Regarding NeRFs:

- Multi Layer Perceptron (MLP) introduced by NeRF is very slow in training and in rendering
  - specifically: the training time required order of hours

Prior to NeRF:

Voxel quantized volumetric grid

Finding of ReLU:

- Optimizing voxel volumes in a stage-wise manner
  → training the voxel grid from coarse to a fine resolution can actually reconstruct the scene much faster compared to NeRF

Time: order of minutes

However prior work is unable match the quality of NeRFs

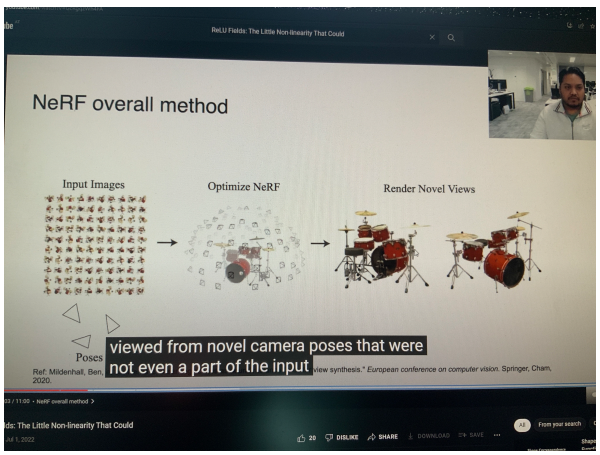
=> ReLU

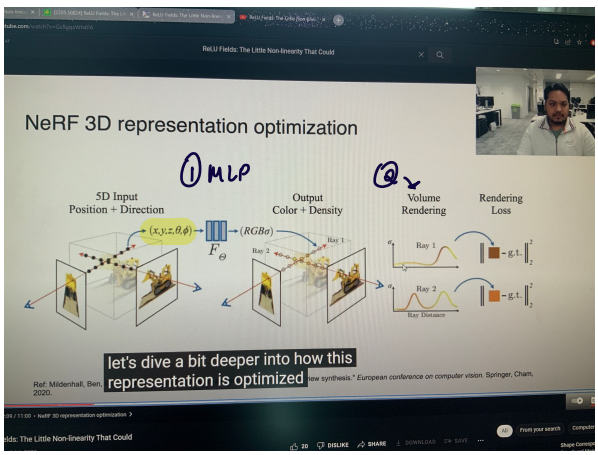
A simple modification to voxel grids that can be trained and evaluated quickly, while also retaining much of the high quality reconstructions that nerf introduced.

Specifically: ReLU fields also takes training time in the order of minutes, similar to voxel grids

Background:

NeRF:



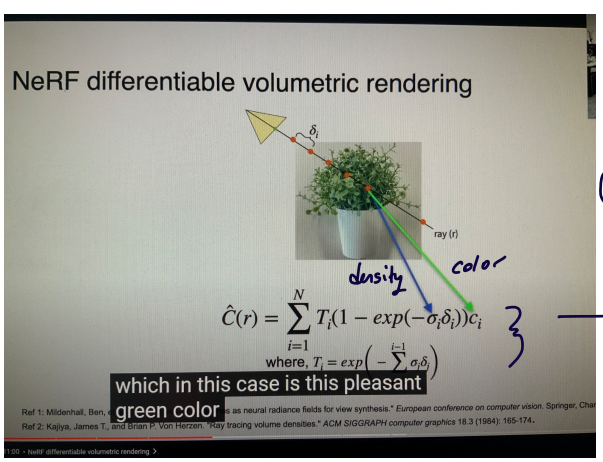


① MLP to represent the 3D scene volumetrically

② Rendering the scene using a physically based volume tracer

③ MLP is trained simply by optimizing mean squared error (MSE) between the rendered pixel values and the ground thruth pixel values

Deep dive: NeRF volumetric rendering



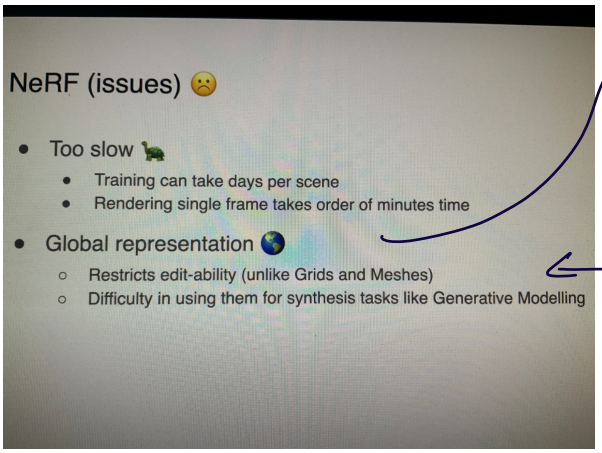
① Rays are marched into the volume to prove the representation for instantaneous sample values along the ray

② Sample values are essentially the instantaneous density and the directional color

→ discretized version of the eq. w/ which the volumetric scene is rendered

③ The discrete deltas δ_i between the adjacent samples is used for compositing the final value for pixel

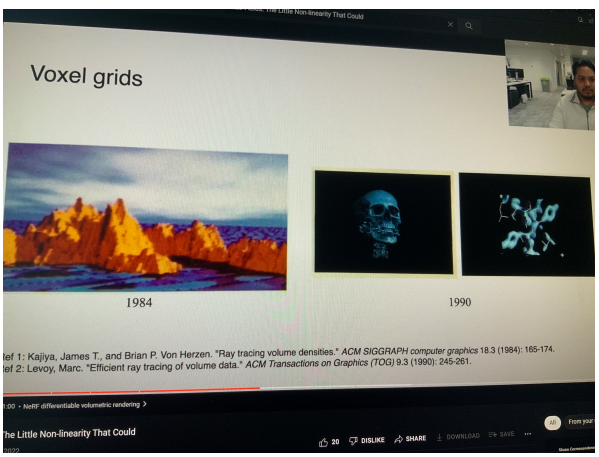
NeRF drawbacks:



→ NeRF's are basically intricate boxed of enigmatic connections

⇒ any edits made to the weights to the MLP locally, may have global effects on the content of the scene ←

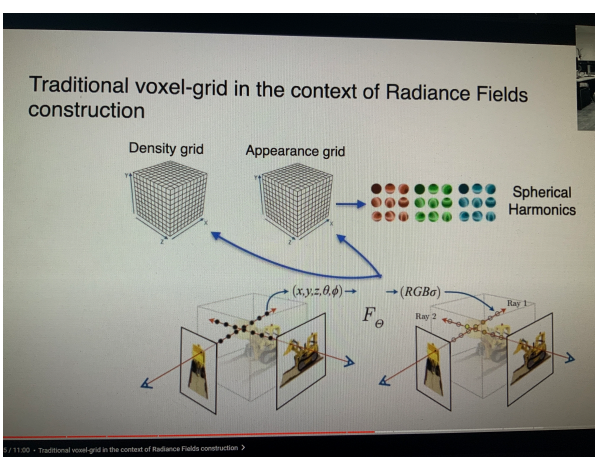
Voxel grids:



Volumetric rendering stores scene info in a regular 3D voxel grid

⊕ simple ⊕ fast

⊖ memory storage !!



Instead of MLP the 3D scene is being represented by two dense voxel grids

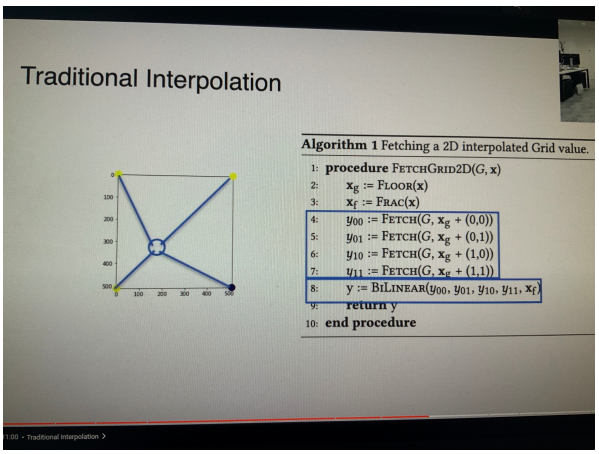
① Volumetric density

② Appearance
- can be modelled by Spherical Harmonics

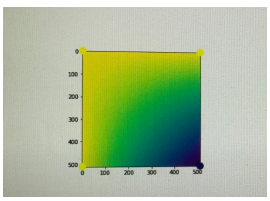
However:

!! ⊖ fuzzy geometry ⊖ blurry appearance

the paper found the Colurryness is somehow related to the grids interpolation:

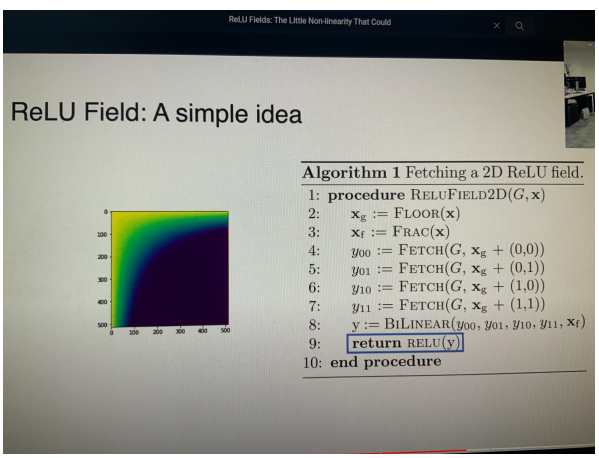


① cell that the point is located
② Data vals at corners are looked up
③ Data value at the point of interest is computed as a weighted avg. of the corners
→ weighted avg: is often linear (bilinear, tri...)



Repeat the procedure for all the points continuosly inside the cell, it gives rise to a smootle pattern
⇒ In the density grids case is the main cause of the Colurryness

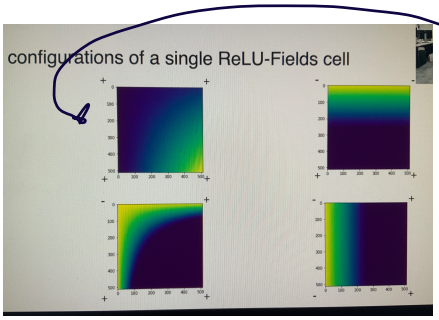
Proposed Method:



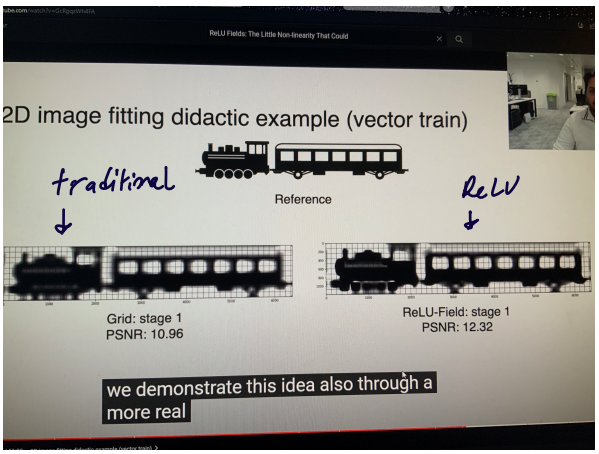
Simply insert ReLU non linearity right after the BILENEAR interpolation

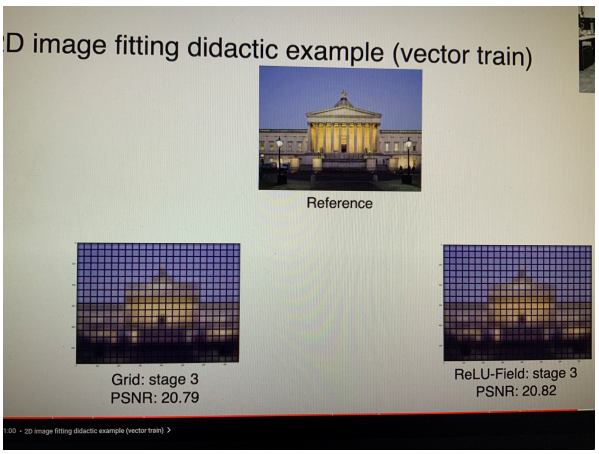
Note:
- we allow the density to take neg. vals.
→ in this case, do not have a physical interpretation on the raw grid

Rest of the rendering process is agnostic to the change due to the non-linearity the ReLU function

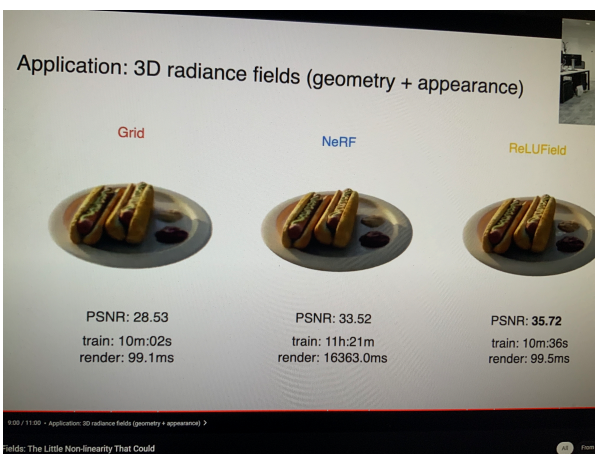


If all the vals on the corners are positive, then the behavior reverse to BILINEAR interpolation
For other combinations, various types of hard creases or surfaces can be modeled





⊖ not observed when voxels natural images
→ mostly contains smooth transitions
→ restrict the use of ReLU-Fields to model the density and not the spherical harmonics coeff.



Limitations:

(1) Expensive memory requirements: $O(n^3)$

(2) cannot model more than one "crease" per voxel