dataset tailored to NeRF's requirements. When training begins, the dataset loader reads a compact file that contains the entire scene: a stack of RGB images, the corresponding camera-to-world transforms, and a shared focal length consistent with the resolution of those images. Rather than iterating over the images in a fixed order, the loader behaves like an infinite stream: each time the trainer asks for a new batch, it randomly chooses image indices and pixel coordinates, effectively sampling rays uniformly across all views and across the whole image plane. For every sampled triplet of (image index, horizontal pixel, vertical pixel), the loader immediately retrieves the true color at that pixel from the preloaded image tensor and retrieves the pose of the camera that observed it. It then passes these sampled image coordinates and camera transforms to the ray-generation stage, which produces the matching ray origins and directions. In this way, each training batch consists

of a large set of randomly selected rays and their ground-truth RGB values, allowing the network to gradually see all viewpoints and all parts of the scene over the course of training without ever needing to materialize the entire ray bundle in memory at once.

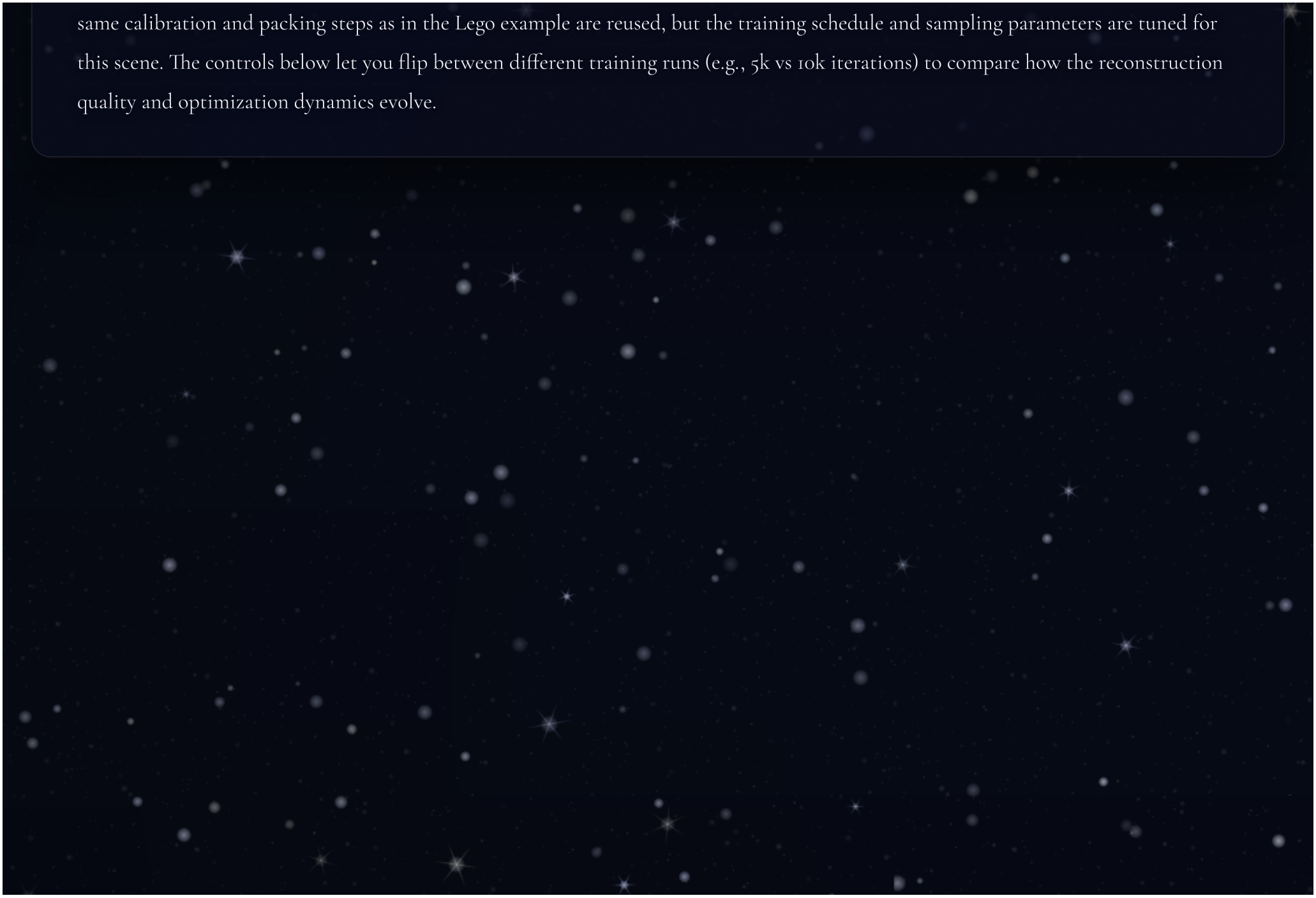


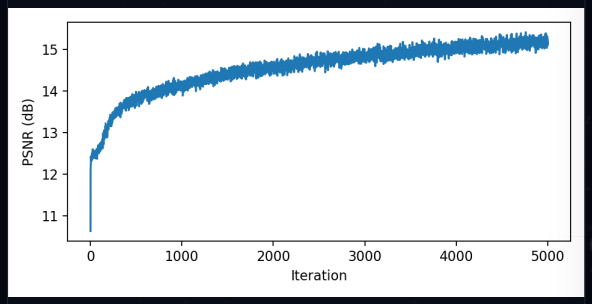
NeRF orbit of the reconstructed scene (10k iterations), illustrating the full pipeline in motion.

Training on My Own Dataset

This experiment applies the NeRF pipeline to a custom object captured from multiple viewpoints. The

same calibration and packing steps as in the Lego example are reused, but the training schedule and sampling parameters are tuned for this scene. The controls below let you flip between different training runs (e.g., 5k vs 10k iterations) to compare how the reconstruction quality and optimization dynamics evolve.

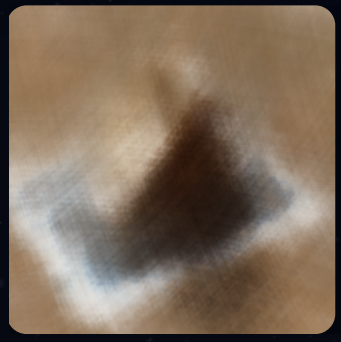


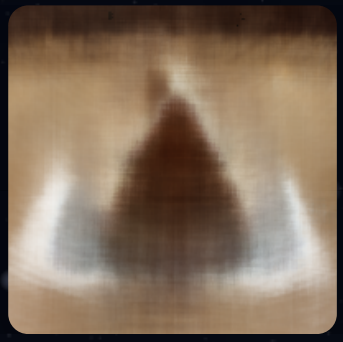


PSNR / loss curve (5k iterations, lr = 5e-4, 64 samples/ray).

Training view reconstruction at 5k iterations.

Held-out validation view at 5k iterations.

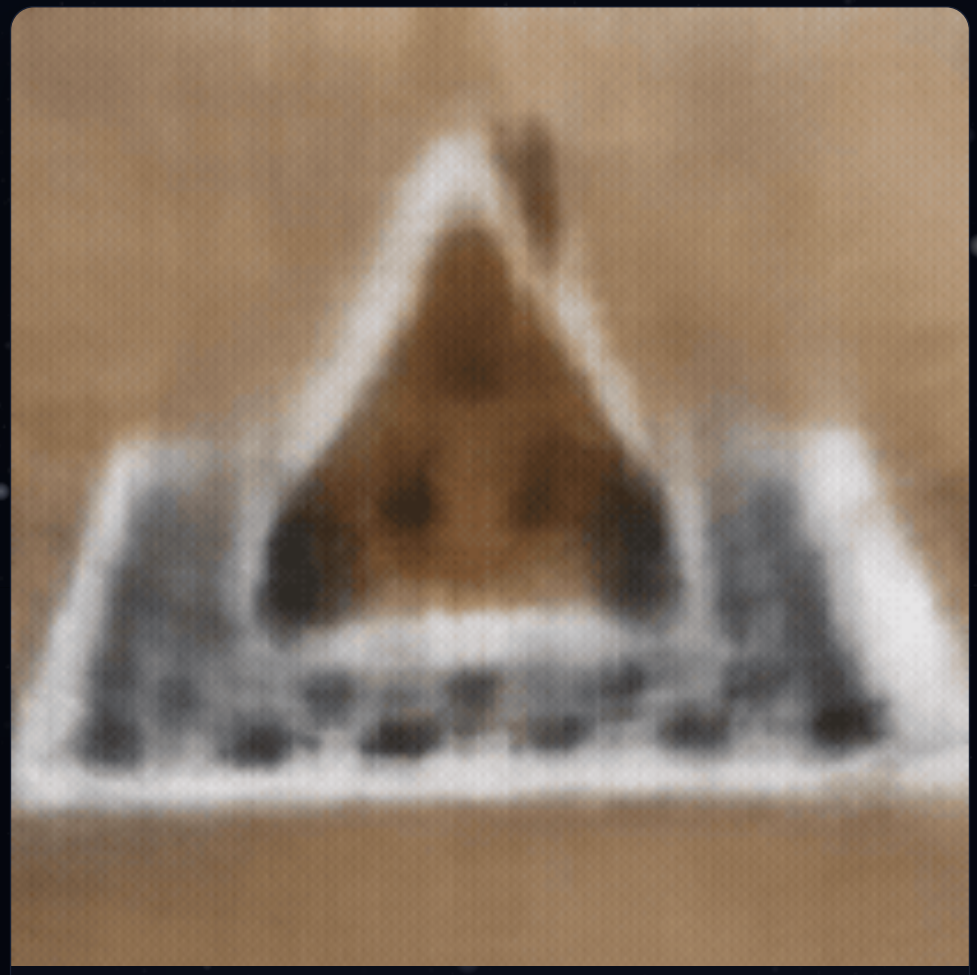




5k iters

10k iters

The two training runs illustrate how additional optimization time mainly affects fine detail rather than the overall layout of the scene. In the 5k-iteration run, the batch PSNR curve rises rapidly over the first few hundred iterations, jumping from roughly 10 dB into the mid-teens, and then increases more gradually as training proceeds. The sequence of intermediate training renders reflects this shape: the image at iteration 0 is an amorphous brown blur, by 1k iterations the triangular object and its supporting board begin to emerge, and by 5k iterations the overall silhouette and shading are stable, with the checkerboard pattern on the board only faintly visible. The corresponding validation views show a similar progression: early frames barely distinguish foreground from background, whereas the 5k snapshot already reconstructs the basic geometry and albedo from a held-out camera, albeit with soft edges and residual mottling in the background.



Orbit of the trained NeRF on the custom scene, showing novel synthesized views.

The 10k-iteration run, trained with the same base learning rate but a cosine decay down to 5e-6 after 5k steps, behaves differently after the midpoint. Its PSNR curve closely tracks the 5k curve up to approximately iteration 5000, then continues to climb for several thousand more iterations, adding another one to two decibels of improvement before flattening. Qualitatively, the renders at 6k and 8k iterations show the triangular peak and its shadow sharpening, the edge between the board and the table becoming crisper, and the subtle grid pattern on the calibration board becoming easier to discern. By iteration 10k, the training view has noticeably fewer low-frequency artifacts and less noise in the background, and the validation view reveals cleaner lines along the object's silhouette while preserving the same global appearance as in the shorter run. Together, these comparisons indicate that five thousand iterations are sufficient to capture the overall shape and color of the scene, while a longer schedule primarily refines small-scale structure and reduces residual blur, yielding smoother textures and slightly higher PSNR at the cost of additional compute.