# NeRF Explained, really.

Roger Lam
lamroger.com

## Neural Radiance Fields convert 2D images to 3D

From **a set of images**, each with a **spatial location (x, y, z) and a viewing direction (θ, φ)**, you can generate a image from an entirely new location and direction.

It gets details well too. Think shadows, textures, transparency.

# Train a NN with ~100 images all around an object

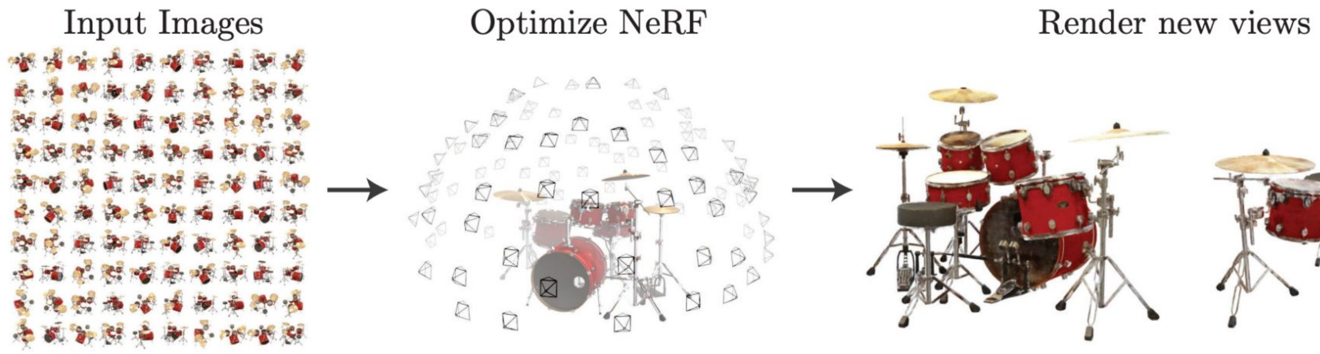


Fig. 1: We present a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. We use techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, we visualize the set of 100 input views of the synthetic *Drums* scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from our optimized NeRF representation.

## Now we have a function that outputs color and density

To generate pixels in our image, we sample across a direction.

For each sample, we know a density. High density = there's something there.

If density is high, draw on our 2D canvas.

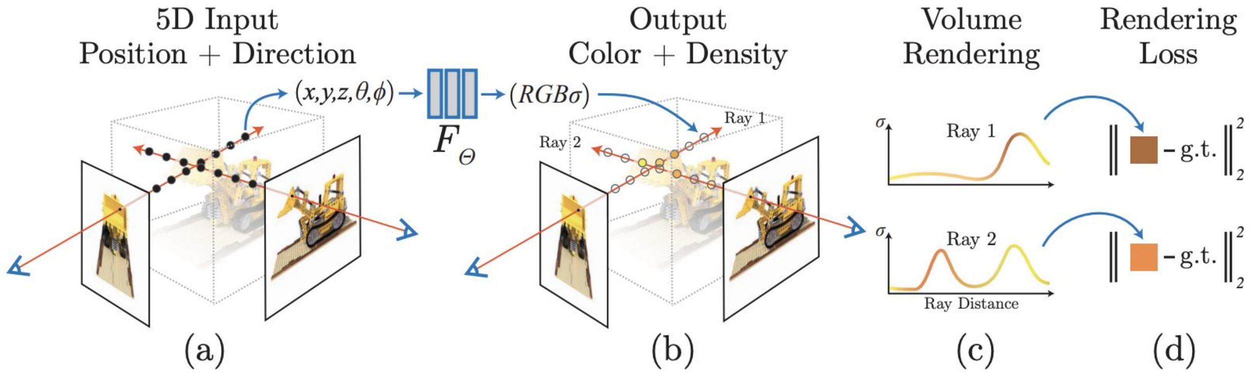


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

# We use differentiable volume and color curves!

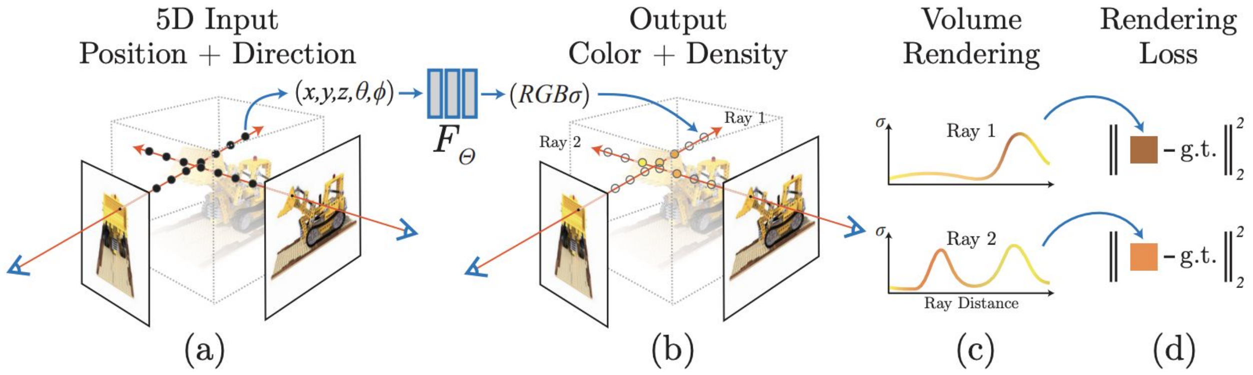
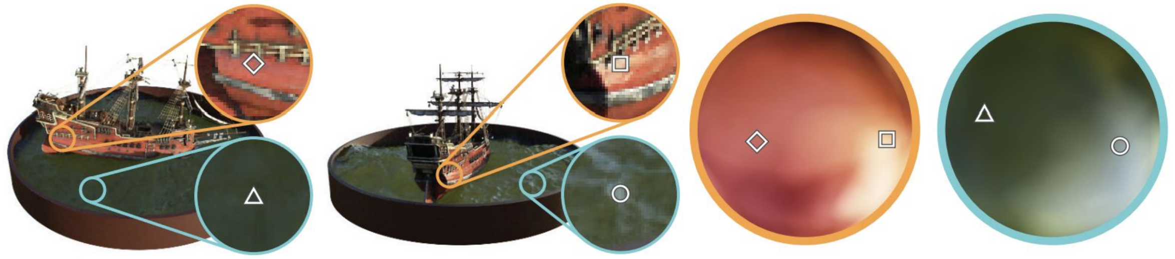


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

# Same spot, different angle, continuous change



(a) View 1 (b) View 2 (c) Radiance Distributions

Fig. 3: A visualization of view-dependent emitted radiance. Our neural radiance field representation outputs RGB color as a 5D function of both spatial position **x** and viewing direction **d**. Here, we visualize example directional color distributions for two spatial locations in our neural representation of the *Ship* scene. In (a) and (b), we show the appearance of two fixed 3D points from two different camera positions: one on the side of the ship (orange insets) and one on the surface of the water (blue insets). Our method predicts the changing specular appearance of these two 3D points, and in (c) we show how this behavior generalizes continuously across the whole hemisphere of viewing directions.

## With a few more techniques, we get amazing detail

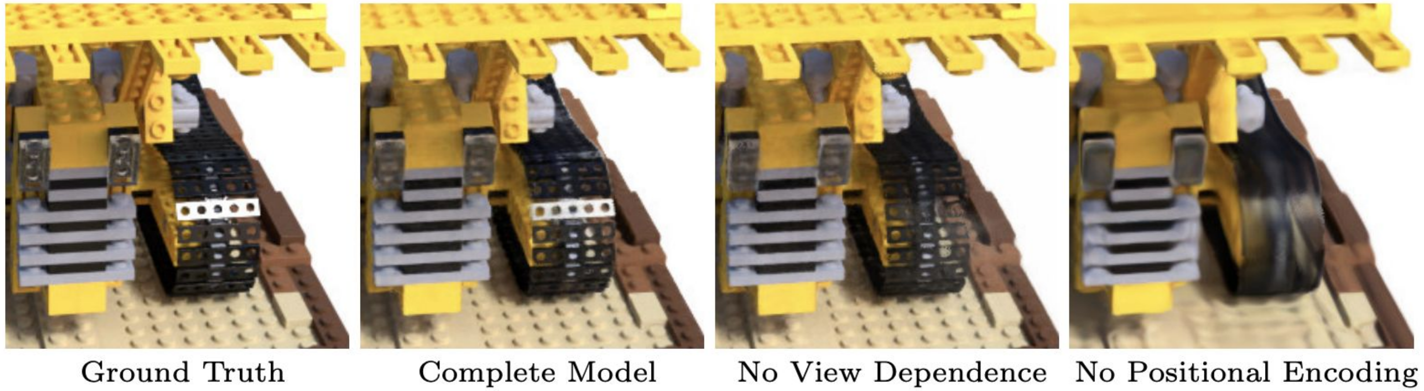
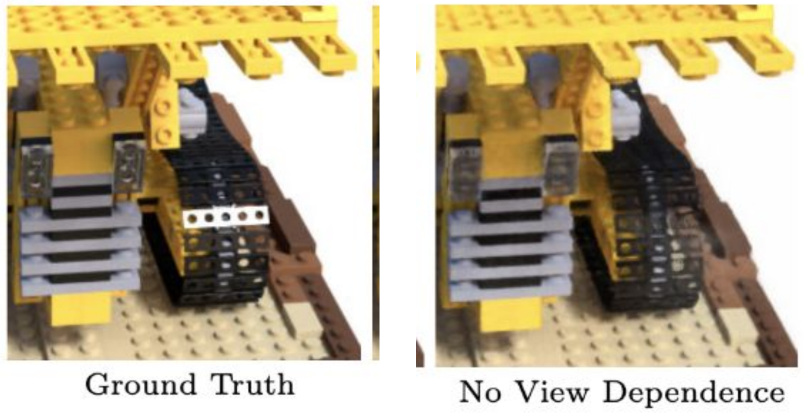


Fig. 4: Here we visualize how our full model benefits from representing view-dependent emitted radiance and from passing our input coordinates through a high-frequency positional encoding. Removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the positional encoding drastically decreases the model's ability to represent high frequency geometry and texture, resulting in an oversmoothed appearance.

## View Dependence = only using (x, y, z) in training

If we don’t include the angle in our training data, we lose reflective properties.

Kinda makes sense. If we only look head on, most things don’t reflect. Then we overfit on things not reflecting.

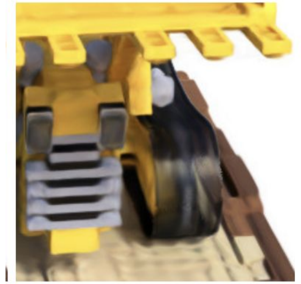


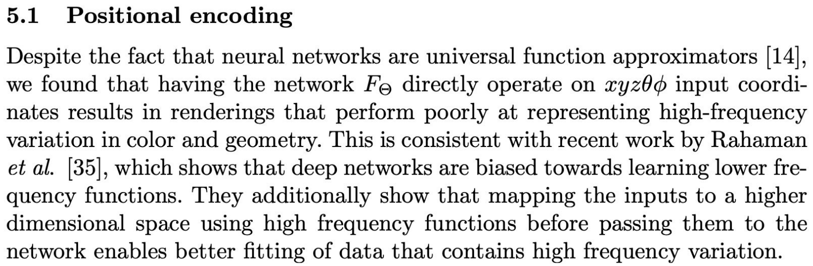
# Positional Encoding maps 5D coordinates to higher dims

Without positional encoding, things are still kinda blurry.

Transformers use it to notice position of text.

NeRF uses it to notice more fine details. More squigglies, more details.


No Positional Encoding



### 5.1 Positional encoding

Despite the fact that neural networks are universal function approximators [14], we found that having the network $F_\Theta$ directly operate on $xyz\theta\phi$ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry. This is consistent with recent work by Rahaman *et al.* [35], which shows that deep networks are biased towards learning lower frequency functions. They additionally show that mapping the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation.

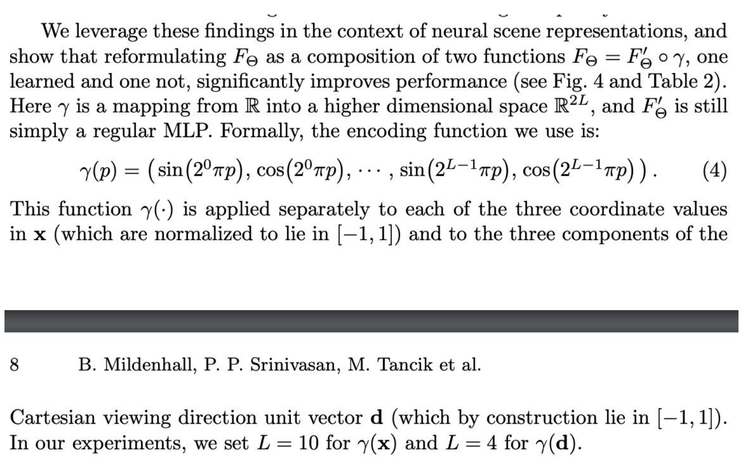
# Map values into a collection of waves

They set L to 10 for **x** (coordinate system) and L to 4 for **d** (direction).

So each x, y, z turns into 10 * 2 values.

Each theta and phi turns into 4 * 2 value.

Reminds me of music gen adding variation



We leverage these findings in the context of neural scene representations, and show that reformulating $F_\Theta$ as a composition of two functions $F_\Theta = F'_\Theta \circ \gamma$, one learned and one not, significantly improves performance (see Fig. 4 and Table 2). Here $\gamma$ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$, and $F'_\Theta$ is still simply a regular MLP. Formally, the encoding function we use is:

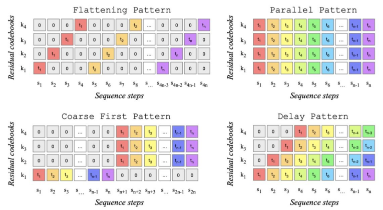
$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right). \tag{4}$$

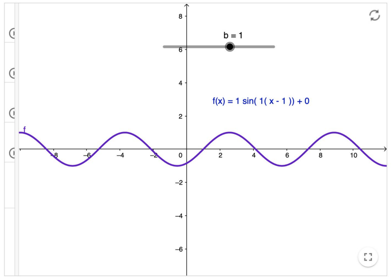
This function $\gamma(\cdot)$ is applied separately to each of the three coordinate values in $\mathbf{x}$ (which are normalized to lie in $[-1, 1]$) and to the three components of the

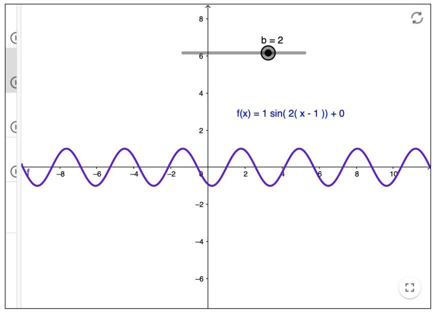
8 B. Mildenhall, P. P. Srinivasan, M. Tancik et al.

Cartesian viewing direction unit vector $\mathbf{d}$ (which by construction lie in $[-1, 1]$). In our experiments, we set $L = 10$ for $\gamma(\mathbf{x})$ and $L = 4$ for $\gamma(\mathbf{d})$.

MusicGen uses a single end-to-end model



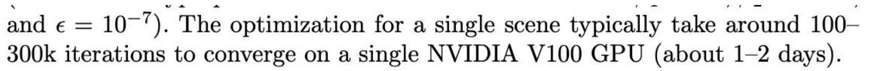




## Took 1-2 Days on a V100 GPU

Not bad. You don't need a giant cluster.

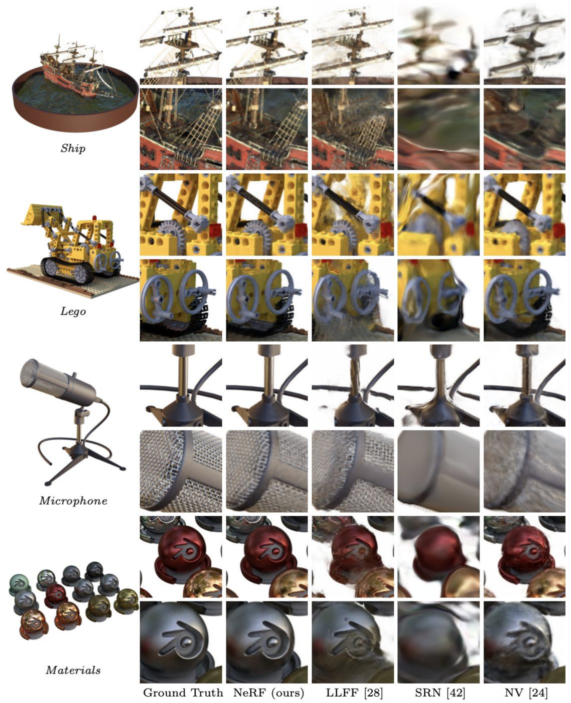
Can expect v2 and more to be better.



and $\epsilon = 10^{-7}$). The optimization for a single scene typically take around 100–300k iterations to converge on a single NVIDIA V100 GPU (about 1–2 days).

## Better than alternatives!

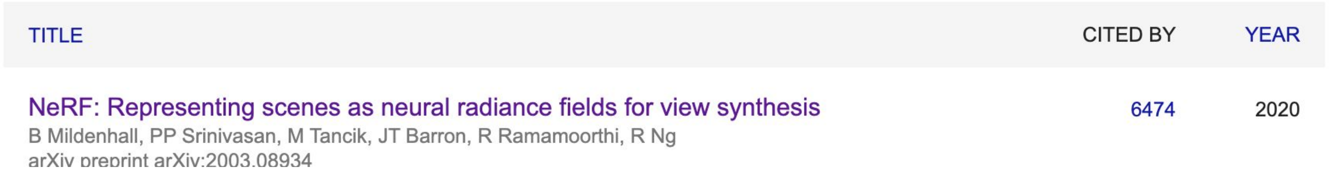
Get details a lot better.



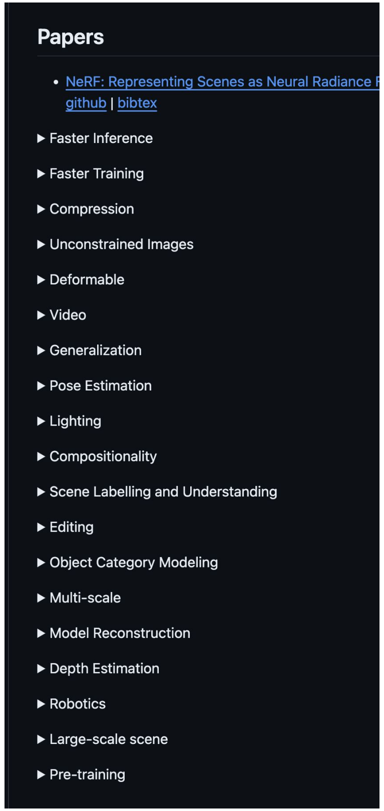
## Lots of work going into improving NeRF

Each section has 1-20 additional papers since 2020. https://github.com/awesome-NeRF/awesome-NeRF

6.4k citations since 2020.



| TITLE | CITED BY | YEAR |
|---|---|---|
| NeRF: Representing scenes as neural radiance fields for view synthesis<br>B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, R Ng<br>arXiv preprint arXiv:2003.08934 | 6474 | 2020 |



Papers

- NeRF: Representing Scenes as Neural Radiance F
  github | bibtex

▶ Faster Inference
▶ Faster Training
▶ Compression
▶ Unconstrained Images
▶ Deformable
▶ Video
▶ Generalization
▶ Pose Estimation
▶ Lighting
▶ Compositionality
▶ Scene Labelling and Understanding
▶ Editing
▶ Object Category Modeling
▶ Multi-scale
▶ Model Reconstruction
▶ Depth Estimation
▶ Robotics
▶ Large-scale scene
▶ Pre-training

## Resources

NeRF Paper

NeRF paper talk from coauthor

NeRF explained Youtube vid

More NeRF papers repo