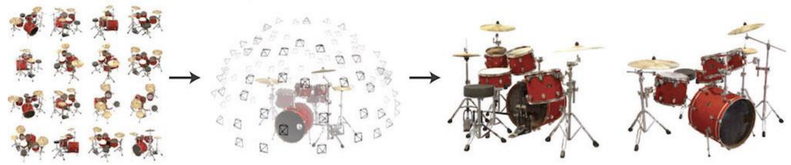
Neural Radiance Field (NeRF) for View Synthesis of arbitrary objects

Marco Aspromonte

Synapse AI symposium

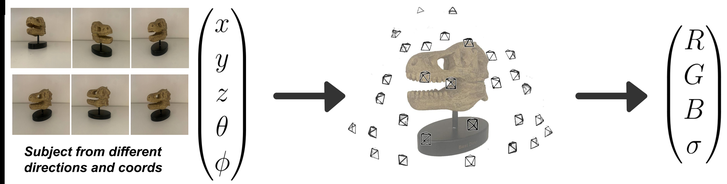
Introduction - Context and objectives

- **NeRF** answers to the necessity of performing fastly **view synthesis** and **360° visualization** of real scenes and arbitrary objects



- **Not trainable** on plain GPUs in less then 15 hours for a normal scene
- No information about **focal lenght** and **position** parameters in plain cameras
- Samples are already provided with such necessary parameters

Render a 3D representation of an arbitrary object starting from self-acquired 2D samples with a common device



Subject from different directions and coords

$$\begin{pmatrix} x \\ y \\ z \\ \theta \\ \phi \end{pmatrix} \rightarrow \begin{pmatrix} R \\ G \\ B \\ \sigma \end{pmatrix}$$

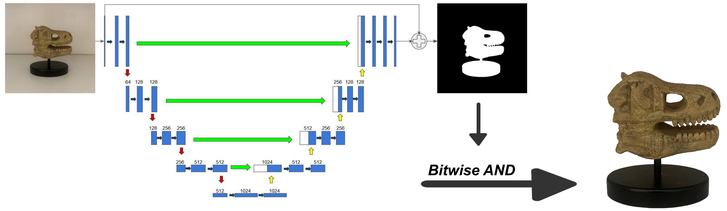
- **Estimate needed parameters** of acquired images with a fast model
- Acquire a well built dataset with **hand made instruments**

- NeRf is often compared to **Lombardi Neural Volumes**
- **Similarly expensive** but worst results
- About **few days** of training (3-5 days)
- Suffers of **lose of details** under **high frequencies** of movements

Comparison between Lombardi and NeRF

Methods and architecture

- Acquisition of an **arbitrary set of the same scene (55 samples)**
- Preprocessing phase to **remove the background** using U-net



- **Estimation of roto-translation parameters** for camera poses with **Nerfmm** library [1]:
  - Initialized as identity matrix
  - **Rotation** is optimized performing axis-angle representation :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0.9942903 & 0.07350748 & 0.07735278 & 0.17515 \\ -0.07965104 & 0.9936406 & 0.07958642 & 0.17786399 \\ -0.07101067 & -0.08529324 & 0.99382216 & -0.40722343 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Initialized matrix — *Computed roto-translation parameters for the first input sample*

- **Focal lenghts** estimation, $f_x = W$ and $f_y = H$
  - Initialized with the **width** and **height** values of the samples
  - All the samples are resized in (1024*1024) to have a valid standard shape:

$$\underset{\textit{Inizialized focal lenght}}{[1024 \quad 1024]} \rightarrow \underset{\textit{Computed focal lenght}}{[406.3 \quad 694.9]}$$

- **Feed the input** to the Tiny-NeRF neural network [2]:
  - Fully connected sequential network **with 8 total layers** (6 linear + 2)
  - **6 linear layers** make training phase lighter, in NeRF there are **8 ones**
  - **RGB and density** fully connected layers at the end

Evaluation

- Possibility to train the model with a single GPU on Colab Pro, assigned randomly between ***K80, P100, T4***
- **< 30 minutes** for all the *55 samples* for *1000 epochs*

100 epochs
PSNR = 16.277
- High noise
- Very blurry

1000 epochs
PSNR = 21.326
- Less noise
- Smoother shape

10k epochs
PSNR = 26.041
- More robust at high frequencies
- Nice features

- **Good rendering** when training for **10k epochs**
- **More or less 3 hours for 10k epochs**

Conclusions

Something about Tiny-NeRF limitations:

- **assume the world as geometrically and materially static**
- **the density and radiance of the world have to be constant**

However, real world **constantly changes** on the light conditions

- Integrate a **U-net for semantic segmentation** helps the model to estimate the uknown parameters focusing on the object and **maximizes PSNR**
- Reduce the Tiny-NeRF model to **6 layers** helps to minimize the training time
- Minimize the error induced by Tiny-NeRF limitations by acquiring samples under **constant light reflection** and following **sensible viewing directions**

[1][Estimation of roto-translation] : https://github.com/ActiveVisionLab/
[2][Tiny-NeRF model] : https://github.com/bmild/nerf/