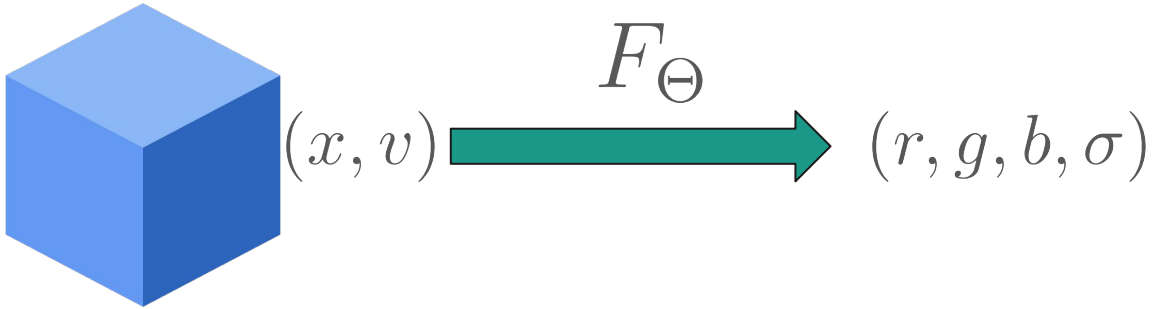
Different types of entry encodings for NeRFs networks

Outil maths 3 février 2025,
Gaspard Thévenon

Overview of NeRFs

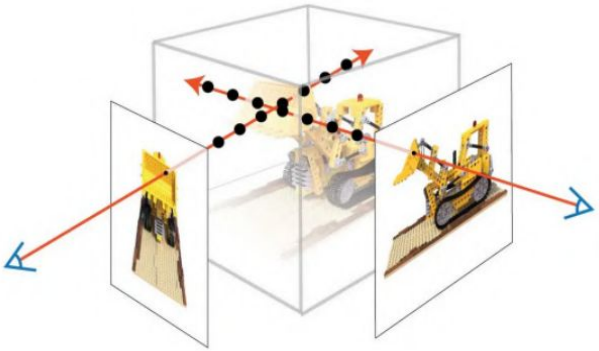
Mildenhall et al, 2020

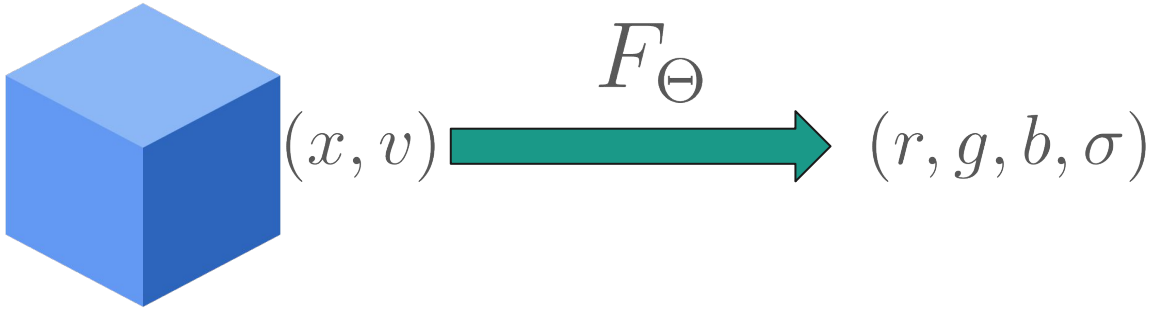
Overview of NeRFs



Mildenhall et al, 2020

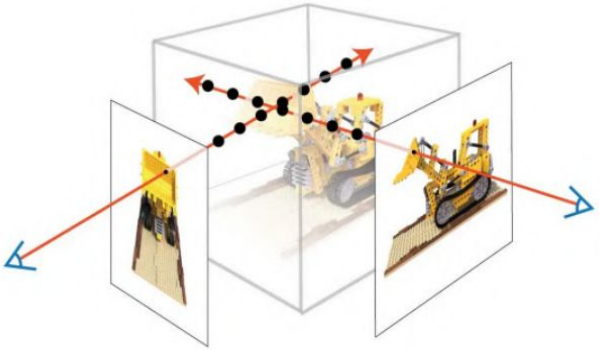
Overview of NeRFs

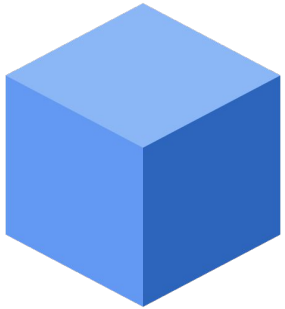




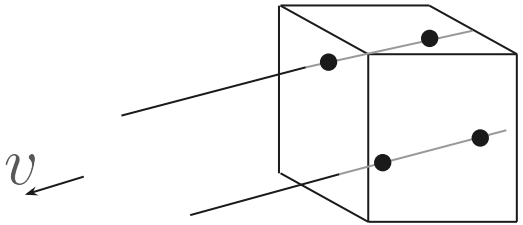
Mildenhall et al, 2020

Overview of NeRFs





$$(x, v) \xrightarrow{F_\Theta} (r, g, b, \sigma)$$



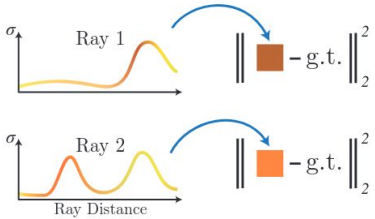
$$C = \sum_i T_i(1 - \exp(-\sigma_i \delta_i))c_i = \sum_i w_i c_i$$

Mildenhall et al, 2020

NeRFs examples

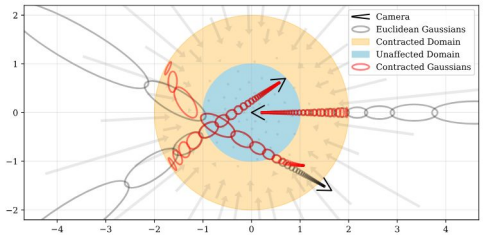
Lots of interesting questions around NeRFs

- Losses



- Smarter sampling strategies along each ray

- Handle unbounded scenes



The need for an encoding

MLP

8 layers, ~500k parameters

(x, y, z) →  → (r, g, b, σ)

F_Θ

ground truth

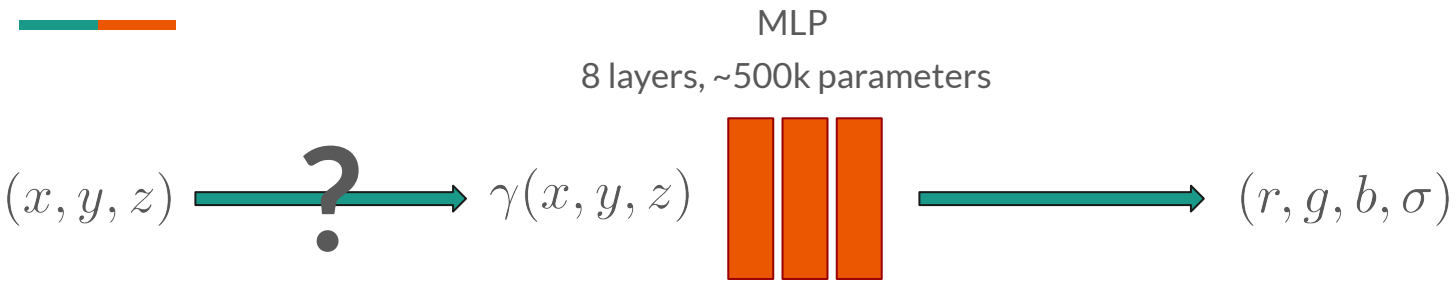
Image-based loss

Retro-propagated to train the MLP parameters

vanilla MLP

Mildenhall et al, 2020, Müller et al, 2022

The need for an encoding



$$F_\Theta$$

Frequency encodings

Frequency encodings

Frequency encodings

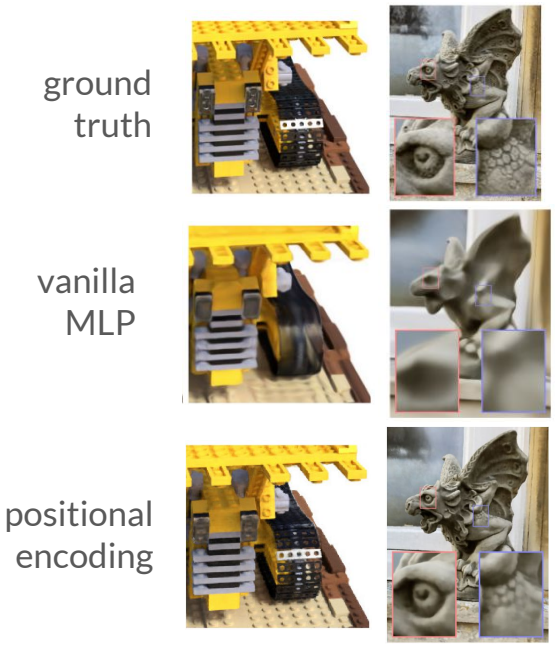
In the original NeRF, the authors use **Positional Encoding**:

$$\gamma(x) = (\sin(2^0 x), \ldots, \sin(2^{L-1} x), \cos(2^0 x), \ldots, \cos(2^{L-1} x))$$

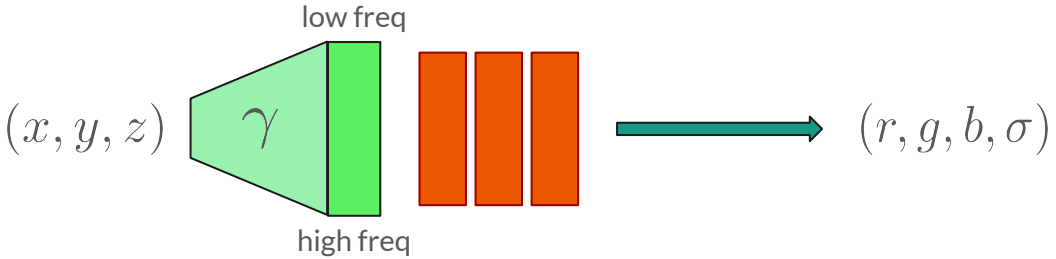
Mildenhall et al, 2020

Frequency encodings

In the original NeRF, the authors use **Positional Encoding**:



$$\gamma(x) = (\sin(2^0 x), \ldots, \sin(2^{L-1} x), \cos(2^0 x), \ldots, \cos(2^{L-1} x))$$

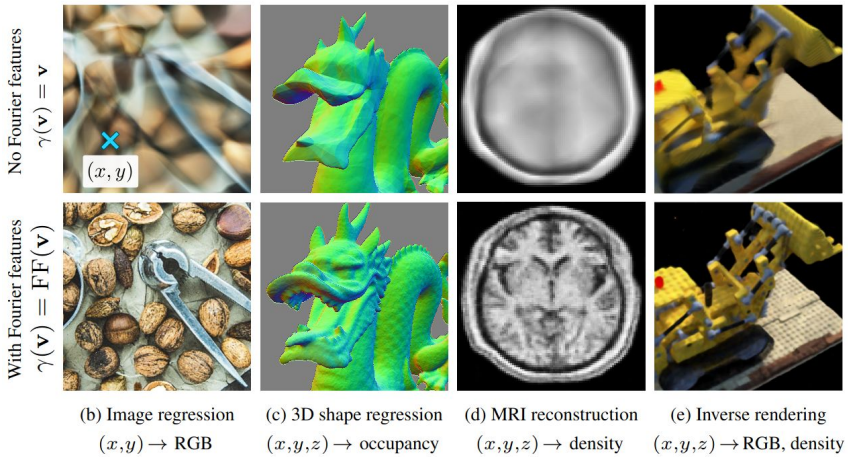


Mildenhall et al, 2020

Frequency encodings

This idea has been extended in various ways in NeRF applications. For example, extended to generic parallel wavefronts.

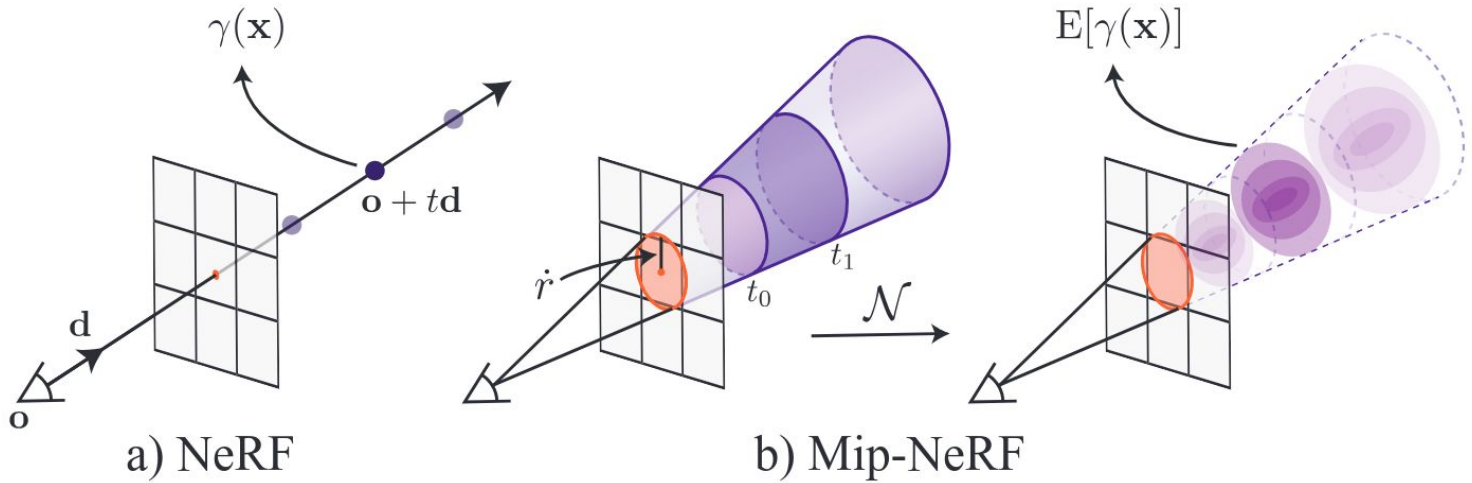
$$\gamma(\mathbf{v}) = \left[a_1 \cos(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), a_1 \sin(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), \ldots, a_m \cos(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v}), a_m \sin(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v})\right]$$



Tancik et al, 2020

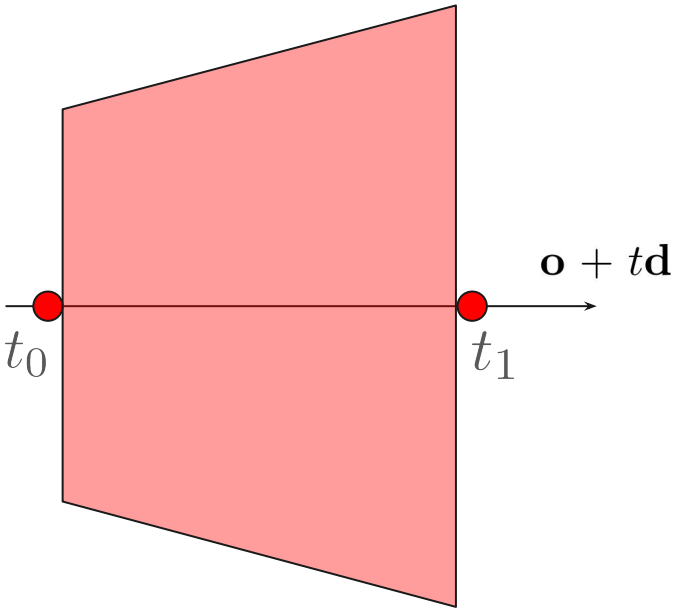
Frequency encodings

Also extended to multiple levels of details, by mip-NeRFs, which implements Integrated Positional Encoding.



Barron et al, 2021a

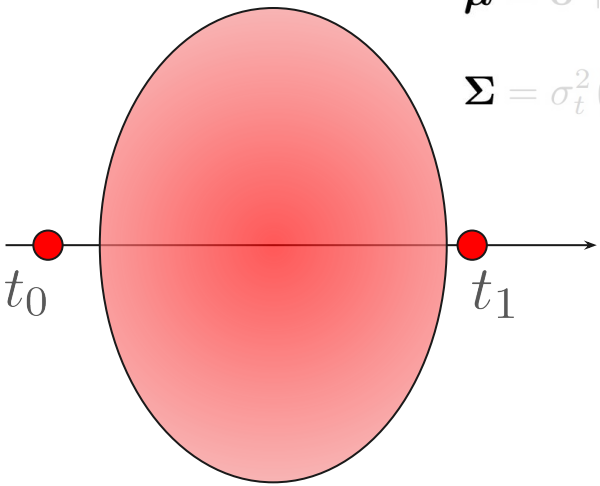
Frequency encodings



$$\gamma(t_0, t_1) = \mathbf{E}_{x \in \blacksquare}[\gamma(x)]$$

Barron et al, 2021a

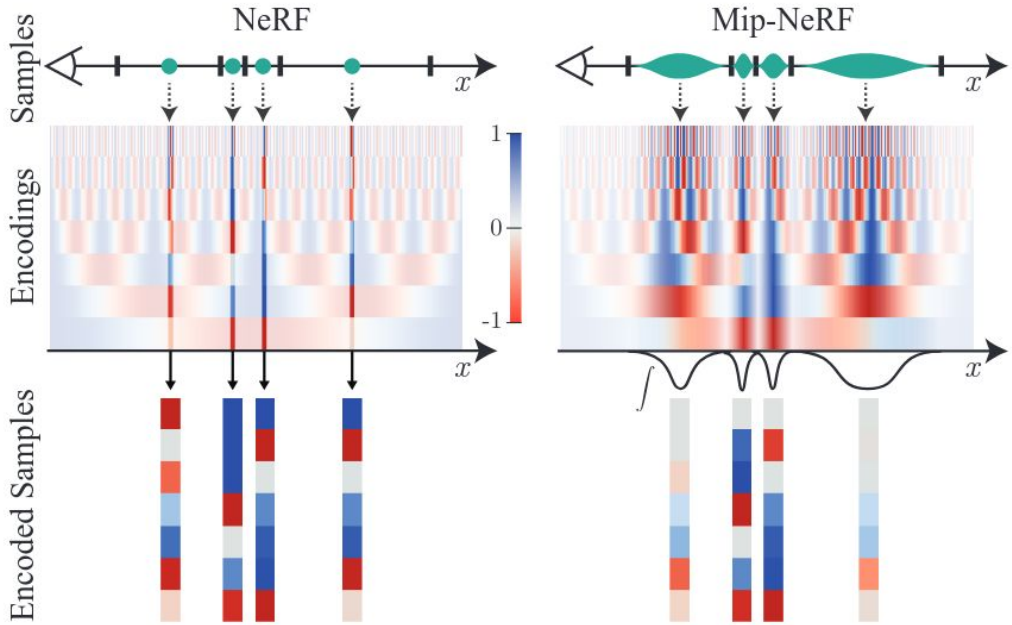
Frequency encodings



$$\gamma(t_0, t_1) = \mathbf{E}_{x \sim \mathcal{N}(\mu, \boldsymbol{\Sigma})}[\gamma(x)]$$

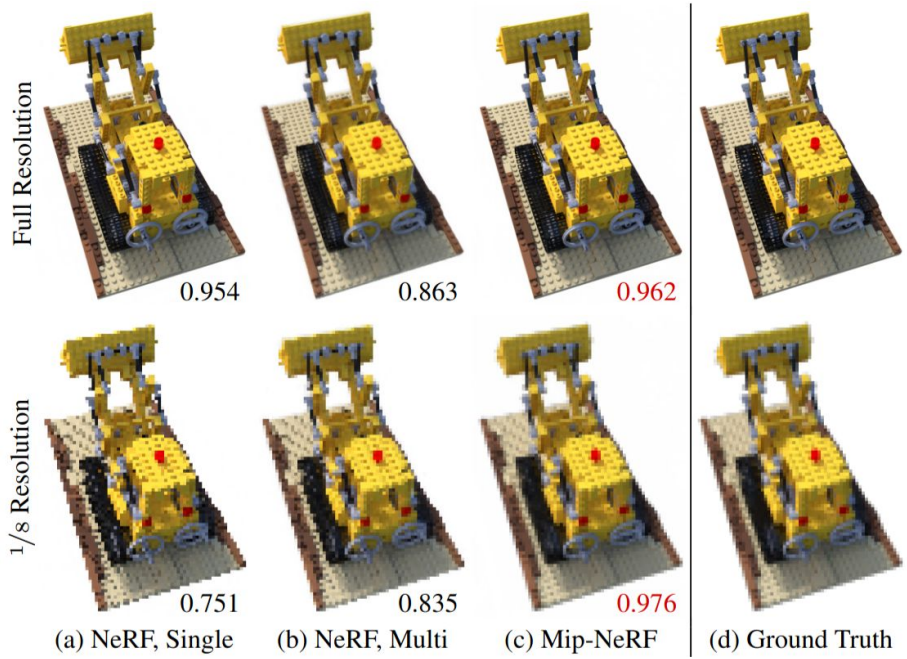
Barron et al, 2021a

Frequency encodings - mip-NeRF



Barron et al, 2021a

Frequency encodings - mip-NeRF



Barron et al, 2021a

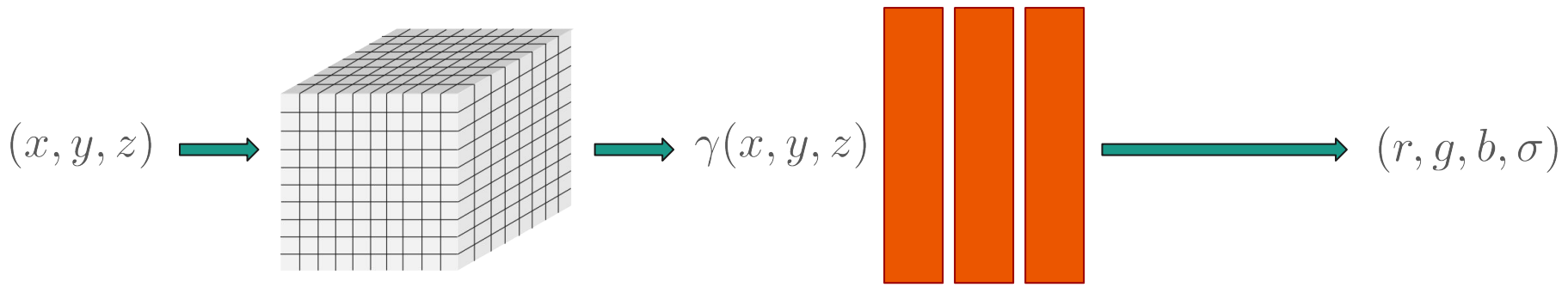
mip-NeRF videos

Parametric encodings

Parametric Encodings

Parametric Encodings - Single level grid

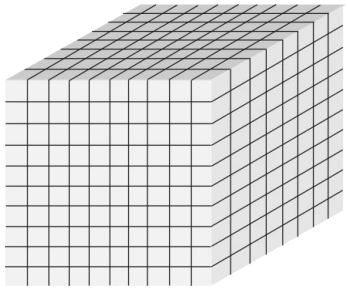
Dense feature grid (128^3) of 16-dim vectors



trilinear interpolation at sampled location

Parametric Encodings - Single level grid

Dense feature grid (128^3) of 16-dim vectors



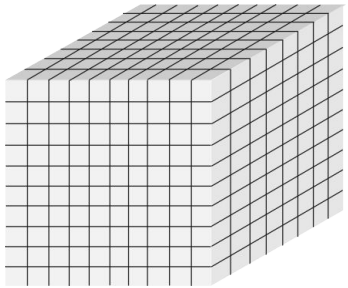
trilinear interpolation at sampled location

(x, y, z) → $\gamma(x, y, z)$ → (r, g, b, σ)

More parameters to learn…

Parametric Encodings - Single level grid

Dense feature grid (128^3) of 16-dim vectors



trilinear interpolation at sampled location

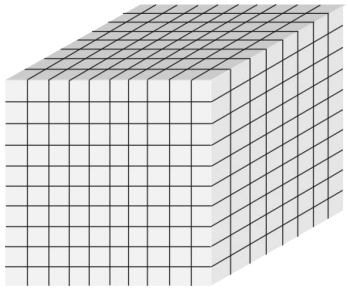
(x, y, z) → $\gamma(x, y, z)$ → (r, g, b, σ)

More parameters to learn…

But actually, only 8 more per sample, so little overhead !

Parametric Encodings - Single level grid

Dense feature grid (128^3) of 16-dim vectors

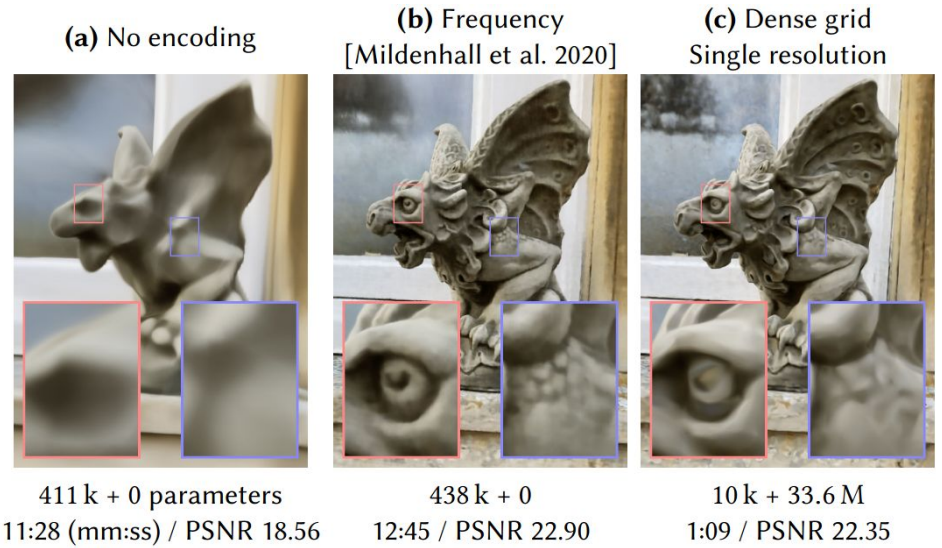
(x, y, z) →  → $\gamma(x, y, z)$ →  → (r, g, b, σ)

trilinear interpolation at sampled location

More parameters to learn… But actually, only 8 more per sample, so little overhead !

AND, we can significantly reduce the size of the MLP !

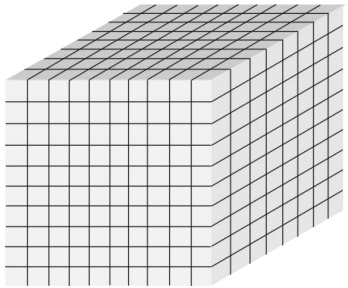
Parametric Encodings - Single level grid



MLP weights + Grid parameters
Training time / PSNR

Parametric Encodings - Single level grid

Dense feature grid (128^3) of 16-dim vectors

(x, y, z) →  → $\gamma(x, y, z)$ →  → (r, g, b, σ)

trilinear interpolation at sampled location

More parameters to learn… But actually, only 8 more per sample, so little overhead !

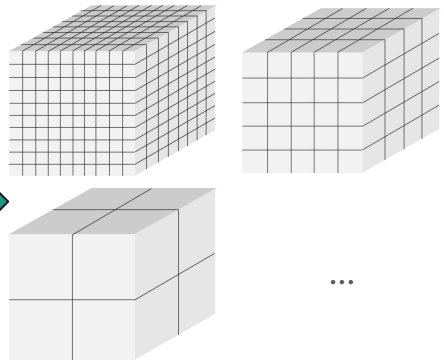
AND, we can significantly reduce the size of the MLP !

But a lot more storage needed, and lots of waste… Usually, special data structure: sparse voxel grid (Plenoxels), octree, …

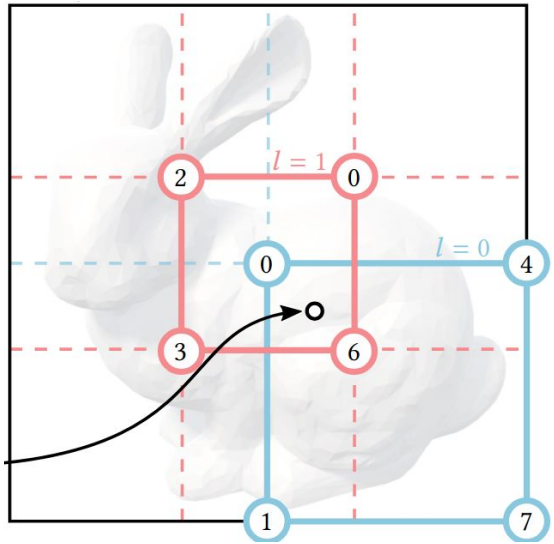
Parametric Encodings - Multi-levels grid

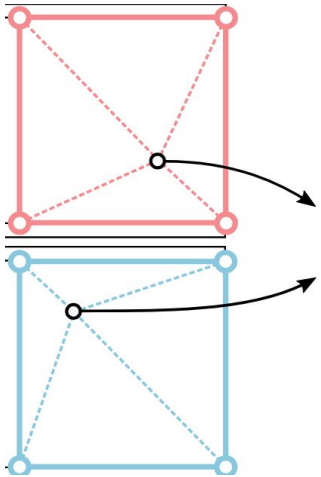
Dense feature grids (8 from 12^3 to 173^3) of 2-dim vectors

(x, y, z)

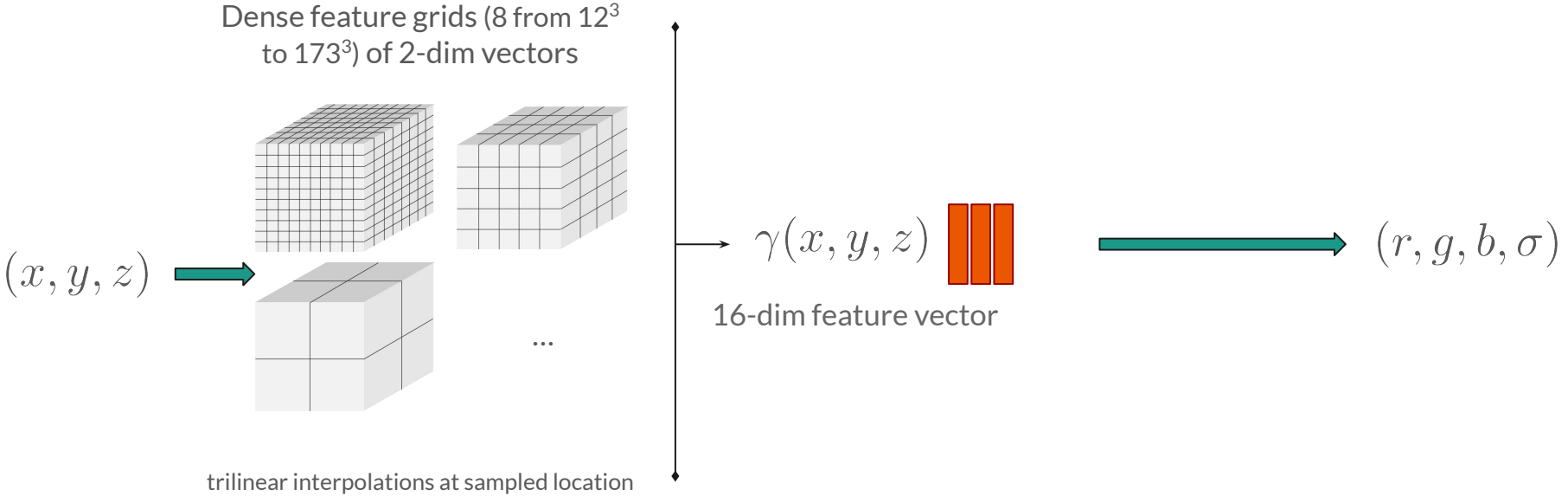


trilinear interpolations at sampled location

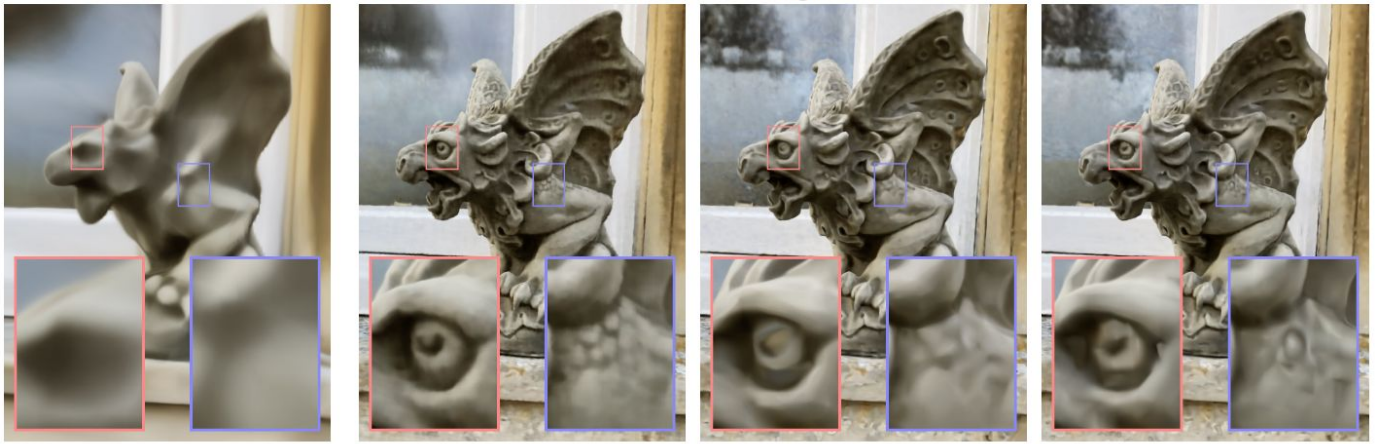




Parametric Encodings - Multi-levels grid

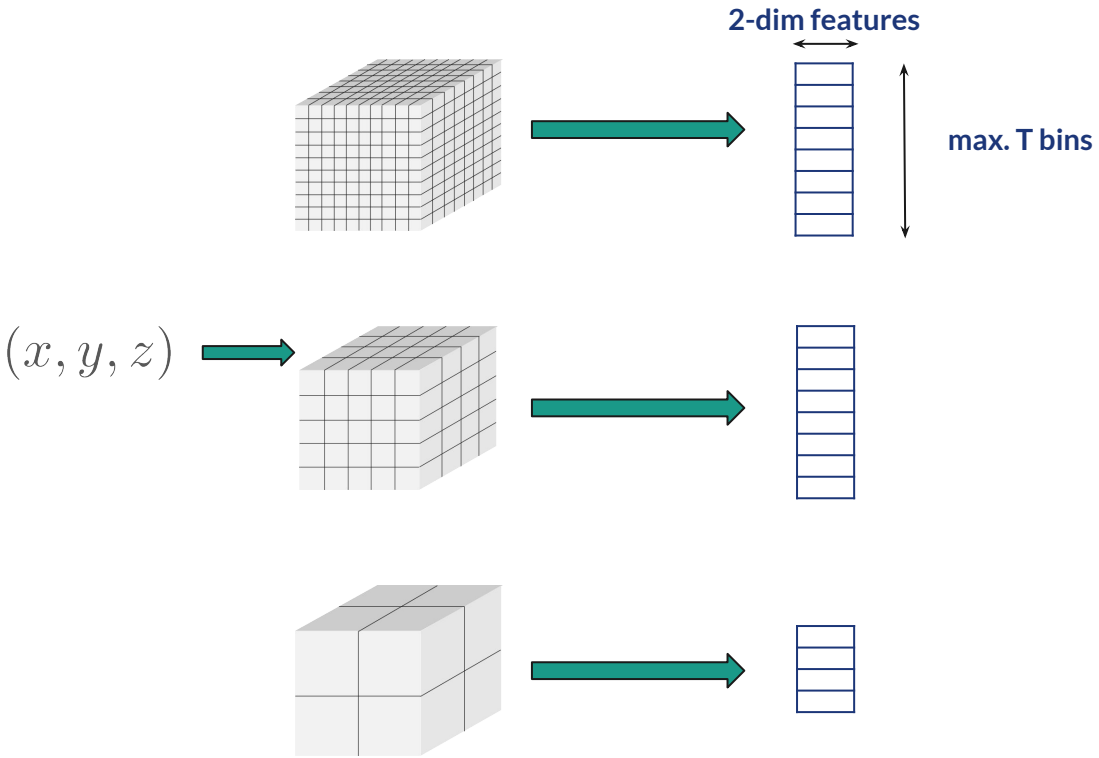


Parametric Encodings - Multi-levels grid



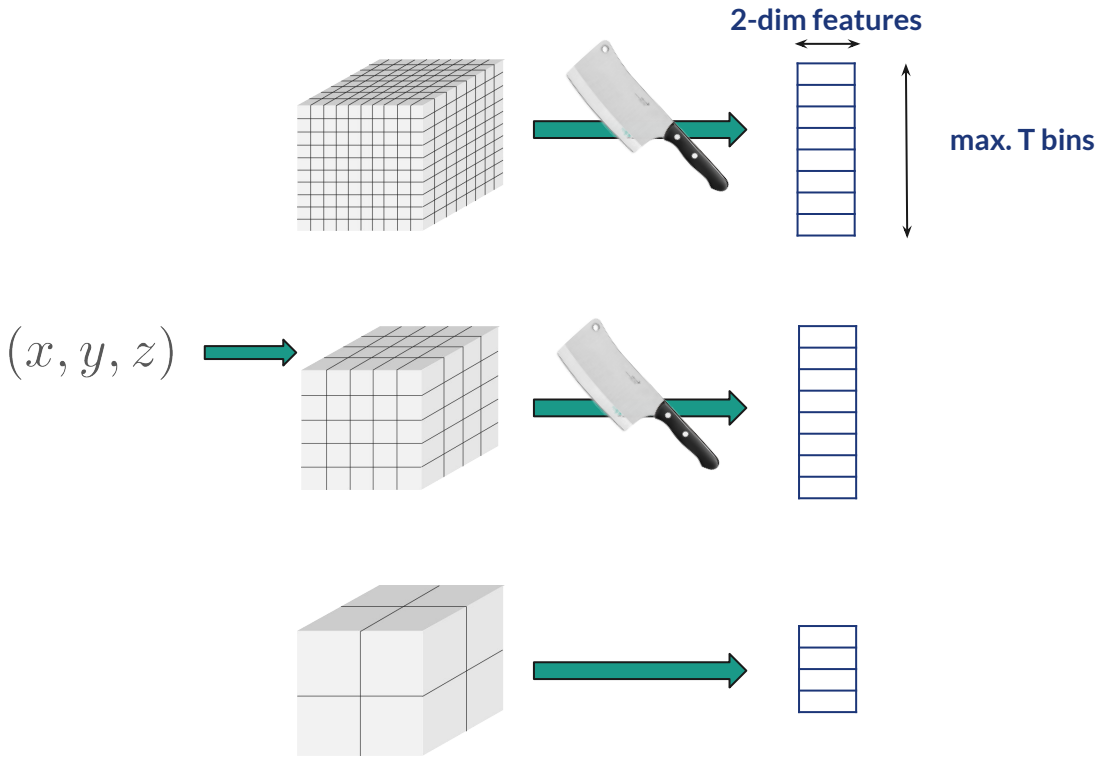
MLP weights + Grid parameters
Training time / PSNR

Parametric Encodings - InstantNGP



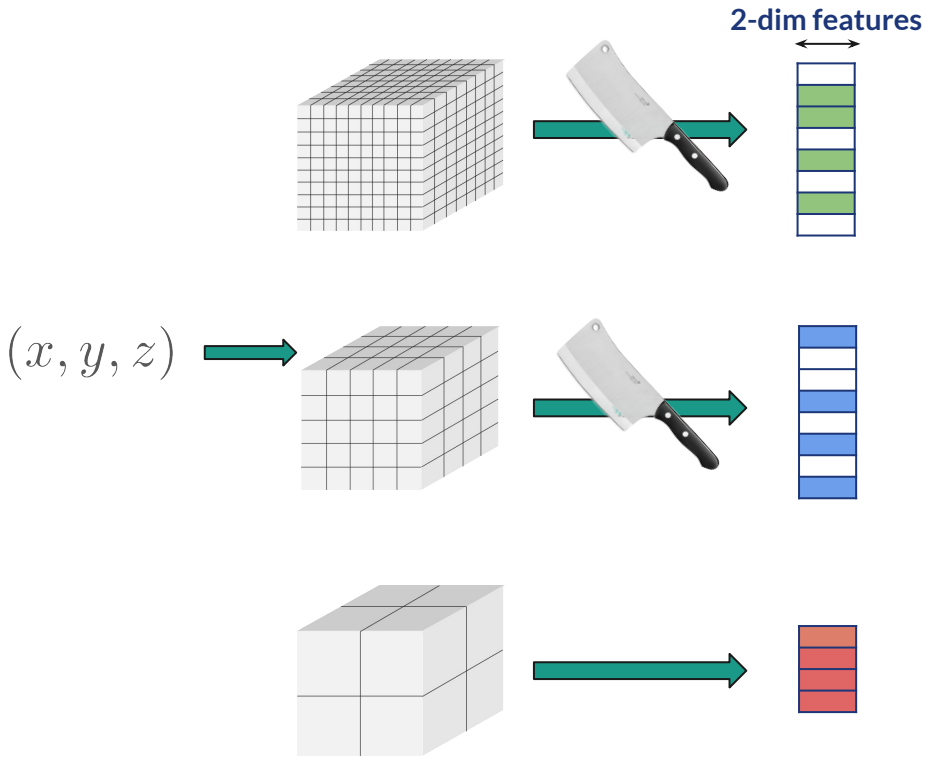
Müller et al, 2022

Parametric Encodings - InstantNGP



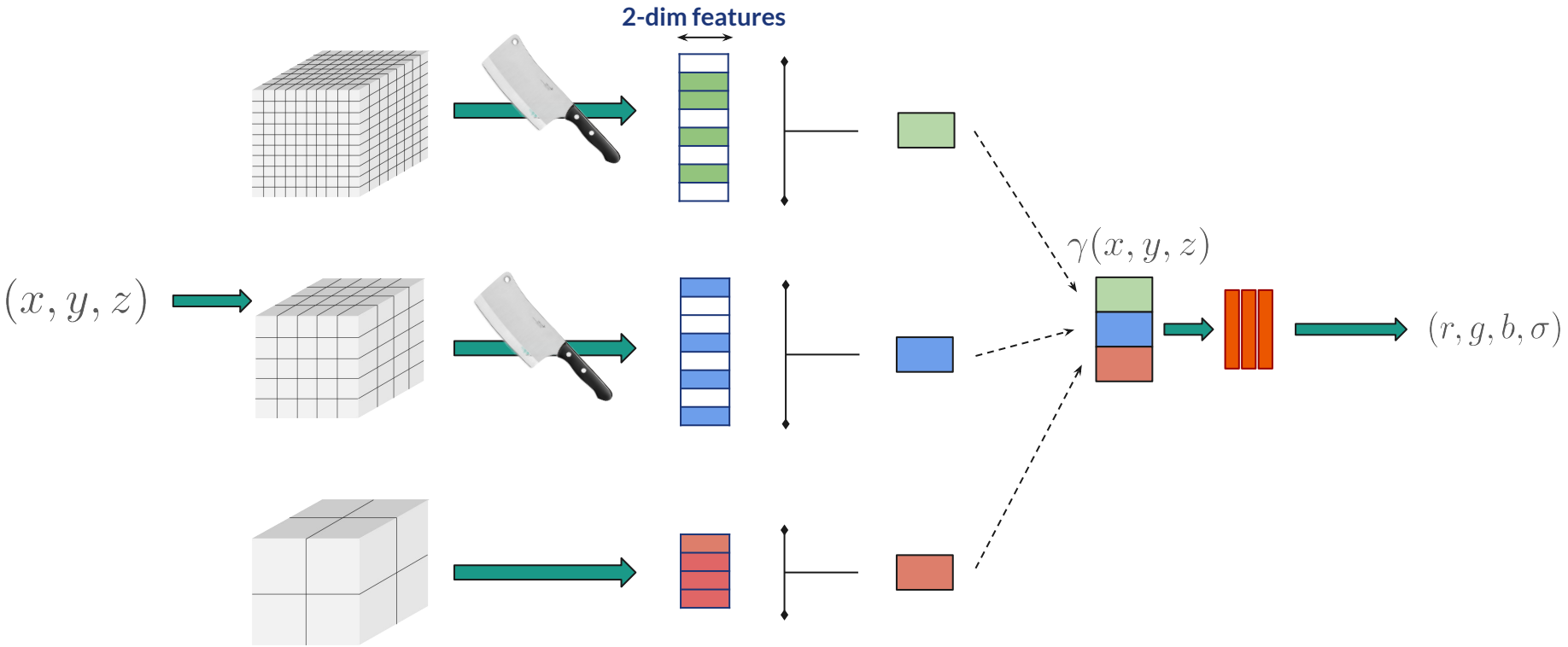
Müller et al, 2022

Parametric Encodings - InstantNGP



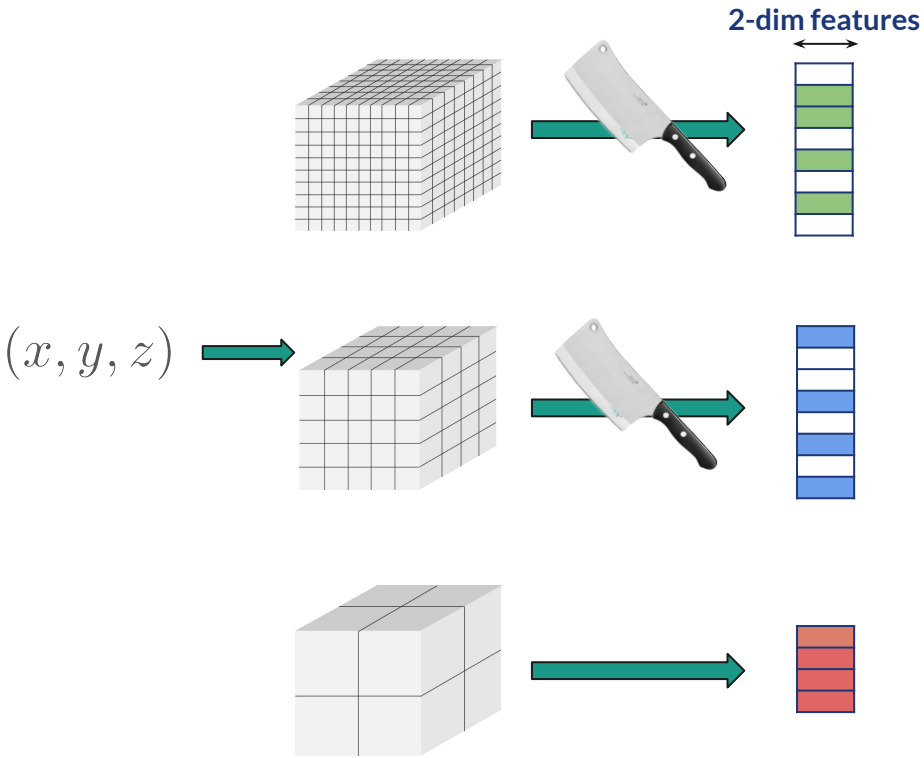
Müller et al, 2022

Parametric Encodings - InstantNGP



Müller et al, 2022

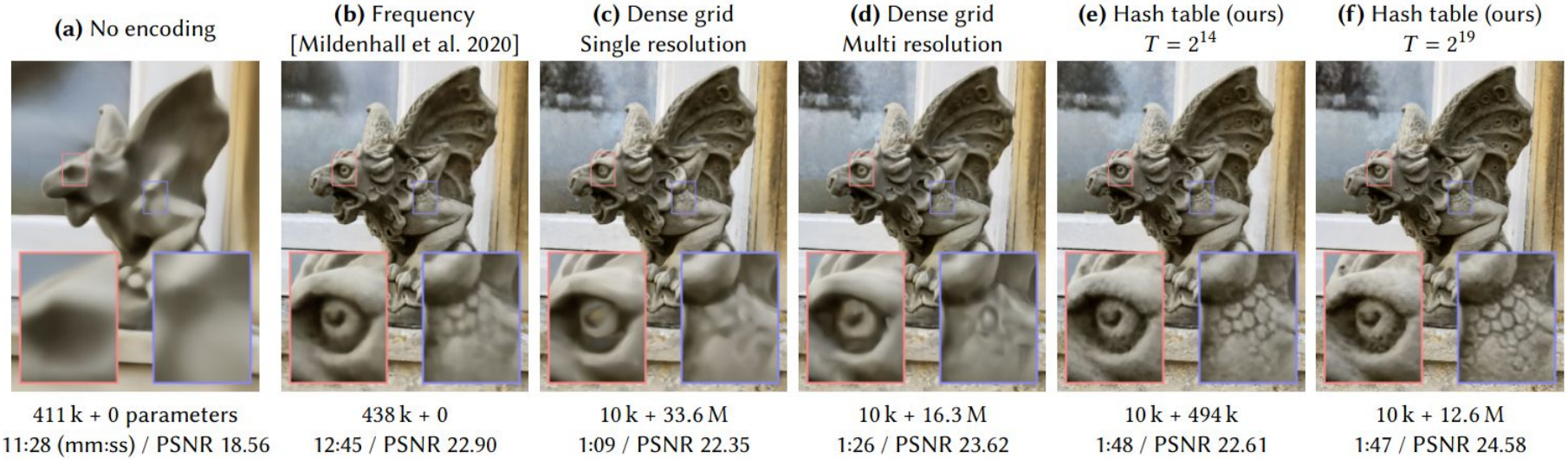
Parametric Encodings - InstantNGP



$$= \left(\bigoplus_{i=1}^{d} x_i \pi_i \right) \mod T$$

Müller et al, 2022

Parametric Encodings - InstantNGP

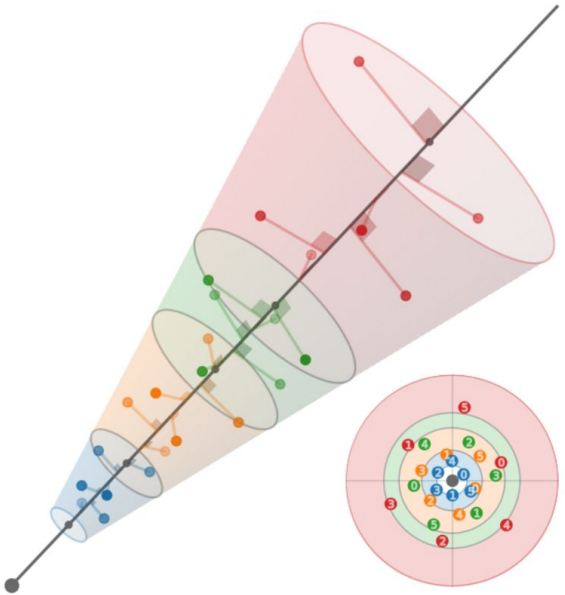


Müller et al, 2022

InstantNGP examples

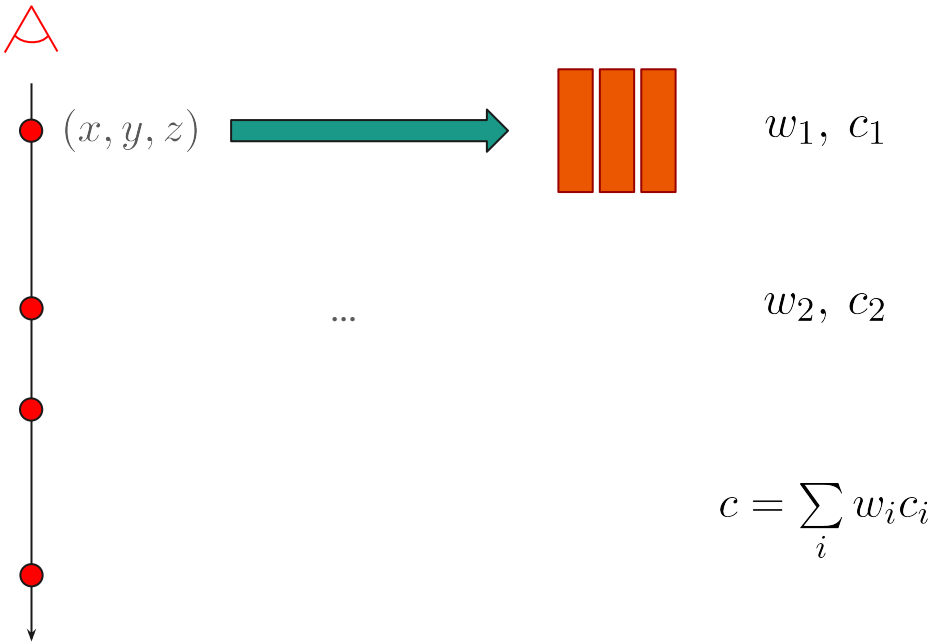
Parametric Encodings - Zip-NeRF

Weighted average of 6 samples in each cone section, given to an iNGP



SNeRG and Deferred shading

Former model:



SNeRG and Deferred shading

Deferred model: only **one call to the MLP per ray**

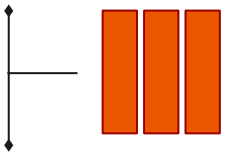
σ_1, c_1, f_1 *: opacity, color, features [1, 3, K=4]. Stored in a sparse voxel grid*

σ_2, c_2, f_2

...

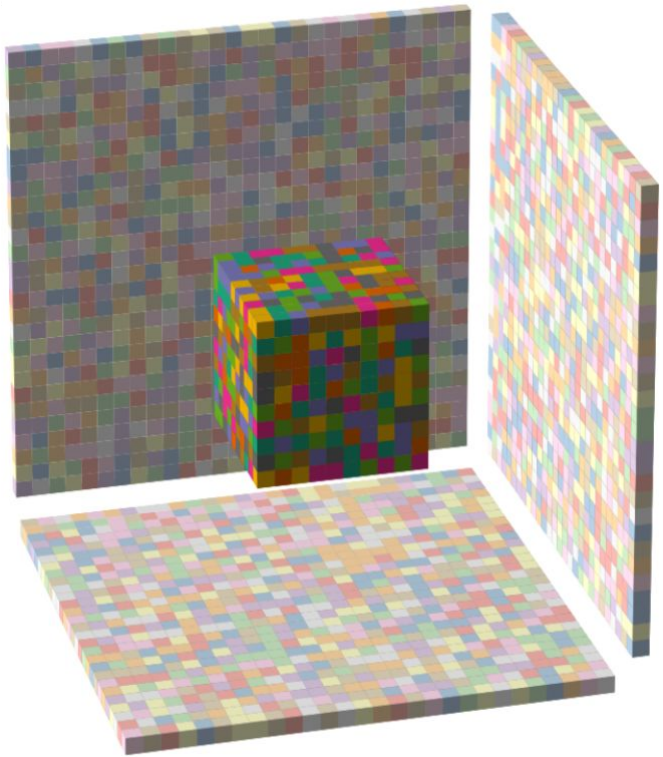
$$c_d = \sum_i w_i c_i$$

$$f = \sum_i w_i f_i$$



$$c = c_d + h_\Theta(c_d, f, v)$$

Parametric Encodings - MERF



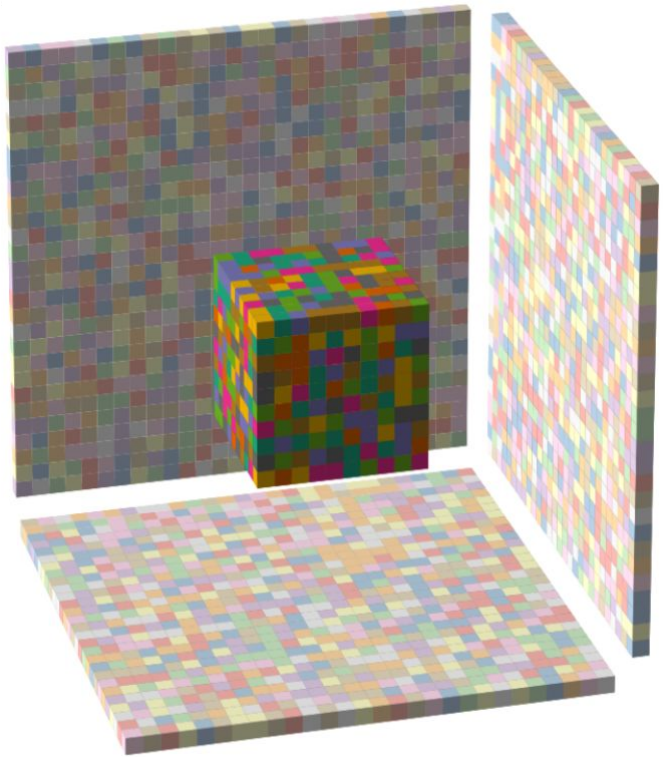
Memory-Efficient Radiance Field

Instead of storing (σ, c, f) as a sparse voxel grid, to reduce memory usage, MERF uses:

- **1 low-resolution 3D grid**
- **3 high-resolution 2D grids**

Typically, low-res = 512^3, high-res = 2048^2

Parametric Encodings - MERF



trilerp on the low-res 3D grid

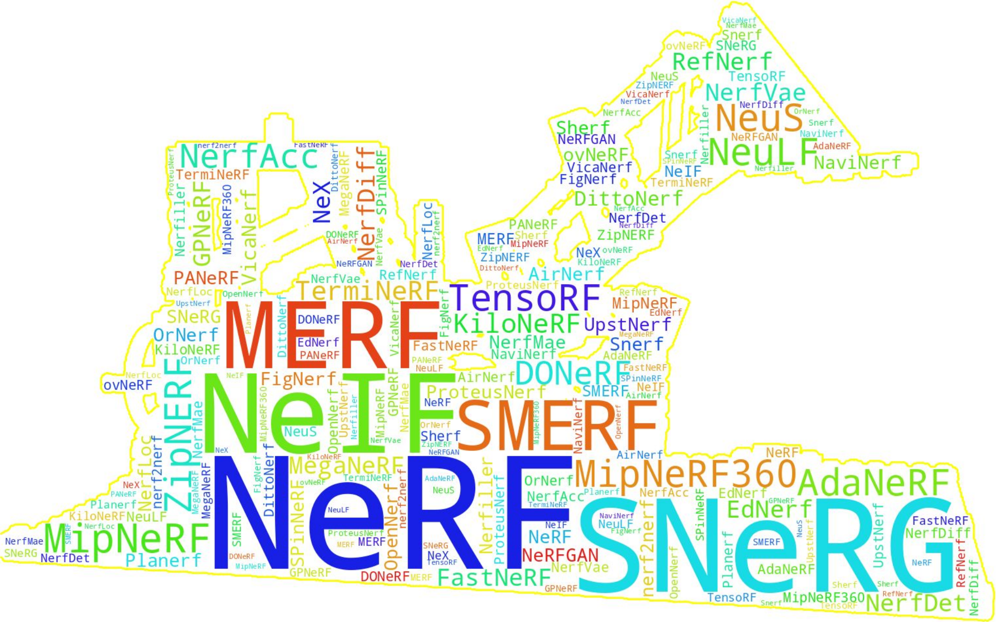
bilerp of the projection on the high-res 2D grid

$$\mathbf{V}(x,y,z) + \mathbf{P}_x(y,z) + \mathbf{P}_y(x,z) + \mathbf{P}_z(x,y)$$

$$= (\sigma, c, f)$$

Demo MERF

Questions ?


NeRF
SNeRG
NeIF
MERF
SMERF
TensoRF
KiloNeRF
MipNeRF360
AdaNeRF
MipNeRF
NeuLF
NeuS
NerfAcc
RefNerf
NerfVae
DittoNerf
DONeRF
TermiNeRF
ZipNERF
NerfDiff
FastNeRF
EdNerf
NeRFGAN
NerfDet
NaviNerf
UpstNerf
AirNerf
MegaNeRF
SPinNeRF
ProteusNerf
FigNerf
PANeRF
GPNeRF
VicaNerf
Sherf
Snerf
ovNeRF
OrNerf
Planerf
OpenNerf
nerf2nerf
Nerfiller
NerfMae
NerfLoc
MipNeRF360
NeX

References

Barron, Jonathan T., Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields.” *CoRR* abs/2103.13415 (2021). https://arxiv.org/abs/2103.13415.

Barron, Jonathan T., Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields.” *CoRR* abs/2111.12077 (2021). https://arxiv.org/abs/2111.12077.

Barron, Jonathan T, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. “Zip-Nerf: Anti-Aliased Grid-Based Neural Radiance Fields.” In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19697–705, 2023.

Duckworth, Daniel, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. “SMERF: Streamable Memory Efficient Radiance Fields for Real-Time Large-Scene Exploration,” 2024. https://arxiv.org/abs/2312.07541.

Hedman, Peter, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. “Baking Neural Radiance Fields for Real-Time View Synthesis.” *ICCV*, 2021.

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.” *CoRR* abs/2003.08934 (2020). https://arxiv.org/abs/2003.08934.

Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding.” *ACM Trans. Graph.* 41, no. 4 (July 2022): 102:1-102:15. https://doi.org/10.1145/3528223.3530127.

Reiser, Christian, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. “MERF: Memory-Efficient Radiance Fields for Real-Time View Synthesis in Unbounded Scenes.” *ACM Transactions on Graphics (TOG)* 42 (2023): 1–12.

Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains.” *CoRR* abs/2006.10739 (2020). https://arxiv.org/abs/2006.10739.

Wang, Peng, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. “NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-View Reconstruction,” 2023. https://arxiv.org/abs/2106.10689.

Yu, Alex, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. “Plenoxels: Radiance Fields without Neural Networks.” *CoRR* abs/2112.05131 (2021). https://arxiv.org/abs/2112.05131.

Yu, Alex, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. “PlenOctrees for Real-Time Rendering of Neural Radiance Fields.” In *ICCV*, 2021.