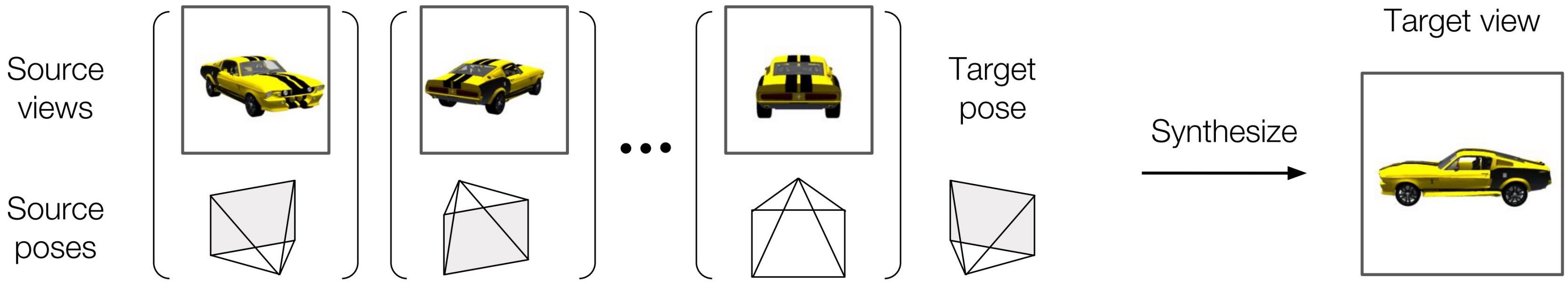
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Presenter: Elvin Yang

2022-09-06

Background: Novel View Synthesis

- The process of using images of a scene and their camera poses to synthesize new images of the scene at arbitrary camera poses.



- Subset of *Neural Rendering*, “deep image or video generation approaches that enable explicit or implicit control of scene properties”. (Ayush Tewari et. al.)
  - E.g. illumination, camera parameters, pose, geometry, appearance, and semantic structure

Image: “Multi-view to Novel View: Synthesizing Novel Views with Self-Learned Confidence” (Sun et al., 2018)

Applications in Robotics

- Visual Forward Models
  - If a robot can predict what it will be looking at, it can plan more effective actions and paths
  - Motion Planning
  - Visual Navigation
- Camera Pose Estimation
  - “Inverted” Neural Radiance Fields
- Improving Object Recognition

Image: “Vision-Only Robot Navigation in a Neural Radiance World” (Adamkiewicz, Chen, et al., 2022)
Image: “Dex-NeRF: Using a Neural Radiance Field to Grasp Transparent Objects” (Ichnowski, Avigal, et al, 2021)

Related Work

- Neural Volumes (NV), 2019
  - Deep 3D convolutional network architecture
  - Predicts a fixed-size discretized voxel grid
  - Limitation: discrete voxel grids do not scale well and lose fine detail at high resolutions
  - Limitation: requires a bounded volume and knowledge of the background
- Scene Representation Networks (SRN), 2019
  - Uses a recurrent neural network to model a rendering function
  - Limited to simple shapes with low geometric complexity
- Local Light Field Fusion (LLFF), 2019
  - 3D convolutional network architecture
  - Predicts multiplane images and fuses them to create new views
  - Fast to train (<10 minutes) at the cost of large storage requirements (~GB for each scene)

NeRF: Key Insights

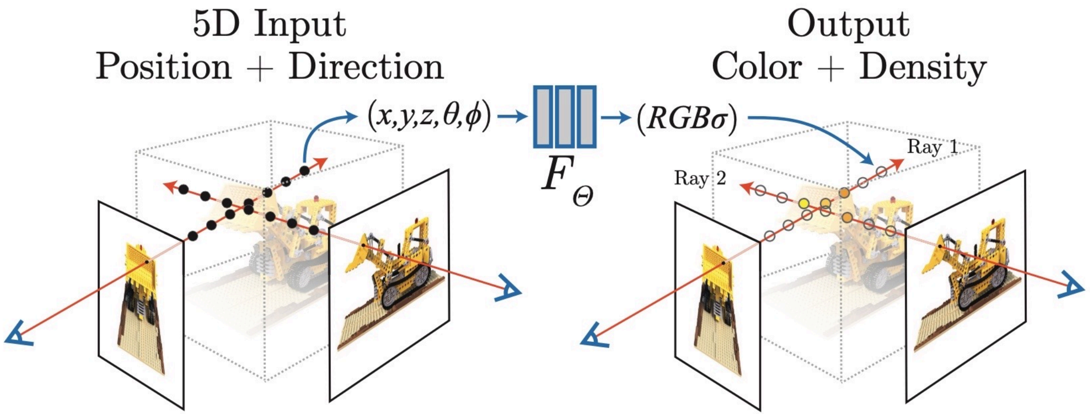
- Represent static scenes in a **continuous** space.
- Encode a continuous radiance field **within the parameters** of a fully-connected neural network.
- Regress directly from viewing location and direction to color and transparency

Problem Setting

- Given a dataset containing RGB images of a static scene, their corresponding camera poses, and intrinsic parameters,
- Predict the color and volume density for every viewing location and direction

Inputs:
- x, y, z Target position of the camera
- θ, ϕ Target orientation of the camera

Outputs:
- c = (R, G, B) Color
- σ Volume density



NeRF: Volume Rendering

- Generating a view from NeRF requires rendering all rays that pass through each pixel of the desired virtual camera

$$\underbrace{C(\mathbf{r})}_{\text{Expected color of a camera ray}} = \int_{t_n}^{t_f} T(t)\underbrace{\sigma(\mathbf{r}(t))}_{\text{Predicted Volume Density}}\underbrace{\mathbf{c}(\mathbf{r}(t), \mathbf{d})}_{\text{Predicted Color}}dt\,, \text{ where } \underbrace{T(t)}_{\text{Probability that nothing has blocked the ray up to this point}} = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

- Numerically estimated using quadrature and stratified sampling

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \underbrace{T_i(1 - \exp(-\sigma_i\delta_i))}_{\text{The relative contribution of this segment}}\underbrace{\mathbf{c}_i}_{\text{Predicted Color}}\,, \text{ where } \underbrace{T_i}_{\text{Probability that nothing has blocked the ray up to this point}} = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$$

- Differentiable: allows optimization using gradient descent

NeRF: Hierarchical Sampling

- Problem: It is inefficient to integrate over empty and occluded spaces in a scene
- Solution: Allocate samples proportionally to their expected effect on the final rendering.

- Evaluate a “coarse” network on a set of N_c locations along a ray to produce a PDF along the ray

- Evaluate a “fine” network on N_c and a second a set of N_f locations sampled from the PDF

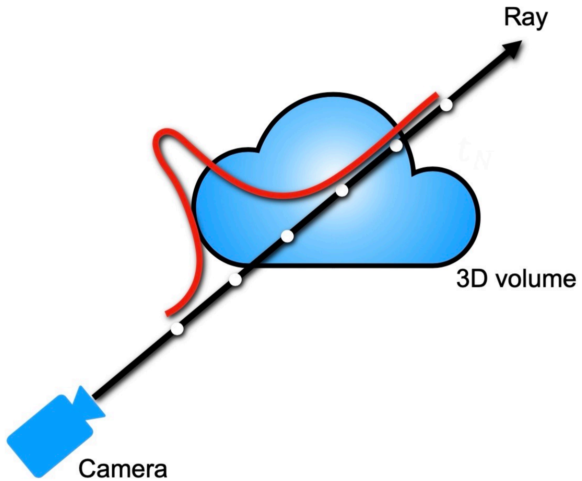


Image: Jonathan Barron

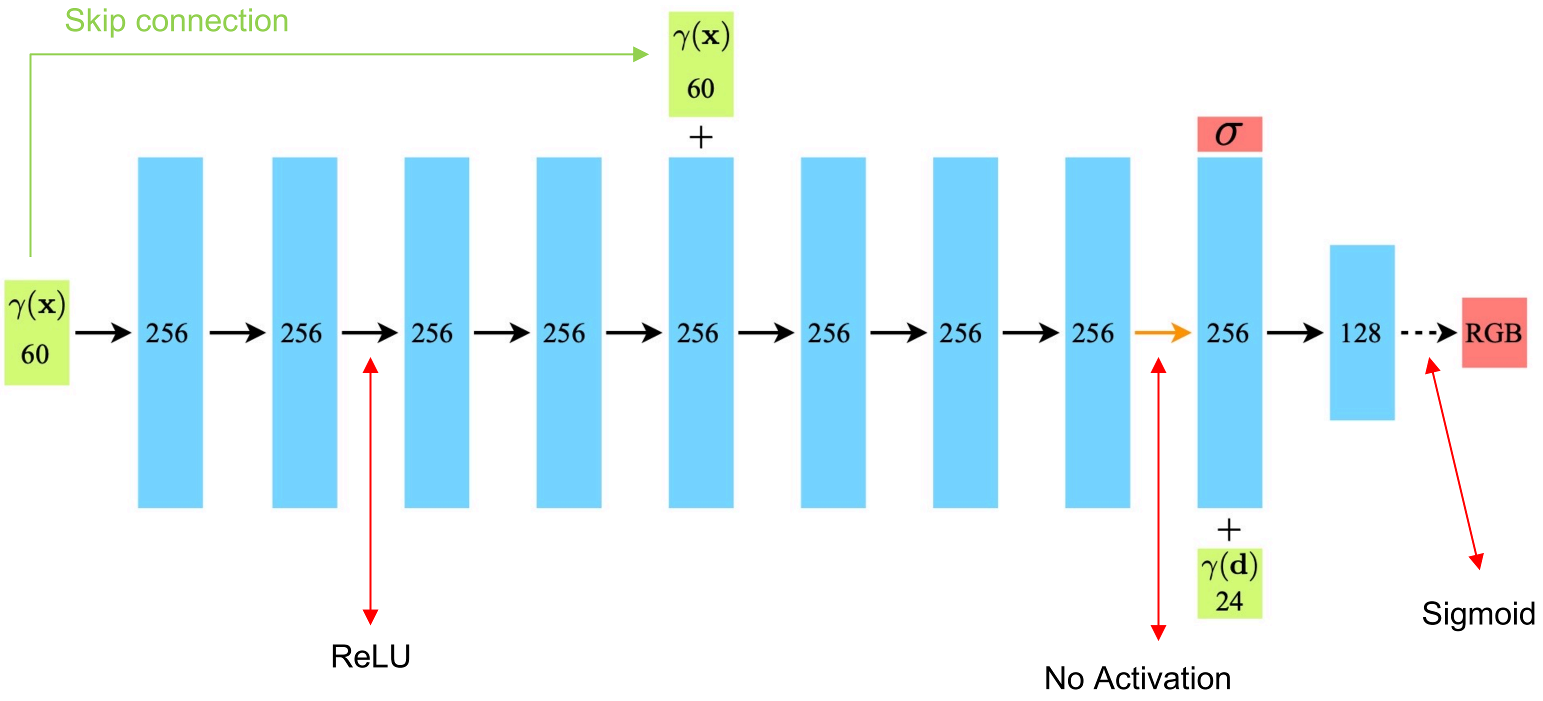
NeRF: Positional Encoding

- Map individual components of position and direction vectors to a higher dimensional space

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

- Similar concept as positional encoding in Transformer Architectures
- Empirically improves the preservation of high-frequency geometry and texture
- Surprising result, explored further in a follow-up work by the same authors
    - "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains" (Tancik, Srinivasan, Mildenhall, et al., 2020)

Network Architecture



Training Summary

- Sample a batch of camera rays from the dataset (bs=4096)
- Use hierarchical sampling to query coarse and fine points
- Use the volume rendering equation to calculate the color of the ray
- Compute the squared error between rendered and true pixel colors

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

- Optimize network parameters using Adam

Experimental Setup

- NeRF is compared against 3 state-of-the art techniques:

  Neural Volumes, Scene Representation Networks, Light Local Field Fusion

3 Datasets:

- “Diffuse Synthetic 360º” (DeepVoxels): 4 objects with simple geometry
- “Realistic Synthetic 360º”: 8 objects with complex geometry and reflections
- “Real Forward-Facing”: Phone camera images of 8 real-world scenes

Metrics:

- Peak Signal-to-Noise Ratio (PSNR), higher is better
- Structural Similarity Index Measure (SSIM), higher is better
- Learned Perceptual Image Patch Similarity (LPIPS), lower is better

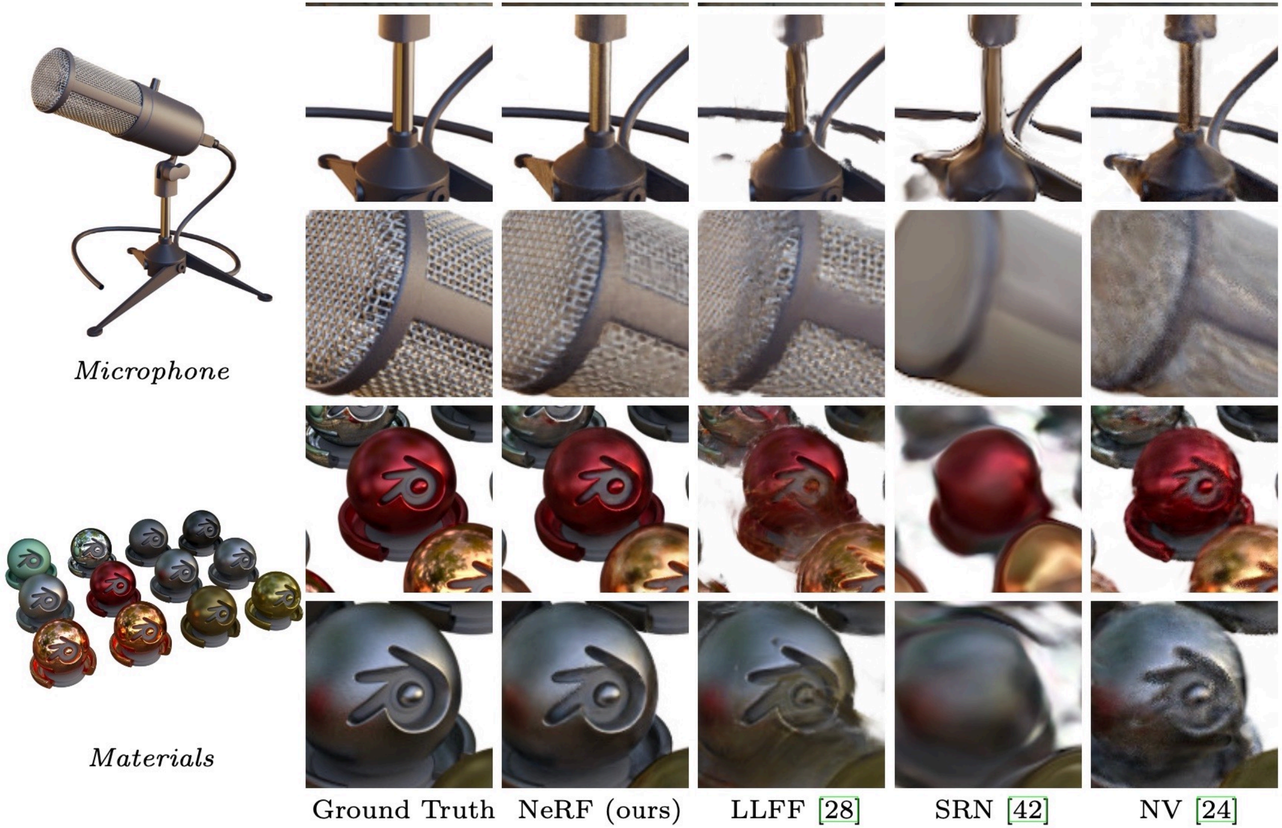
Experimental Results

- Outperforms existing works in nearly every tested metric

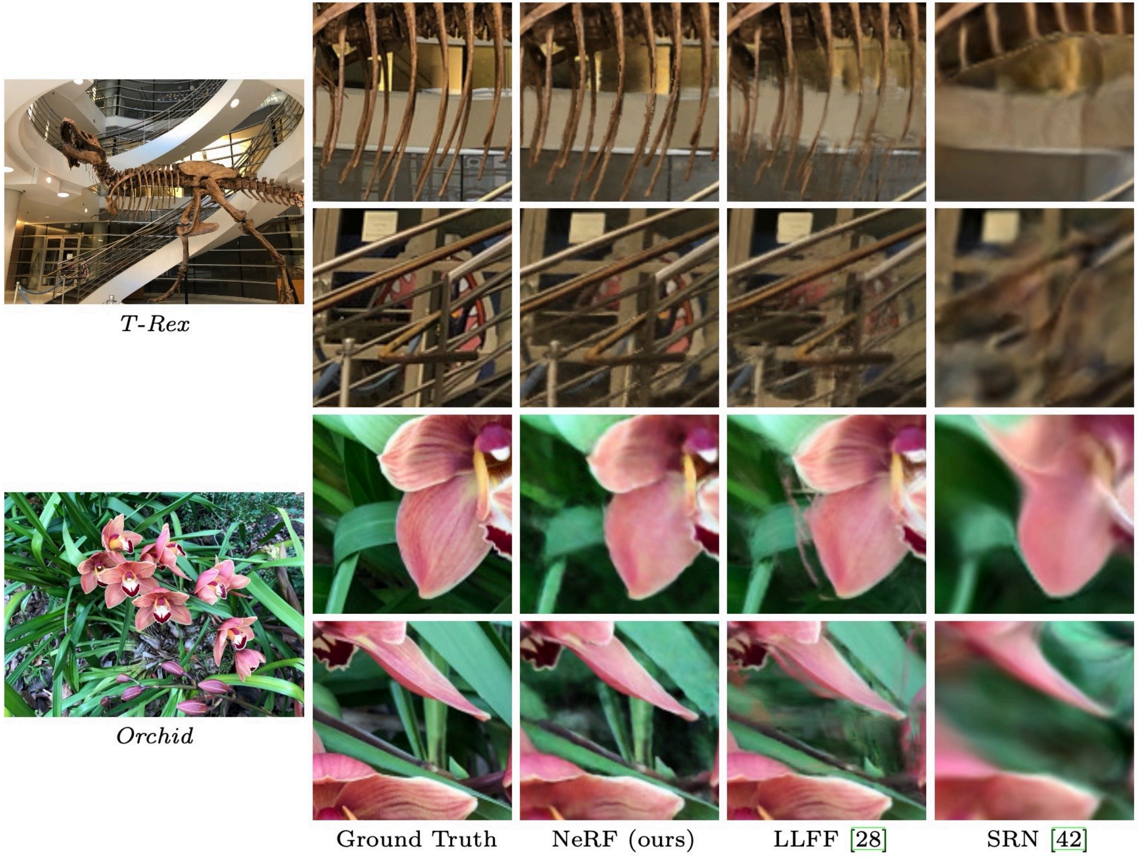
Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

- NeRF preserves fine details much better than other algorithms
- NeRF is able to render partially occluded regions
- The trained MLP has relatively low storage requirements: about 5MB

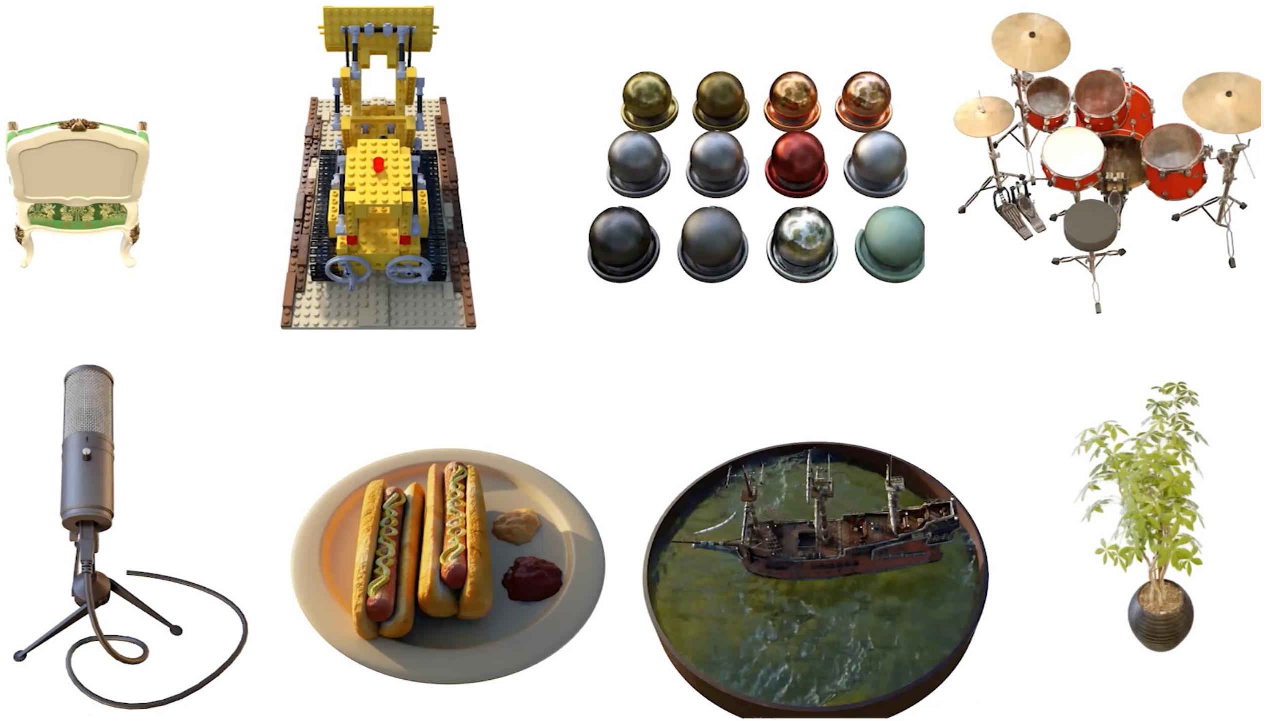
Qualitative Results: Realistic Synthetic Objects



Qualitative Results: Real-world scenes



Animated Results



Video: https://www.matthewtancik.com/nerf

Limitations

- NeRF only works with static scenes
- A trained NeRF model does not generalize to more than one scene
- Computationally expensive: 1-2 days to train each individual scene on a modern GPU
- Inference is slow: each pixel in a synthesized image requires volume rendering

Future Works and Extended Readings

- Survey Papers
  - “State of the Art on Neural Rendering.” Tewari et al., 2020
  - ”Neural Volume Rendering: NeRF And Beyond.” Dellaert et al., 2020
- Improving Speed
  - “DeRF: Decomposed Radiance Fields.” Rebain et al., 2020
  - “Plenoxels: Radiance Fields without Neural Networks.” Yu et al., 2021
- Improving Generalizability
  - “TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis.” Attal et al., 2021
  - “NeRF in the Dark.” Mildenhall et al., 2021
- Many more papers here: https://github.com/yenchenlin/awesome-NeRF

Summary

- Novel view synthesis generates images of scenes at previously unseen viewpoints.
- Novel view synthesis can improve robot planning and object recognition.
- Prior works are limited to simple shapes and do not scale well to high-resolution images
- NeRF encodes a static scene within the parameters of a feedforward neural network.
- The authors show very impressive qualitative results and show state-of-the-art performance with quantitative metrics and different scene types.