NERF – GENERATING 3D WORLD FROM A BUNCH OF IMAGES

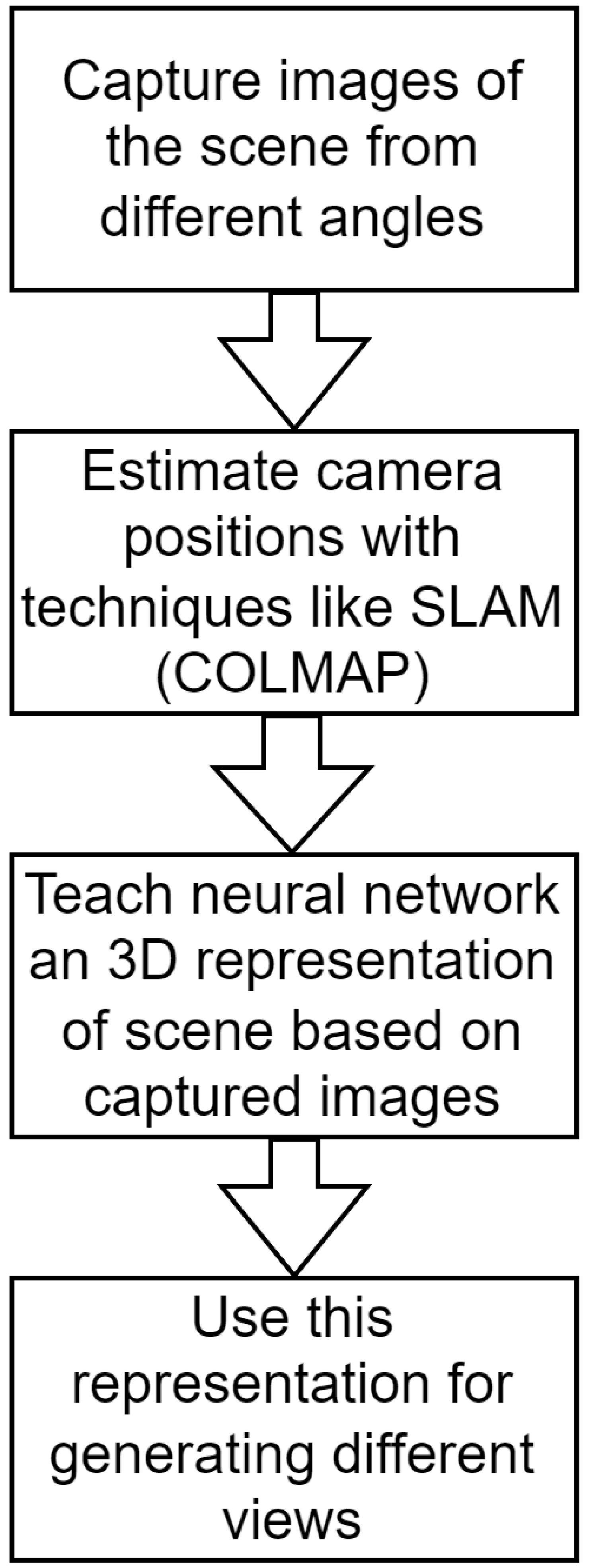
ŁUKASZ PIERŚCIENIEWSKI, SENIOR SOFTWARE ENGINEER - AI

NERF: NEURAL RADIANCE FIELDS

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (nvlabs.github.io)

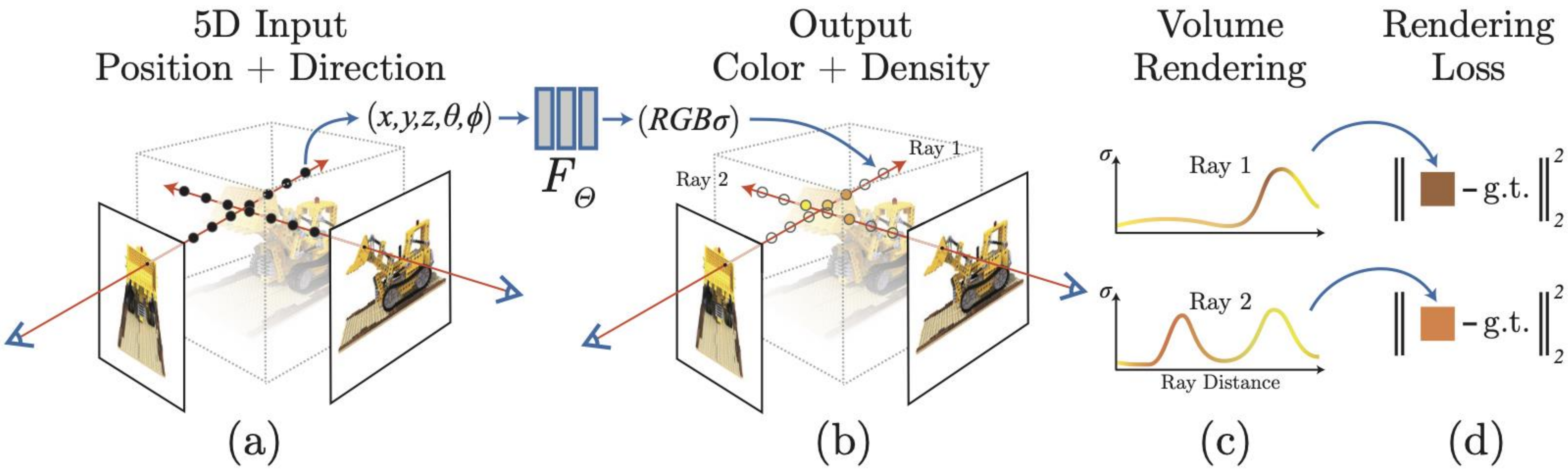
NERF: NEURAL RADIANCE FIELDS

How do they work?



NERF: NEURAL RADIANCE FIELDS

How do they work?



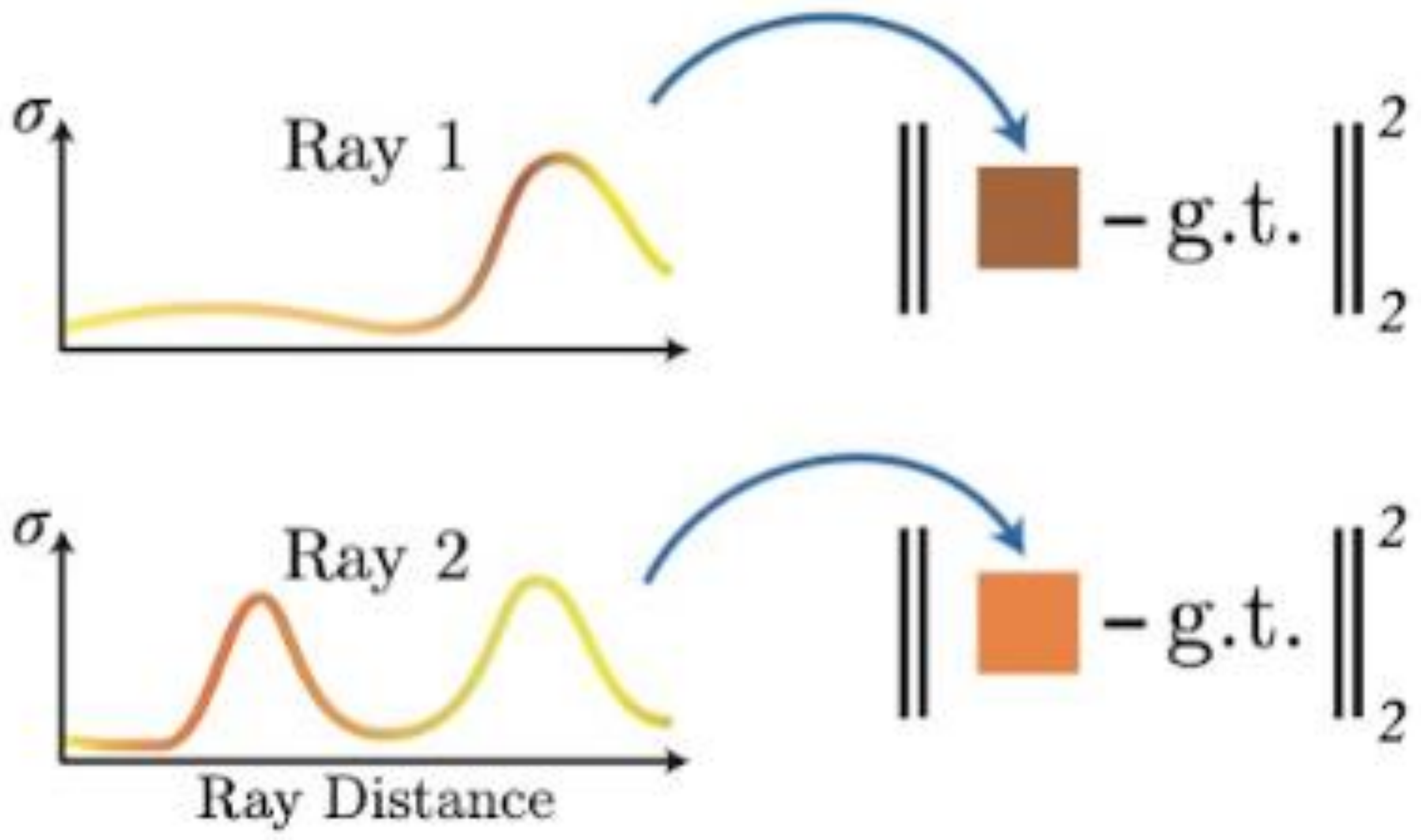
[NeRF: Neural Radiance Fields (matthewtancik.com)](matthewtancik.com)

NERF: NEURAL RADIANCE FIELDS

How to extract color of the pixel?

- The model predicts occupancy for a given point
- The model predicts color based on the given point and view direction
- All samples along the ray are taken into account when calculating final pixel color
  - Color is a weighted sum of colors of samples on ray, according to the equation below

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$



$\mathbf{r}$ - ray

t_n, t_f - near and far bounds

$\mathbf{d}$ - camera position

$\mathbf{c}$ - predicted color

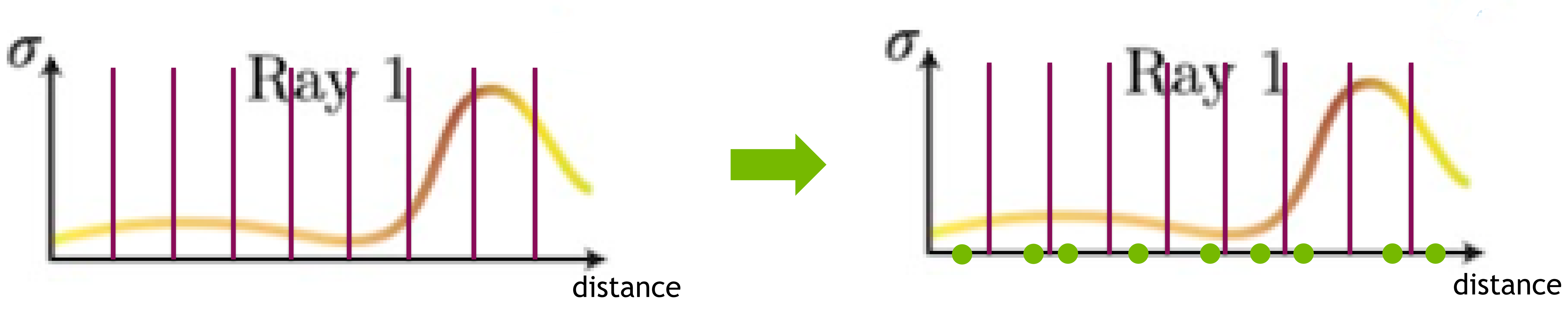
σ - predicted occupancy value

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

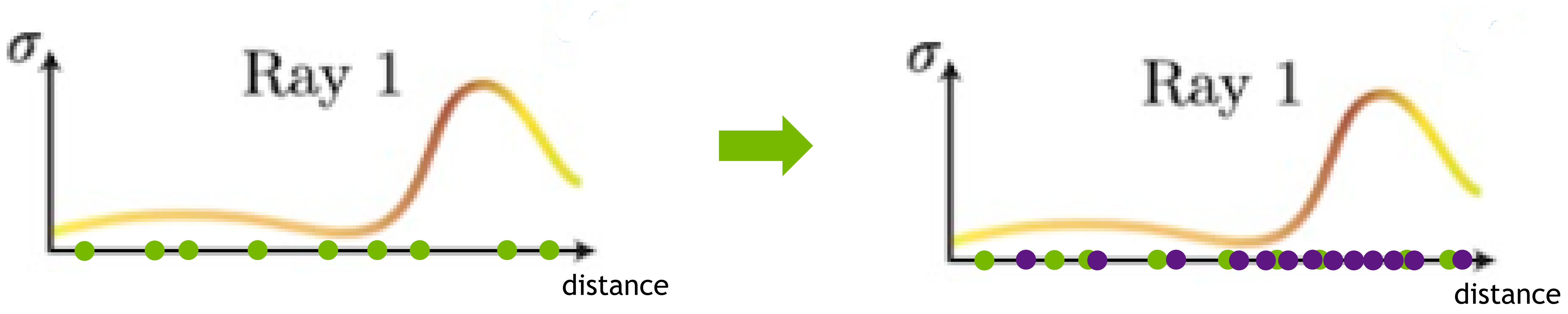
NERF: NEURAL RADIANCE FIELDS

How to extract color of the pixel?

- Coarse network



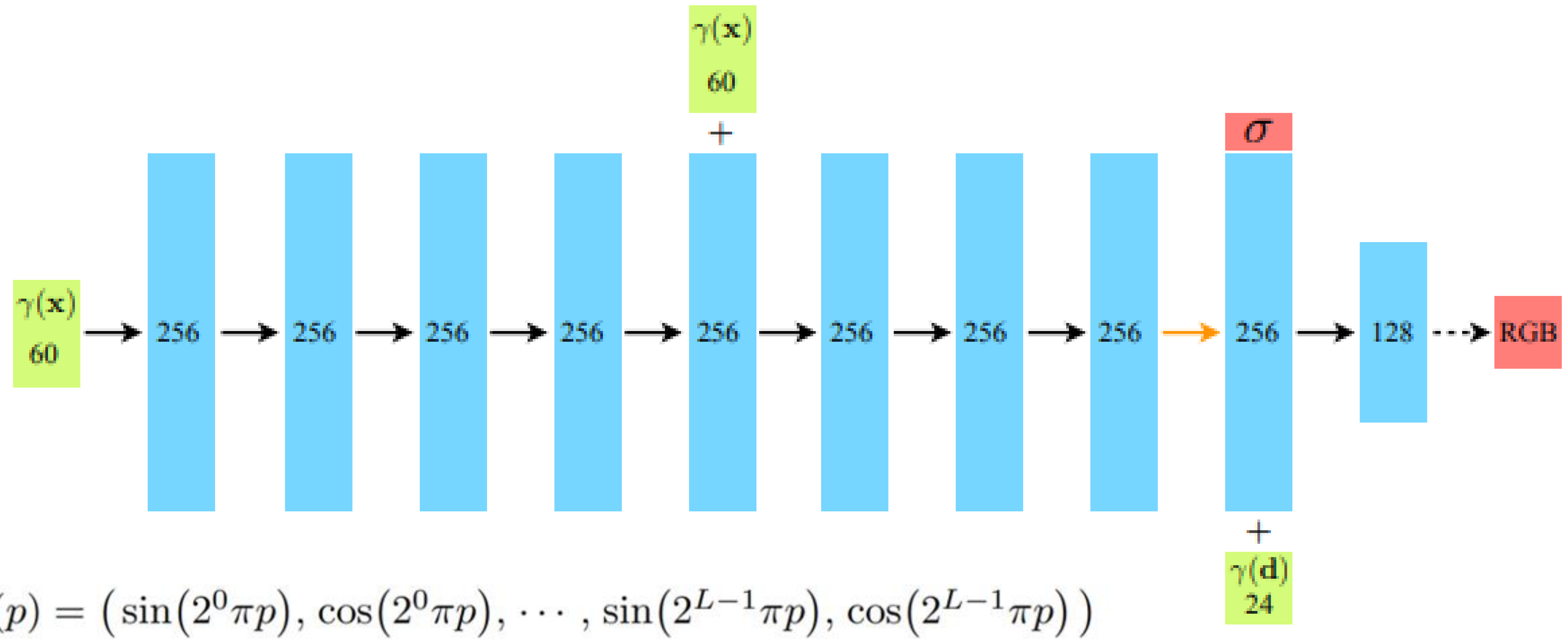
- Fine network
  - Calculate CDF from coarse network
  - Sample uniformly from CDF and invert it



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

NERF: NEURAL RADIANCE FIELDS

Neural network architecture



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

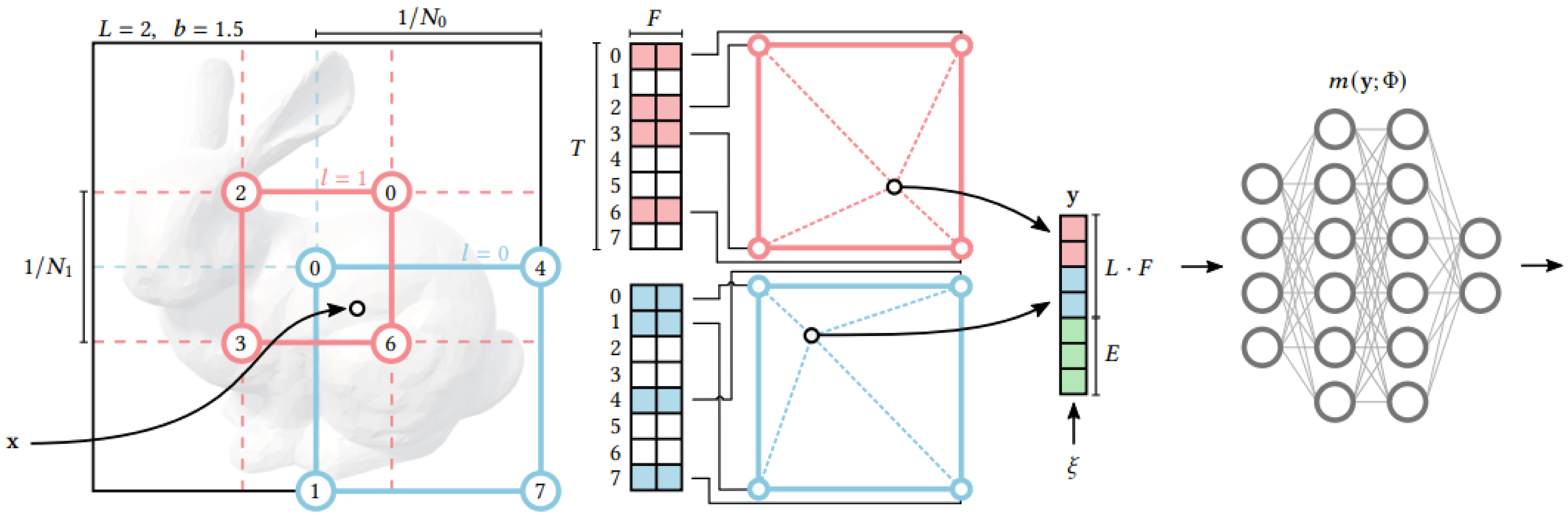
Where p is either (the same encoding for both input position and direction):

x – points coordinates in 3d space
d – direction vector in 3d space

MLP + ReLU layer

IMPROVEMENTS

Positional encoding – Multidimensional hash encoding

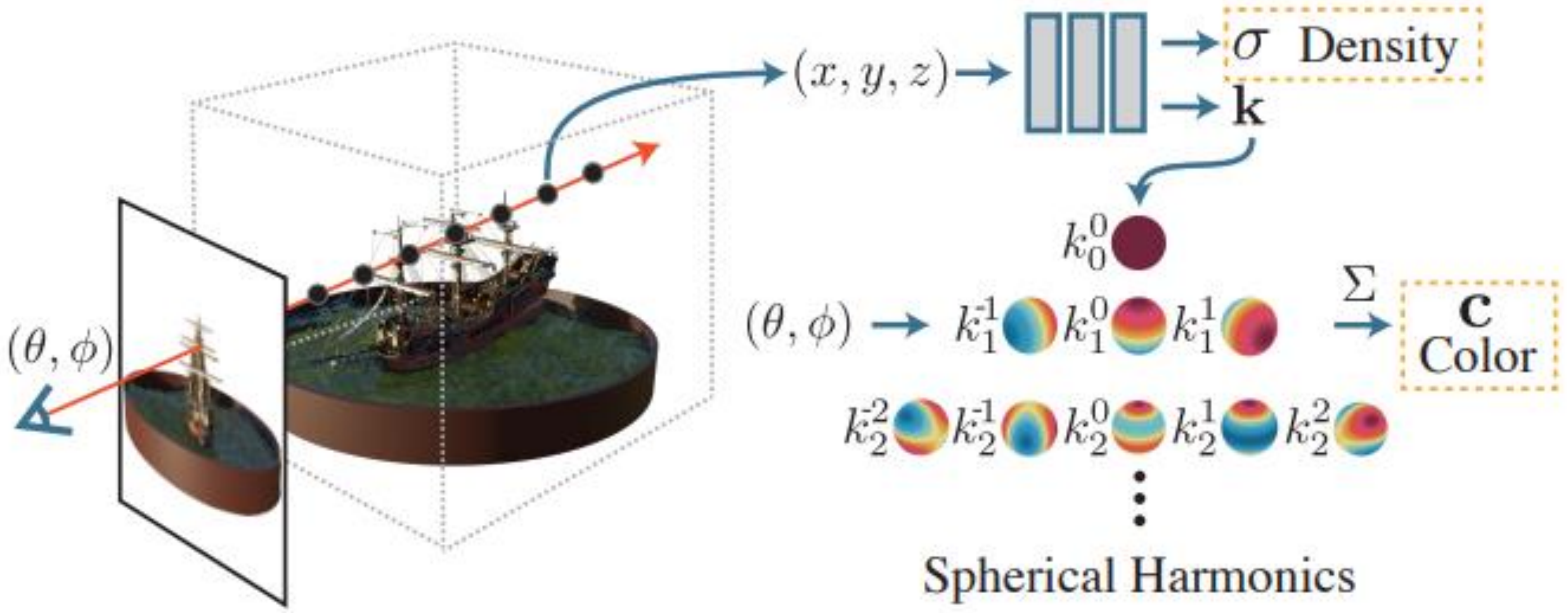


(1) Hashing of voxel vertices (2) Lookup (3) Linear interpolation (4) Concatenation (5) Neural network

[Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (nvlabs.github.io)]

IMPROVEMENTS

Positional encoding – Spherical harmonics



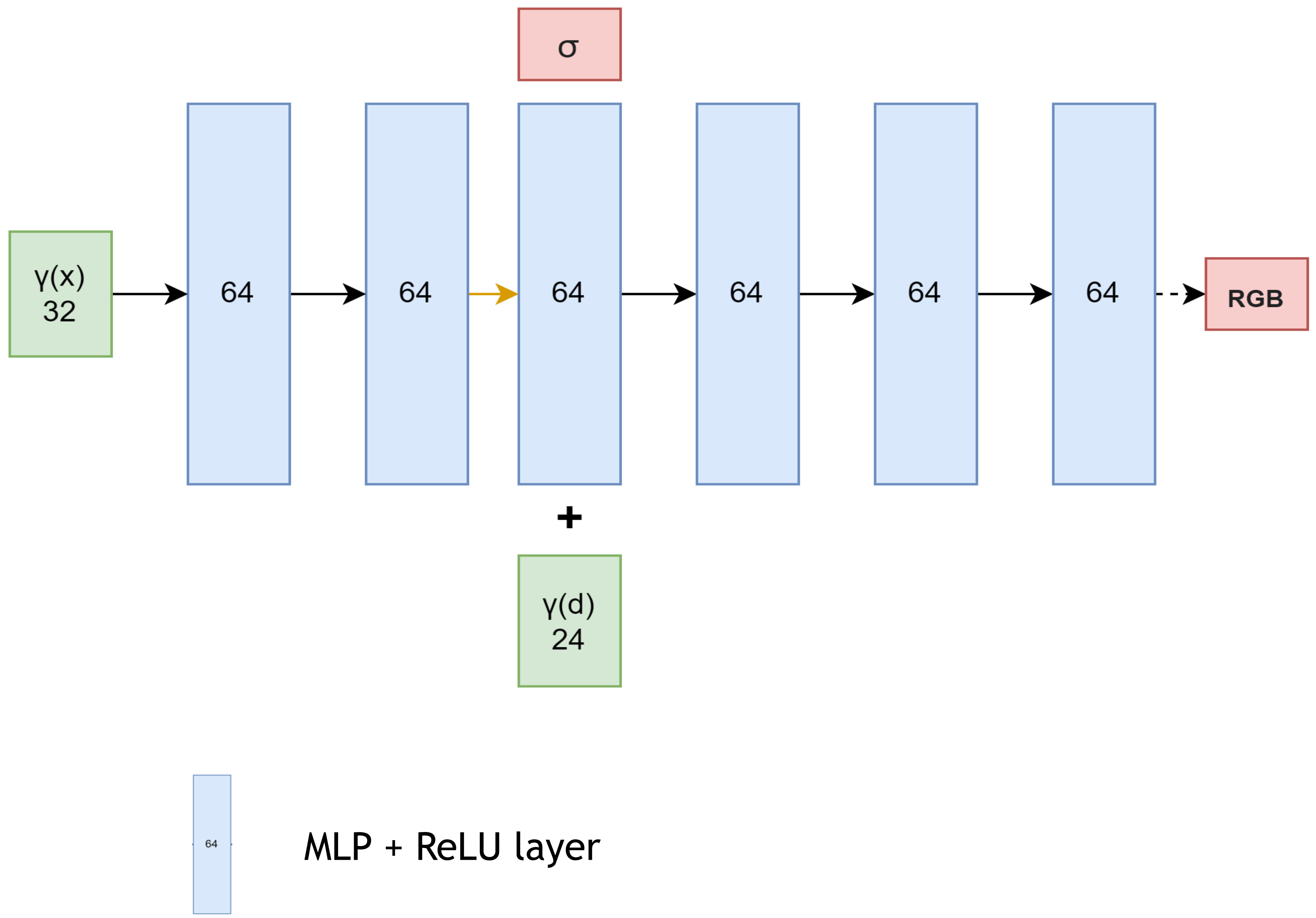
$$\mathcal{L}_{TV} = \frac{1}{|\mathcal{V}|} \sum_{\substack{\mathbf{v} \in \mathcal{V} \\ d \in [D]}} \sqrt{\Delta_x^2(\mathbf{v}, d) + \Delta_y^2(\mathbf{v}, d) + \Delta_z^2(\mathbf{v}, d)}$$

Δ^2 – MSE between voxel and next voxel
V – voxel considered
d – value in the encoding

[PlenOctrees for Real-time Rendering of Neural Radiance Fields (alexyu.net)]

HASH-NERF

Better encoding = Smaller network (~8 times)

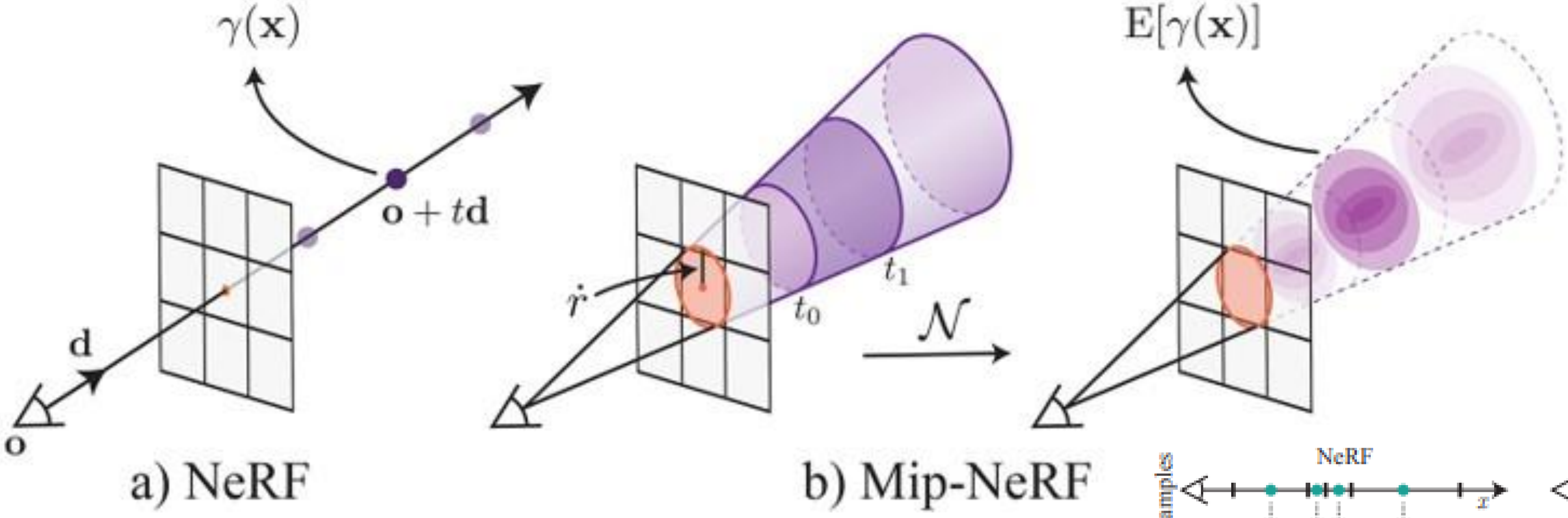


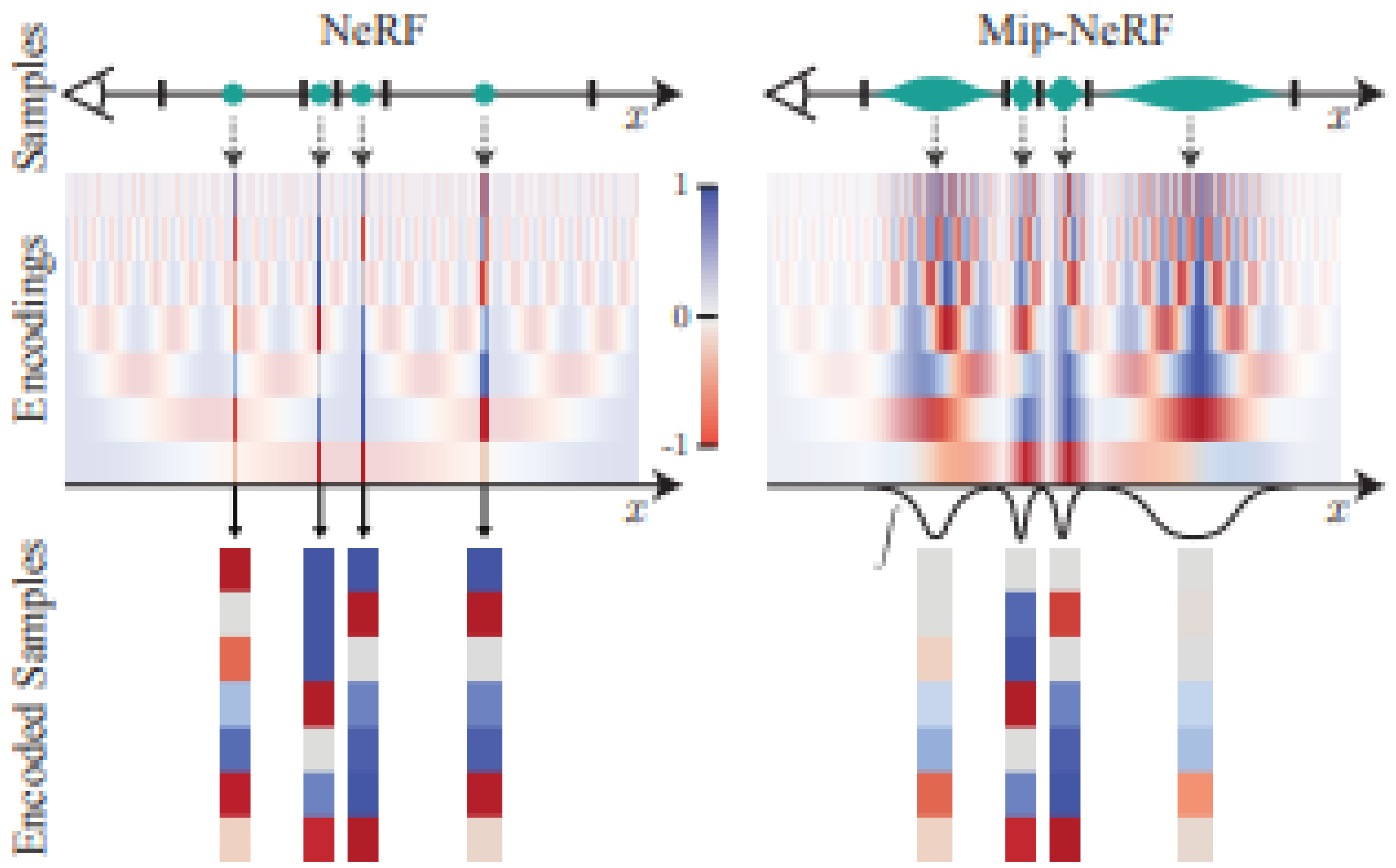
- Smaller occupancy network
  - Cheaper to check if the point should be evaluated further
  - Allows for skiping empty space
  - No more coars and fine distinction!
- Bigger color network

- Much faster convergence in terms of iterations needed (25k vs 500k)
- Much faster convergence in terms of iterations per second (100k rays/s vs 10k rays/s)
- Less compute needed (few minutes vs days)
- Larger batch size allowed

MIP-NERF: A MULTISCALE REPRESENTATION FOR ANTI-ALIASING NEURAL RADIANCE FIELDS

Because pixel information is inperfect





Better quality overall several datasets

- 31.1 PSNR for NeRF (~days)
- 33.5 PSNR for MIP-NeRF (~hours)
- 33.1 PSNR for HashNeRF (~minutes)

NERF: NEURAL RADIANCE FIELDS

Results with depth map

Depth map can be obtined with sending a ray and seeing when it hits something (occupancy above threshold).

NERF: GENERATING 3D WORLD

Images are okay, but what about 3d representation

- Can render any camera positions, moving in 3d world
- We can generate pointclouds base on depth map
  - Converting pointclouds to mesh might be needed if desired
- Filtering from background/noise is possible, but has to be done
- We have to choose between:
  - Speed/resources with HashNeRF approach
  - Quality with MIP-NeRF approach
- You would be able to train it by your own!
  NeRF implementation will be available on GitHub:
  - https://github.com/NVIDIA/DeepLearningExamples

QUESTIONS?

REFERENCES

Representing Scenes as Neural Radiance Fields for View Synthesis (2020)

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng

https://www.matthewtancik.com/nerf
https://arxiv.org/abs/2003.08934

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains (2020)

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, Ren Ng

https://arxiv.org/abs/2006.10739

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (2022)

Thomas Müller, Alex Evans, Christoph Schied, Alexander Keller

https://nvlabs.github.io/instant-ngp/

Inverting Neural Radiance Fields for Pose Estimation (2021)

Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, Tsung-Yi Lin

https://yenchenlin.me/inerf/
https://arxiv.org/abs/2012.05877

Bundle-Adjusting Neural Radiance Fields (2021)

Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, Simon Lucey

https://arxiv.org/abs/2104.06405

Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis (2021)

Ajay Jain, Matthew Tancik, Pieter Abbeel

https://arxiv.org/abs/2104.00677