Seminar: Recent Advances in 3D Computer Vision

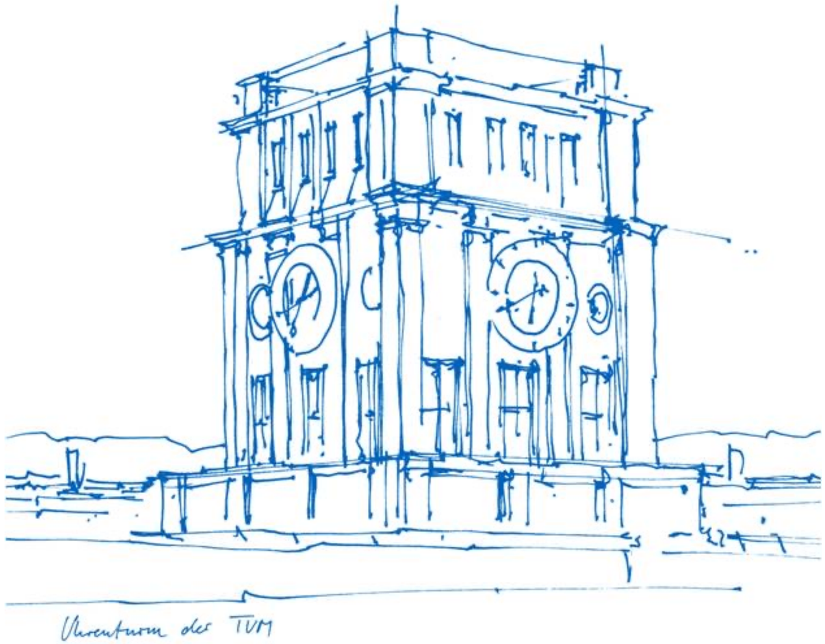
Speaker: Lukas Schneidt

Supervisor: Björn Häfner

Technische Universität München

Computer Science – Computer Vision Group

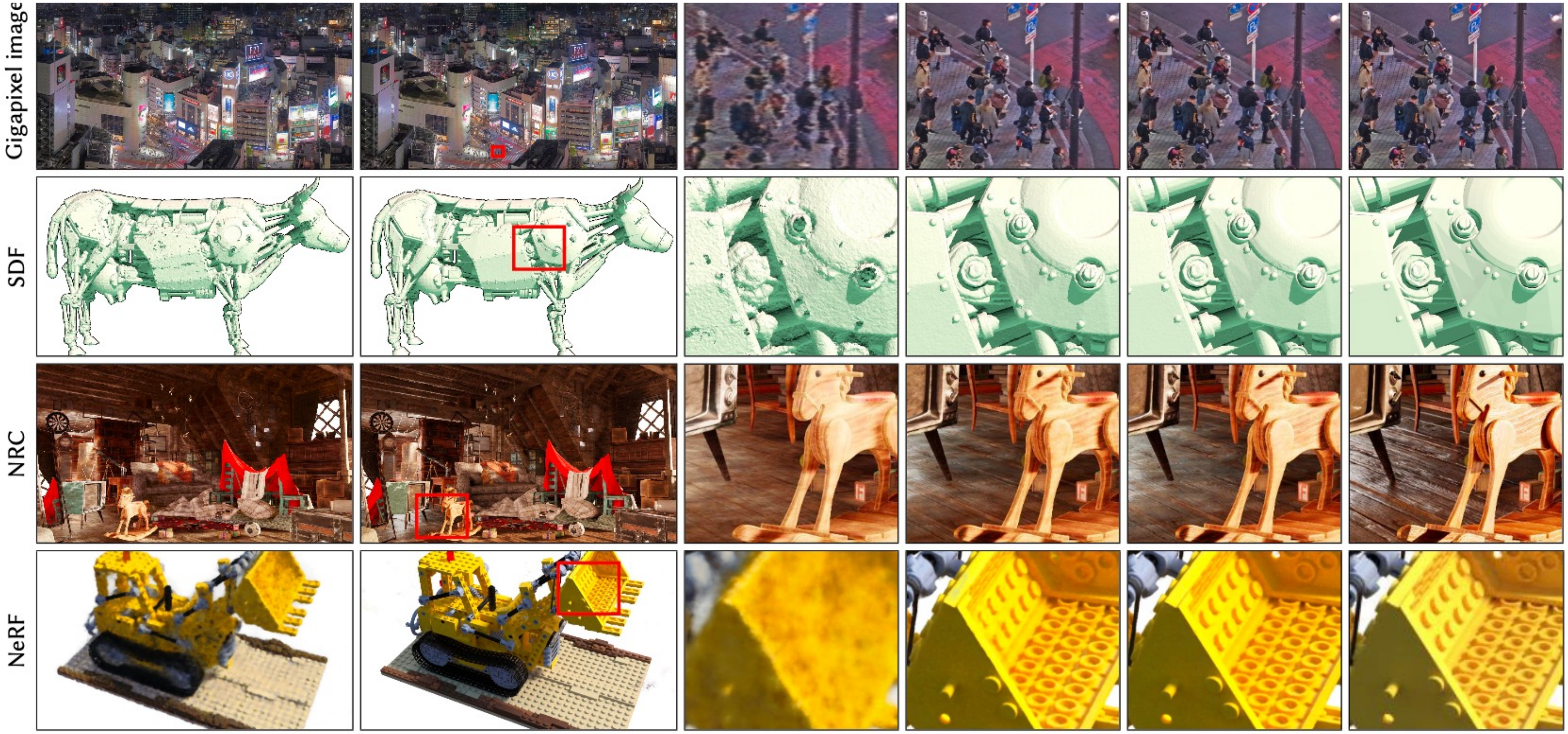
München, 11.10.2022



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Thomas Müller, NVIDIA, Switzerland, Alex Evans, NVIDIA, United Kingdom, Christoph Schied, NVIDIA, USA, Alexander Keller, NVIDIA, Germany

SIGGRAPH 2022 – Best paper award

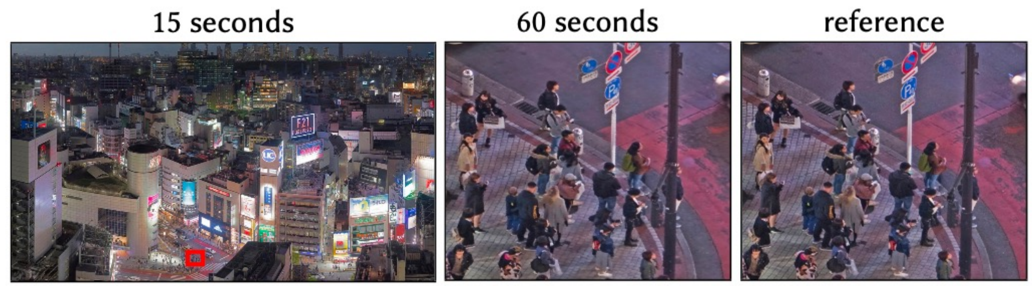


Content

- Introduction
- Background and Related Work
- Multiresolution Hash Encoding
- Experiments
- Discussion and Future Work
- Summary

Introduction

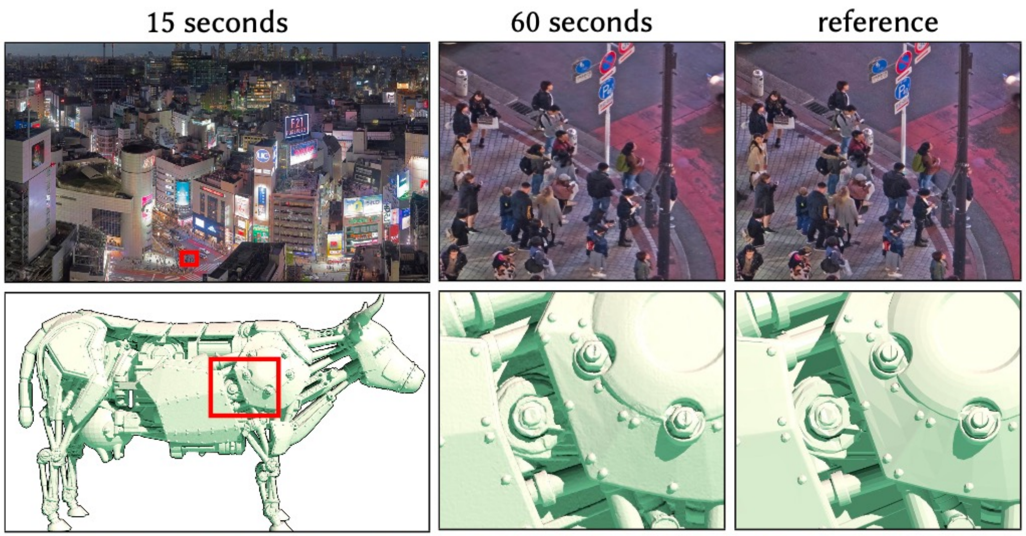
Gigapixel Image



Introduction

Gigapixel Image

Neural signed distance functions (SDF)

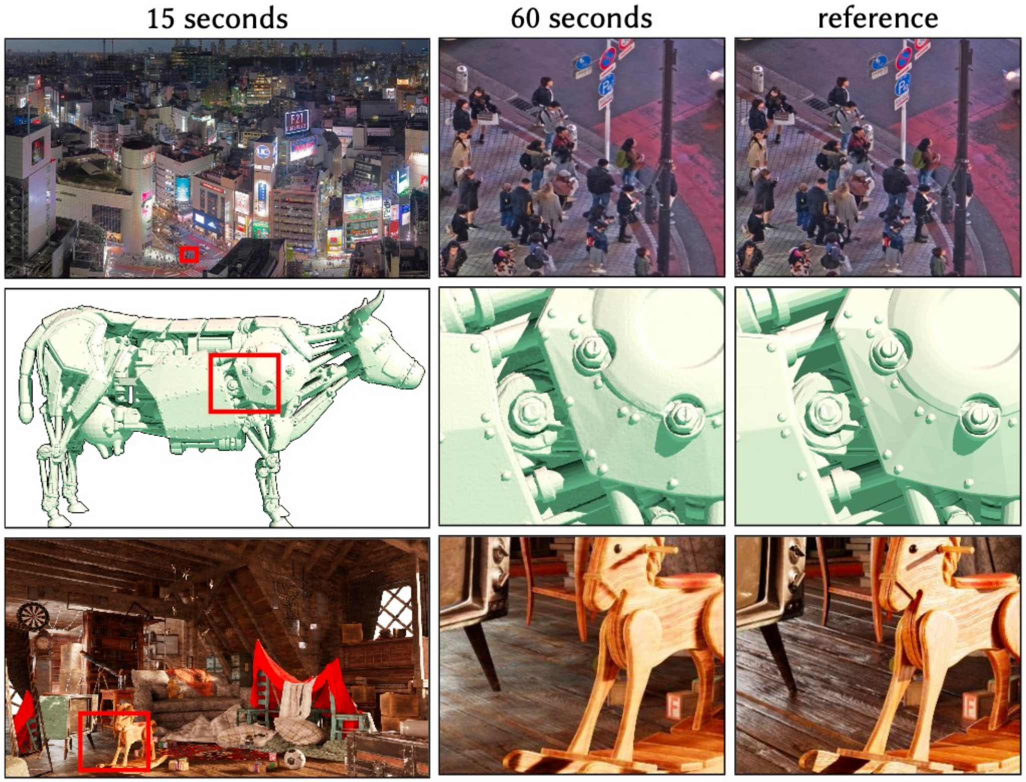


Introduction

Gigapixel Image

Neural signed distance functions (SDF)

Neural radiance caching (NRC)



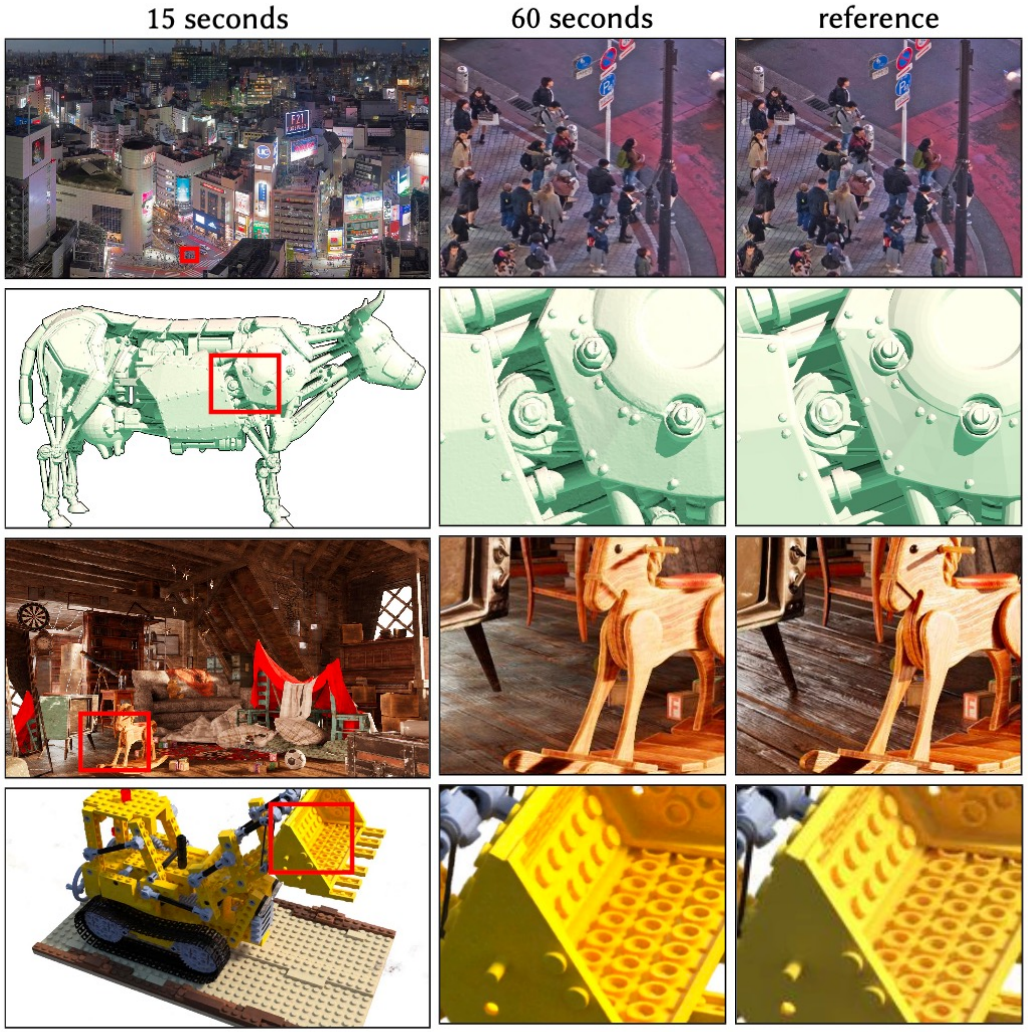
Introduction

Gigapixel Image

Neural signed distance functions (SDF)

Neural radiance caching (NRC)

Neural radiance and density fields (NeRF)



Background and Related Work

Parametric encodings

Arrange additional trainable parameters in an auxiliary data structure, such as a grid or a tree

Look-up and interpolate parameters

Trade-off between larger memory footprint and smaller computational cost

Background and Related Work

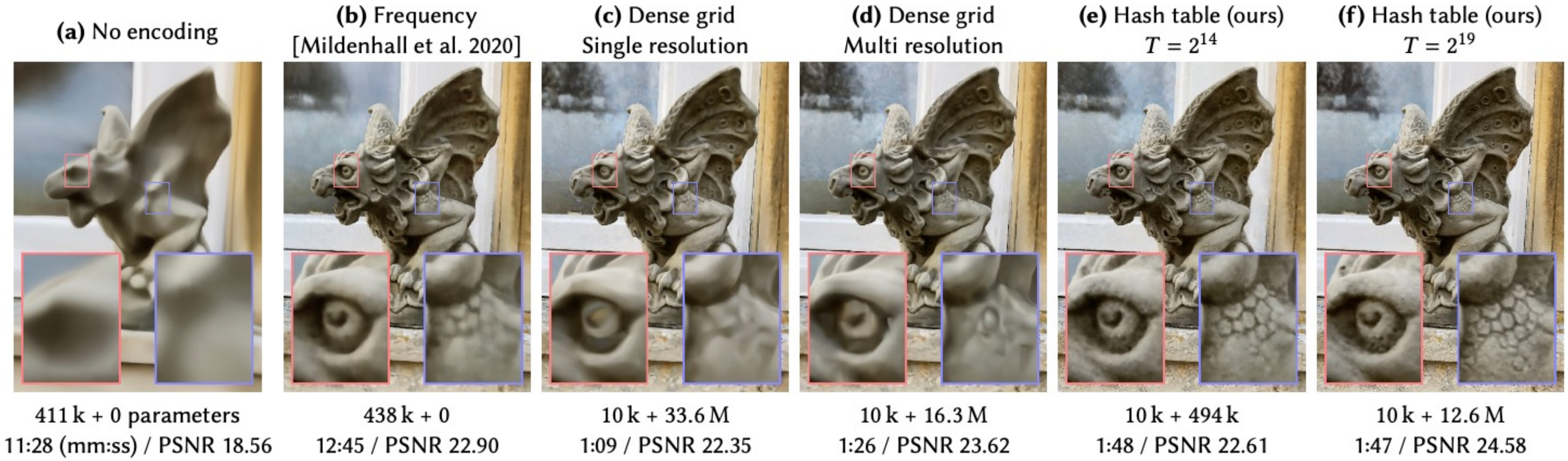
Parametric encodings

Arrange additional trainable parameters in an auxiliary data structure, such as a grid or a tree

Look-up and interpolate parameters

Trade-off between larger memory footprint and smaller computational cost

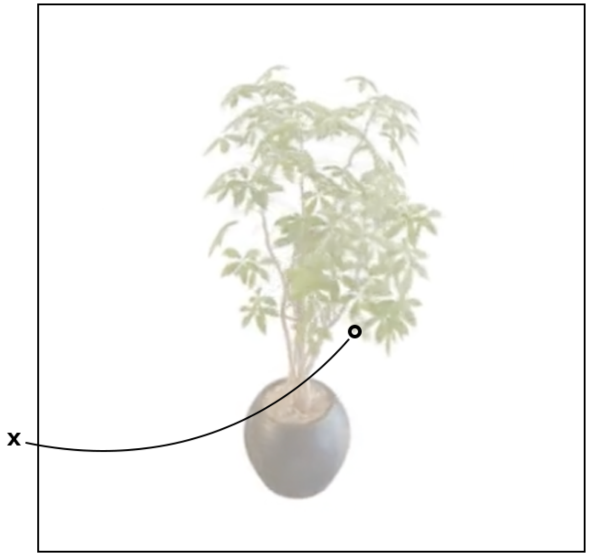
Sparse parametric encodings



Multiresolution Hash Encoding

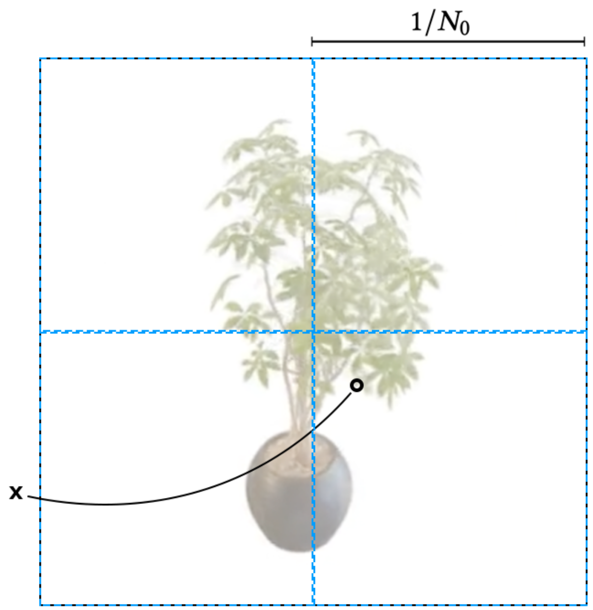
(1) Hashing of voxel vertices

Multiresolution Hash Encoding



(1) Hashing of voxel vertices

Multiresolution Hash Encoding

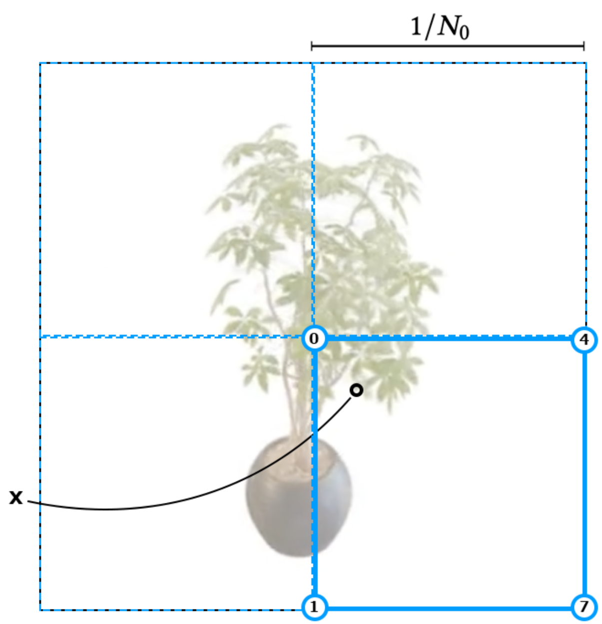


(1) Hashing of voxel vertices

1. Scale Input x
   1. $b := \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L := \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$

2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$

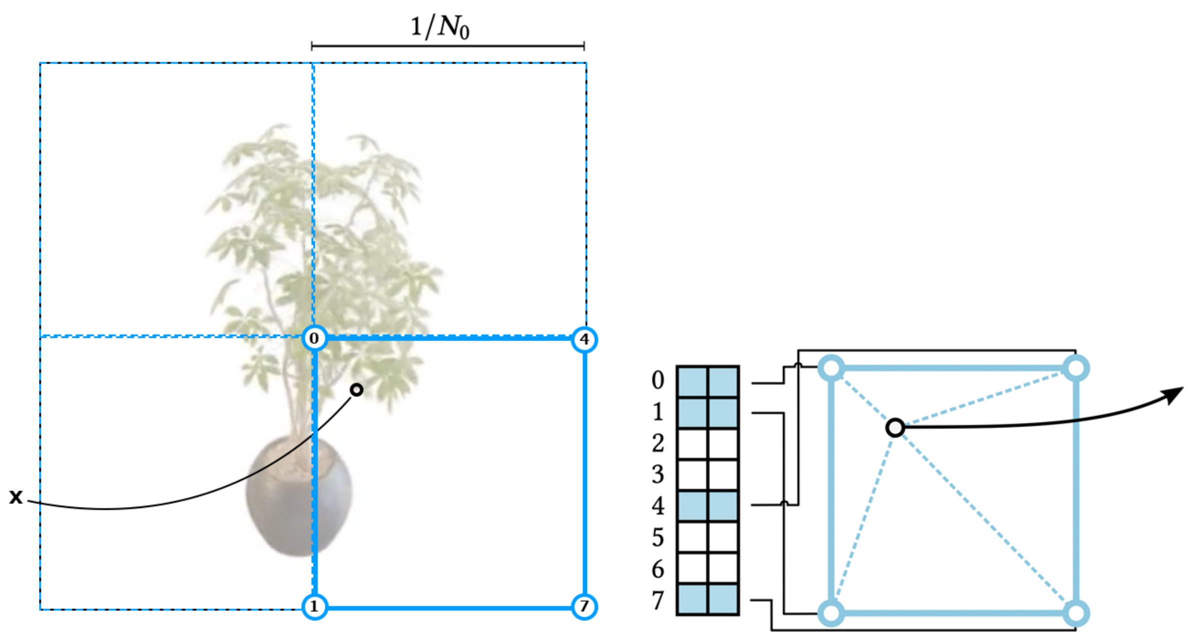
Multiresolution Hash Encoding



(1) Hashing of voxel vertices

1. Scale Input x
   1. $b \coloneqq \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L \coloneqq \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$
2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$
3. Span voxel

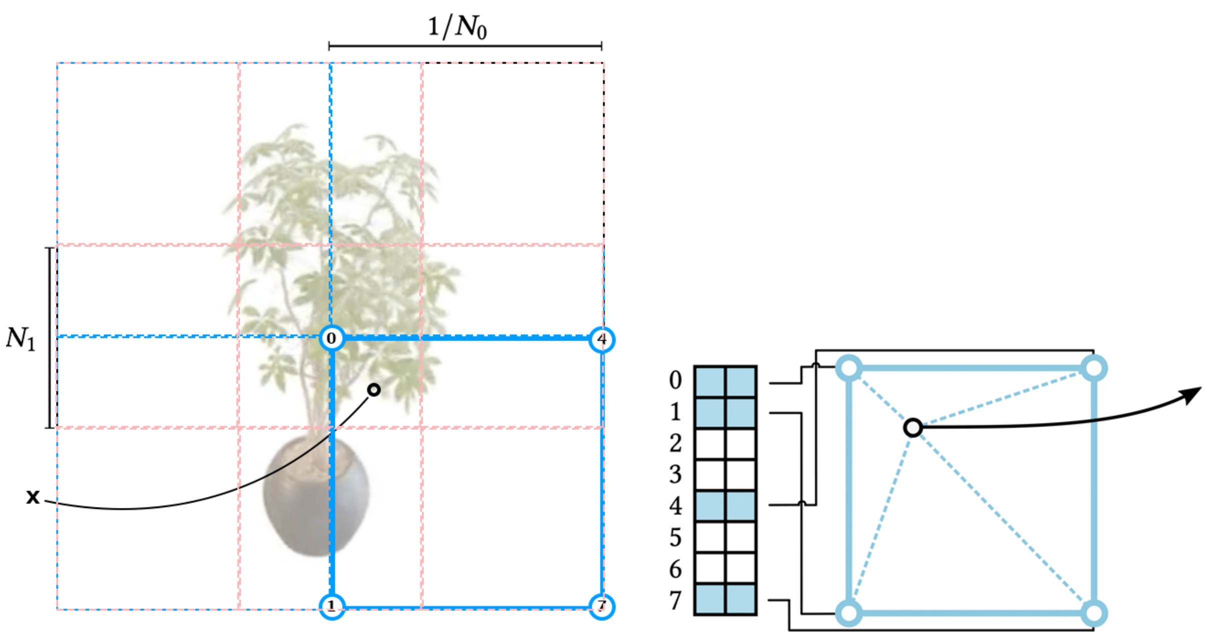
Multiresolution Hash Encoding



(1) Hashing of voxel vertices (2) Lookup (3) Linear Interpolation

1. Scale Input x
   1. $b \coloneqq \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L \coloneqq \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$
2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$
3. Span voxel
4. Map corners to entries in respective feature vector and interpolate

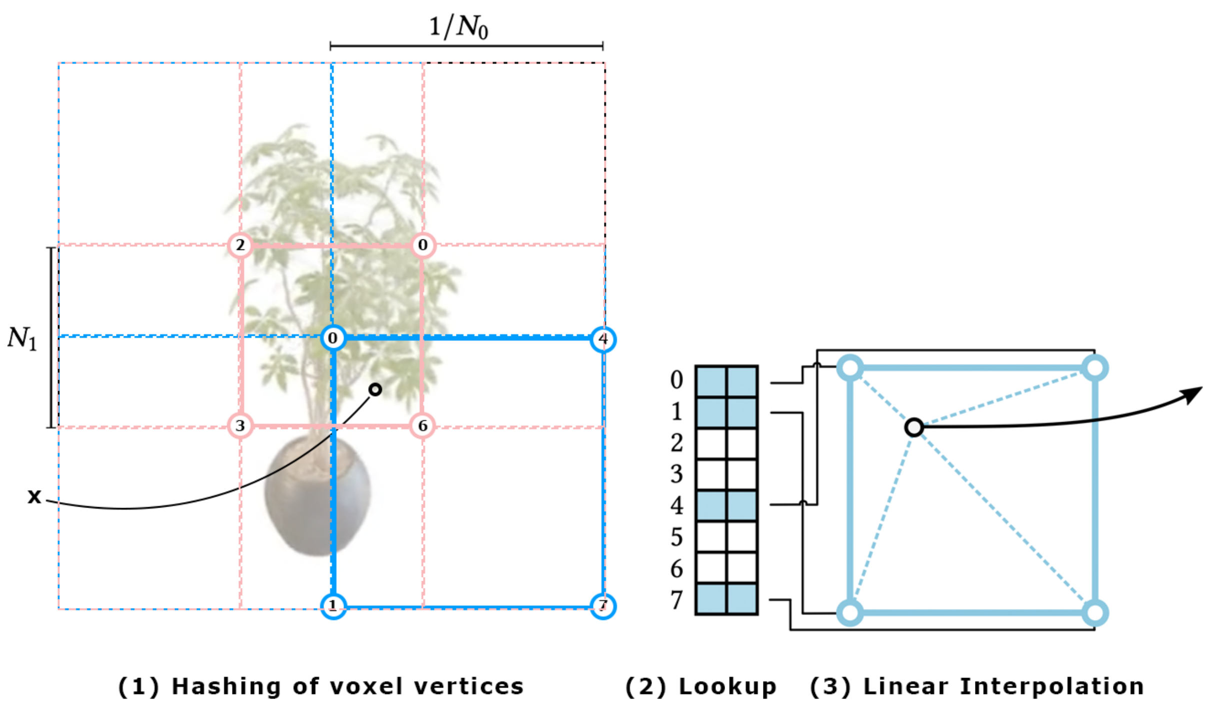
Multiresolution Hash Encoding



(1) Hashing of voxel vertices (2) Lookup (3) Linear Interpolation

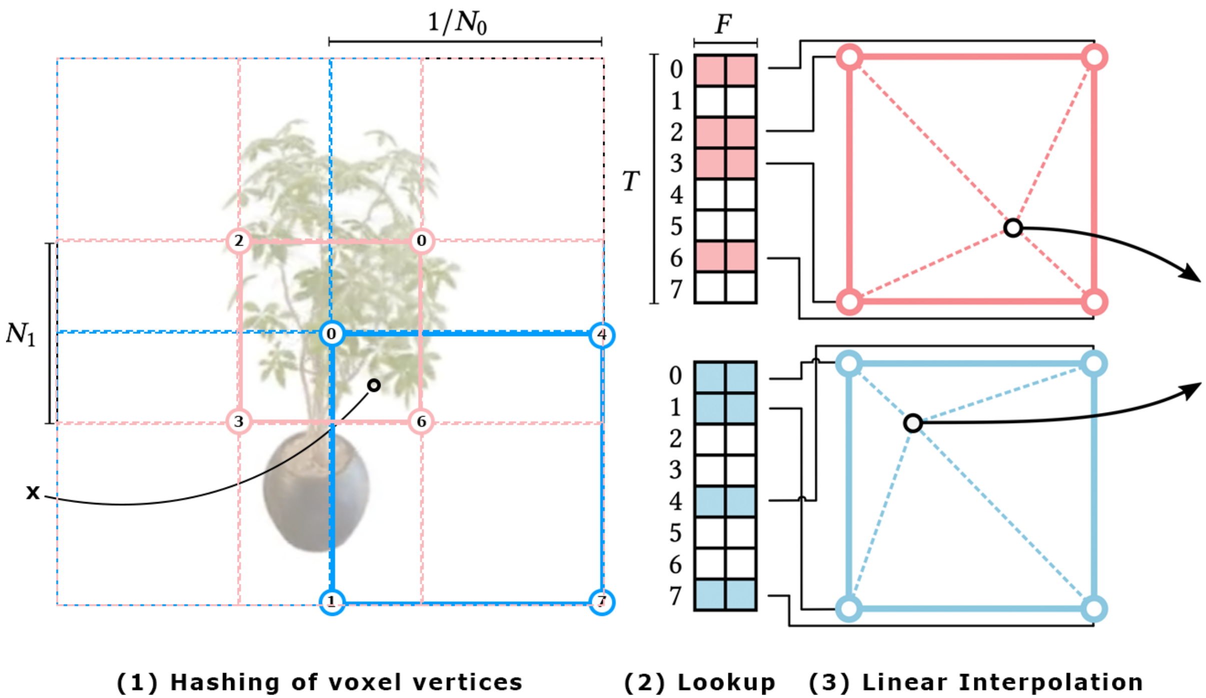
1. Scale Input x
   1. $b := \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L := \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$
2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$
3. Span voxel
4. Map corners to entries in respective feature vector and interpolate
5. Repeat for all resolutions

Multiresolution Hash Encoding



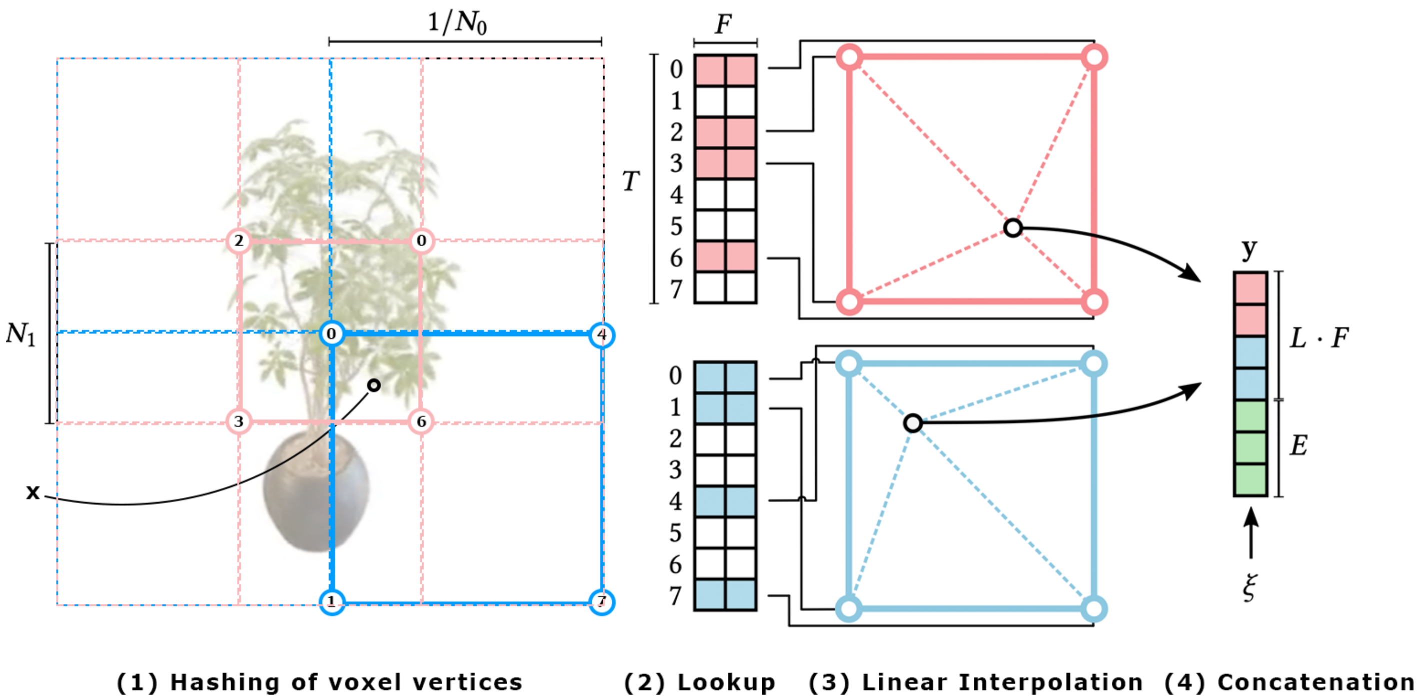
1. Scale Input x
   1. $b := \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L := \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$
2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$
3. Span voxel
4. Map corners to entries in respective feature vector and interpolate
5. Repeat for all resolutions

Multiresolution Hash Encoding

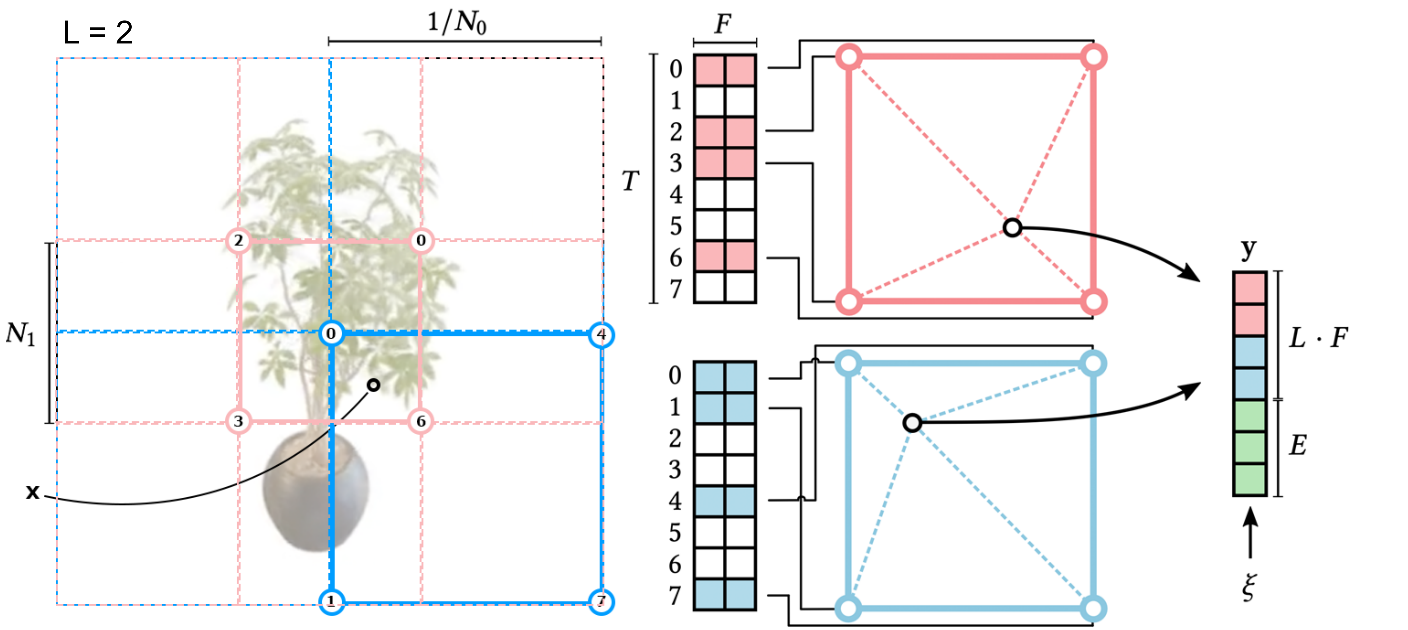


1. Scale Input x
   1. $b \ := \exp(\frac{\ln N_{max} - \ln N_{min}}{L-1})$
   2. $N_L \ := \lfloor N_{min} * b^l \rfloor$
   3. $x \ * \ N_l$
2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x \ * \ N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x \ * \ N_l \rceil$
3. Span voxel
4. Map corners to entries in respective feature vector and interpolate
5. Repeat for all resolutions

Multiresolution Hash Encoding



Multiresolution Hash Encoding



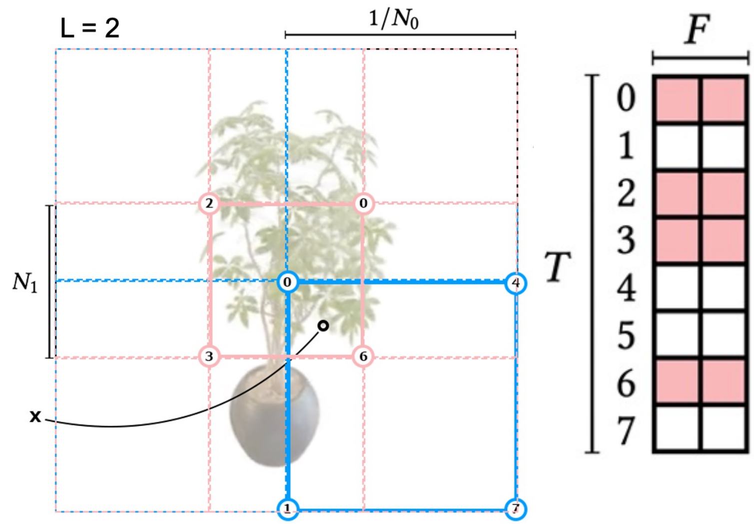
(1) Hashing of voxel vertices (2) Lookup (3) Linear Interpolation (4) Concatenation

Multiresolution Hash Encoding
Implicit Hash Collision Resolution

Finer resolution levels:

+ Capture small features

- Many collisions



Multiresolution Hash Encoding
Implicit Hash Collision Resolution

Finer resolution levels:

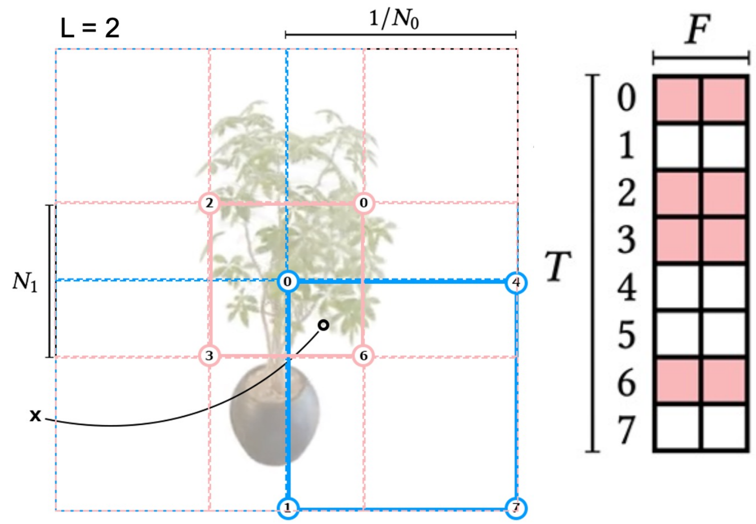
+ Capture small features

- Many collisions

Coarser resolution levels:

+ No Collisions

- Only represent low-resolution scene

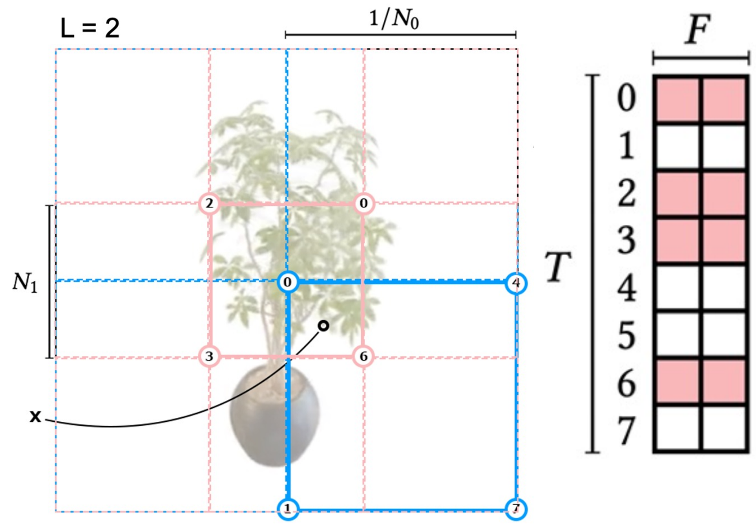


Multiresolution Hash Encoding
Implicit Hash Collision Resolution

Finer resolution levels:

+ Capture small features

- Many collisions



Coarser resolution levels:

+ No Collisions

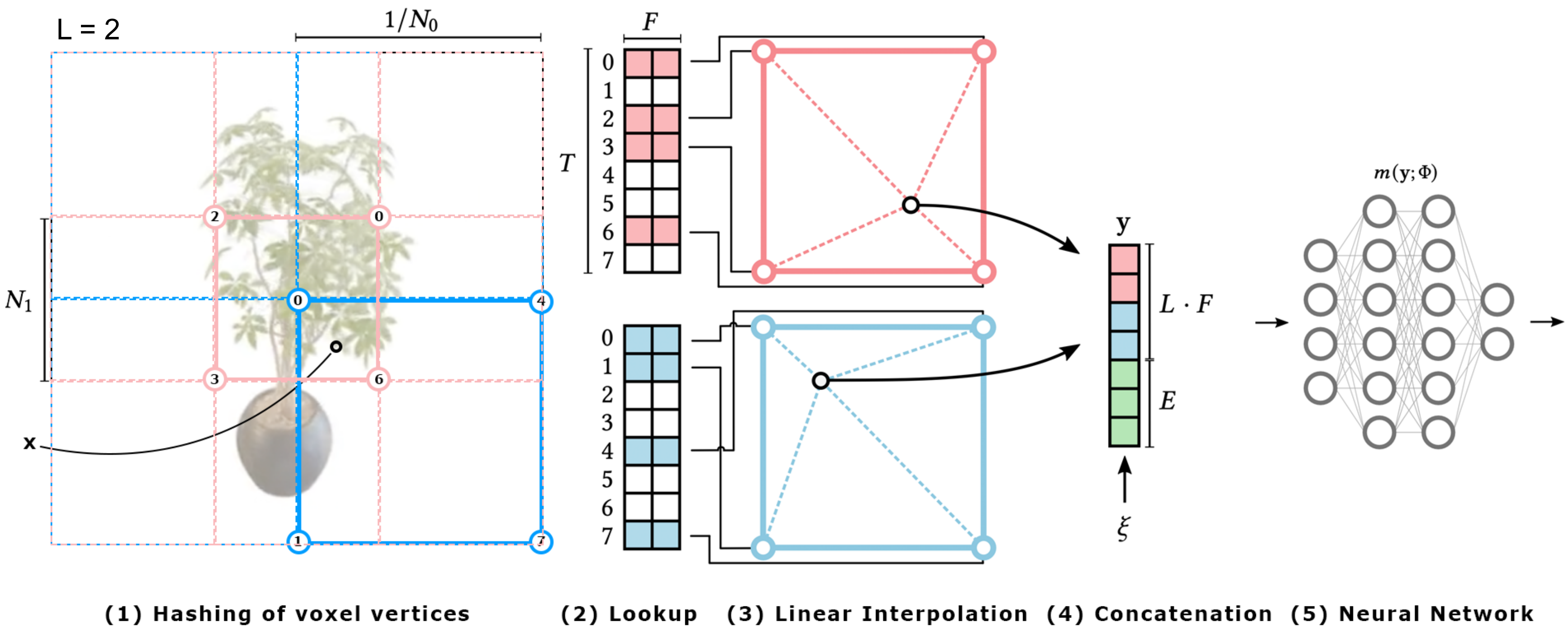
- Only represent low-resolution scene

Collision → average gradients:

Point on surface of radiance field contributes strongly

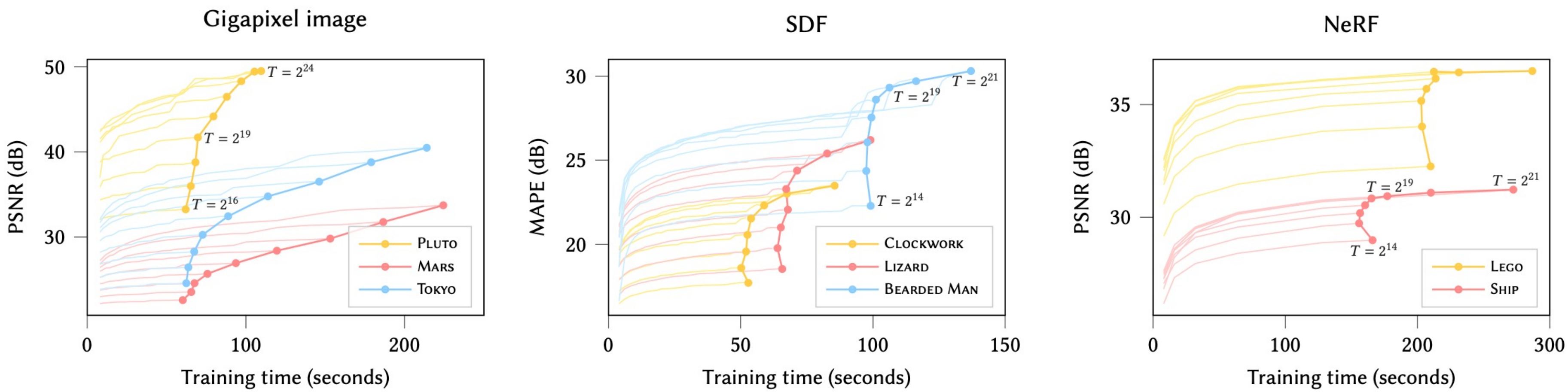
Point in empty space contributes weakly

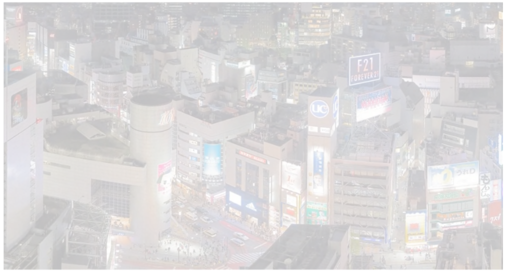
Multiresolution Hash Encoding

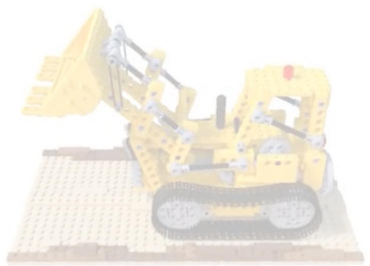


Multiresolution Hash Encoding
Performance vs. Quality

Hash Table Size: *T*



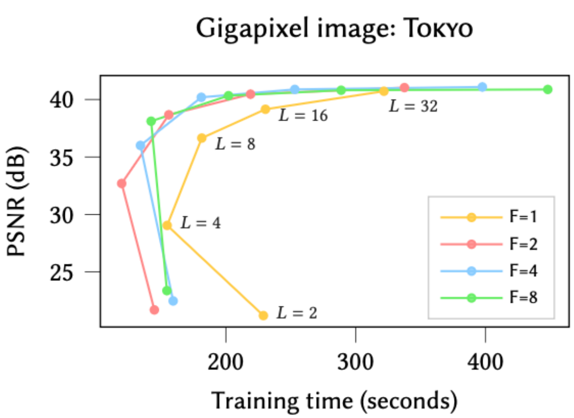


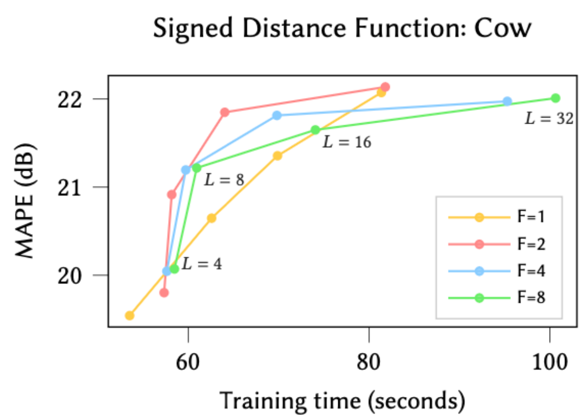


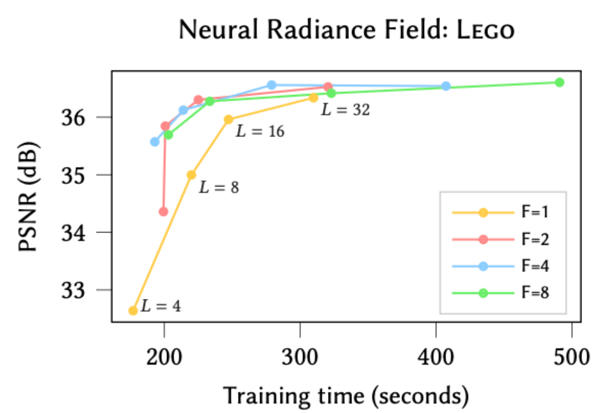
Multiresolution Hash Encoding
Performance vs. Quality

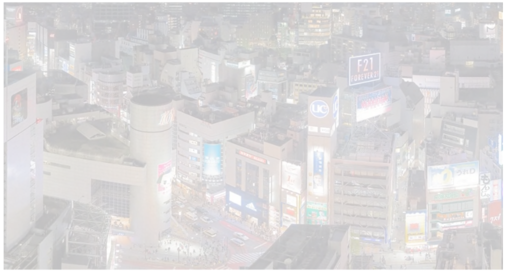
Number of Levels *L*

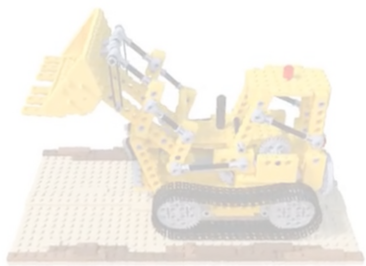
Number of feature dimensions *F*











Multiresolution Hash Encoding
Online Adaptivity and d-Linear Interpolation

Online Adaptivity:

If distribution of inputs changes during training, finer grid levels will experience fewer collisions
→ more accurate function can be learned

Multiresolution Hash Encoding
Online Adaptivity and d-Linear Interpolation

Online Adaptivity:

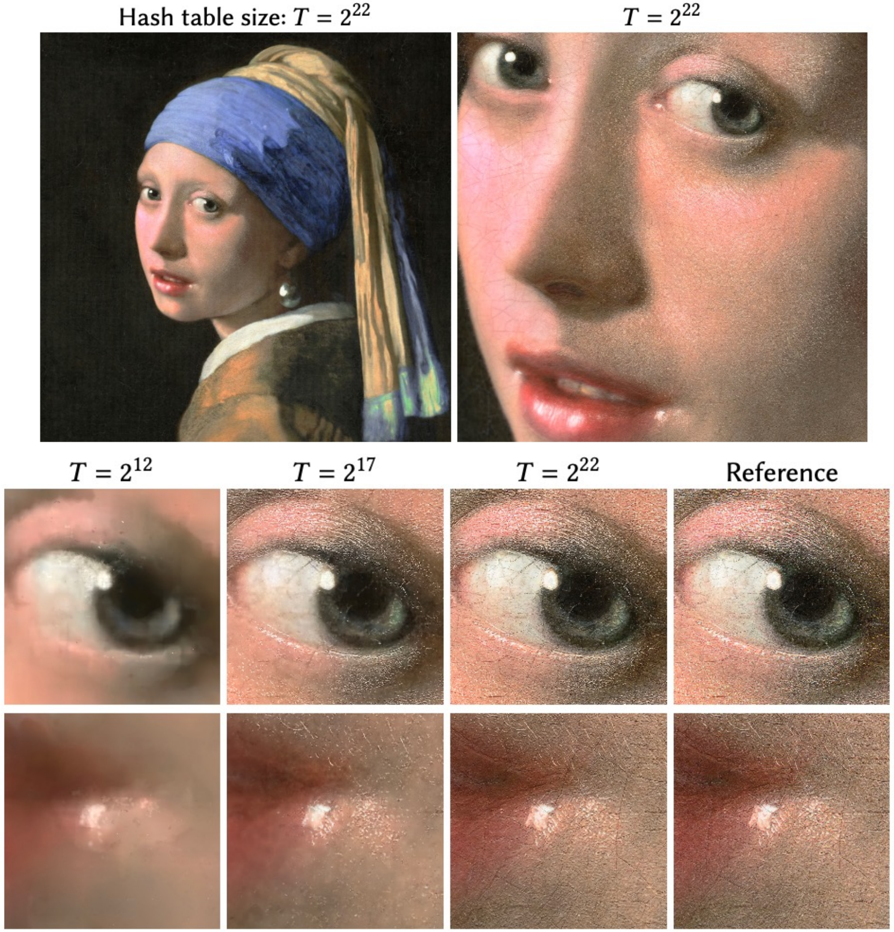
If distribution of inputs changes during training, finer grid levels will experience fewer collisions
→ more accurate function can be learned

d-linear Interpolation:

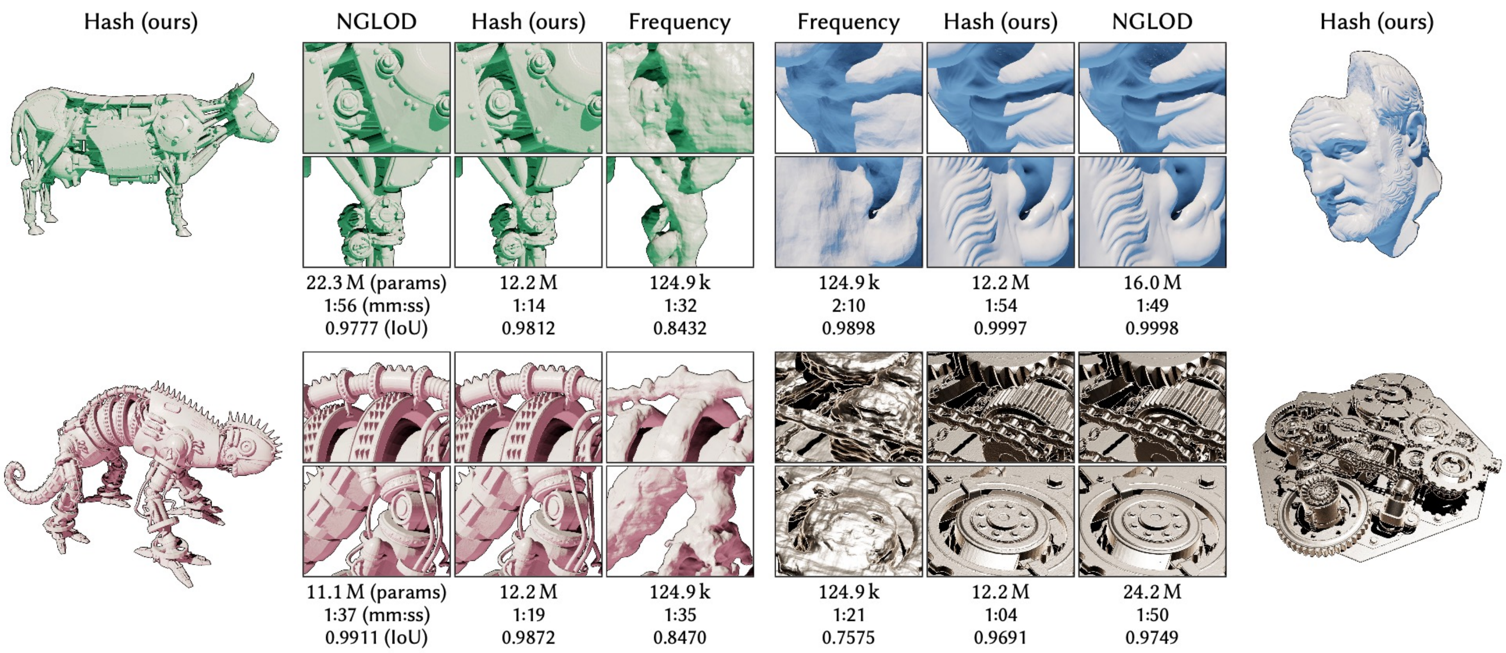
Interpolation ensures that encoding and its composition with the neural network are continuous.

Experiments
Gigapixel Image Approximation

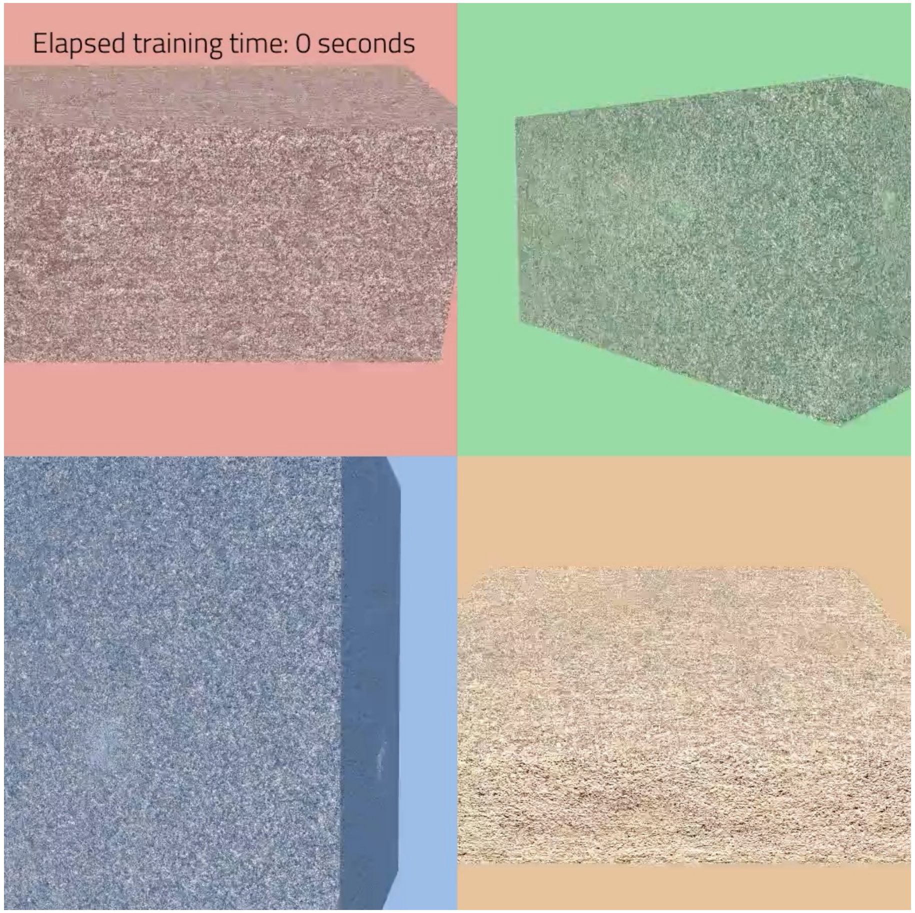
Experiments
Gigapixel Image Approximation



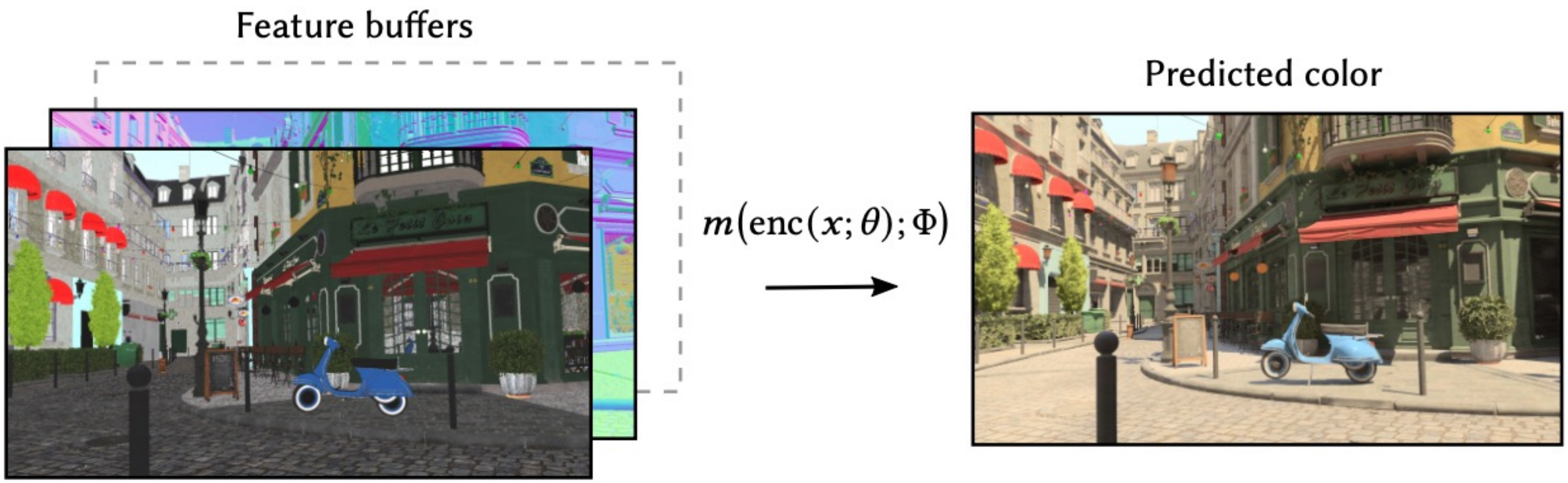
Experiments
Signed Distance Functions



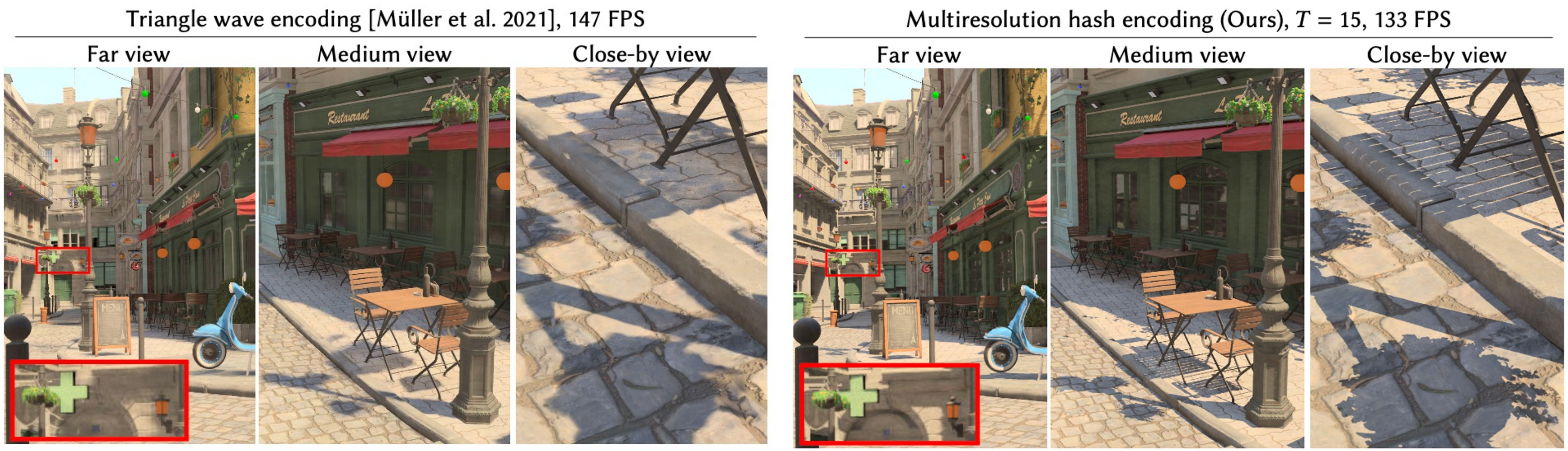
Experiments
Signed Distance Functions



Experiments
Neural Radiance Caching

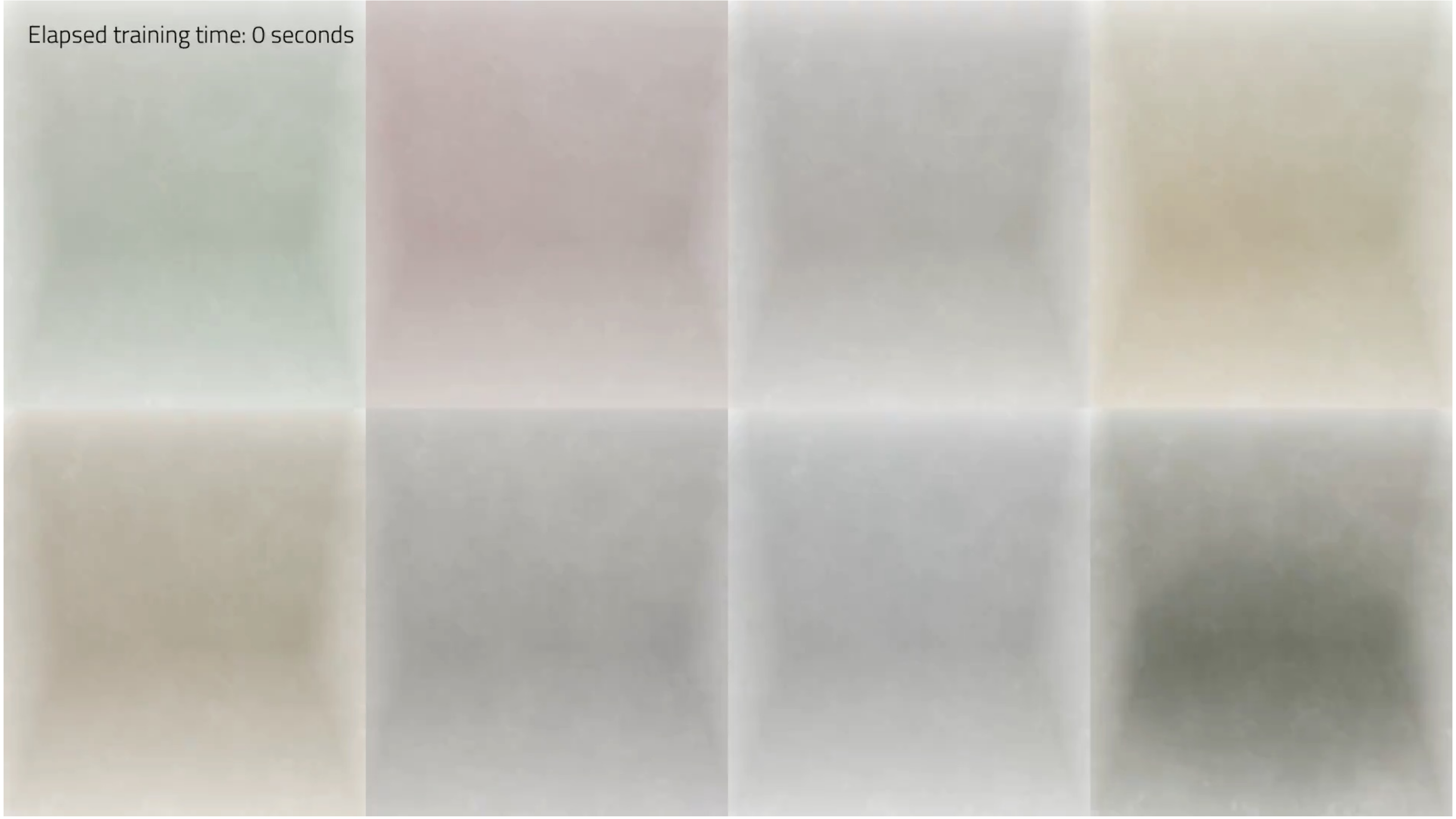


Experiments
Neural Radiance Caching

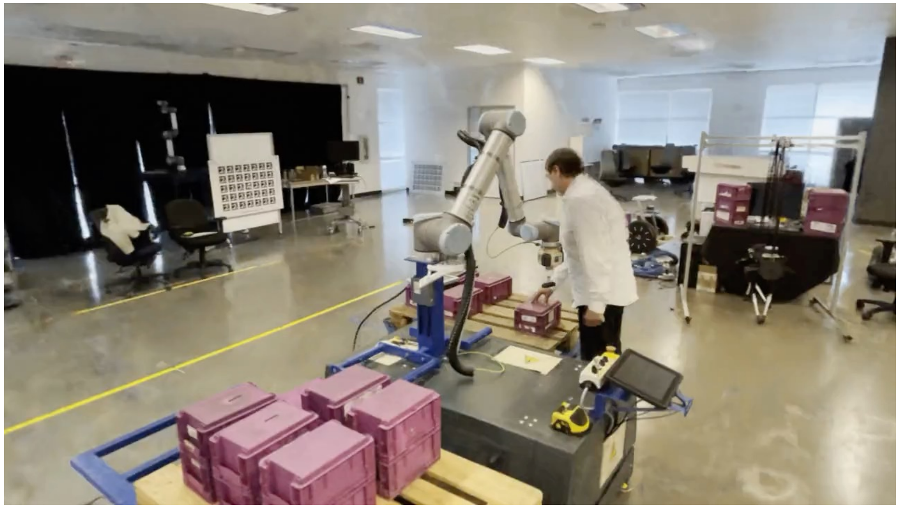


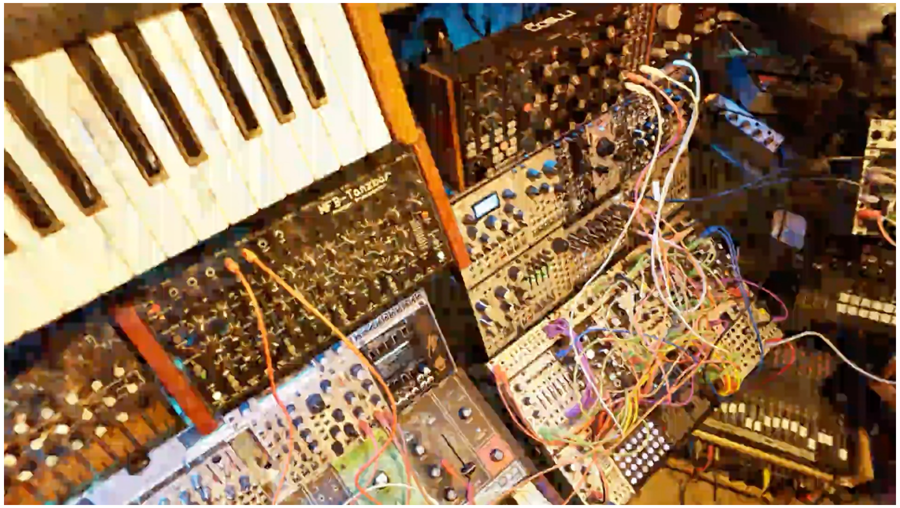
Experiments
Neural Radiance Caching

Experiments
Neural Radiance and Density Fields (NeRF)



Experiments
Neural Radiance and Density Fields (NeRF)





Experiments
Neural Radiance and Density Fields (NeRF)

Comparison with high-quality offline NeRF

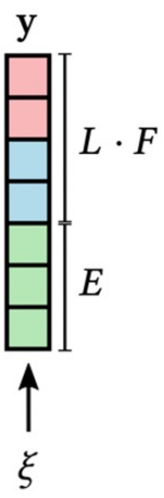
	Mic	Ficus	Chair	Hotdog	Materials	Drums	Ship	Lego	avg.
Ours: Hash (1 s)	26.09	21.30	21.55	21.63	22.07	17.76	20.38	18.83	21.202
Ours: Hash (5 s)	32.60	30.35	30.77	33.42	26.60	23.84	26.38	30.13	29.261
Ours: Hash (15 s)	34.76	32.26	32.95	35.56	28.25	25.23	28.56	33.68	31.407
Ours: Hash (1 min)	35.92	33.05	34.34	36.78	29.33	25.82	30.20	35.63	32.635
Ours: Hash (5 min)	36.22	33.51	35.00	37.40	29.78	26.02	31.10	36.39	33.176
mip-NeRF (~hours)	36.51	33.29	35.14	37.48	30.71	25.48	30.41	35.70	33.090
NSVF (~hours)	34.27	31.23	33.19	37.14	32.68	25.18	27.93	32.29	31.739
NeRF (~hours)	32.91	30.13	33.00	36.18	29.62	25.01	28.65	32.54	31.005
Ours: Frequency (5 min)	31.89	28.74	31.02	34.86	28.93	24.18	28.06	32.77	30.056
Ours: Frequency (1 min)	26.62	24.72	28.51	32.61	26.36	21.33	24.32	28.88	26.669

Experiments
Neural Radiance and Density Fields (NeRF)

Discussion and Future Work

Concatenation vs. Reduction
Concatenation allows for independent, fully parallel processing of each resolution
Reduction of dimensionality of encoded result may be too small to encode useful information

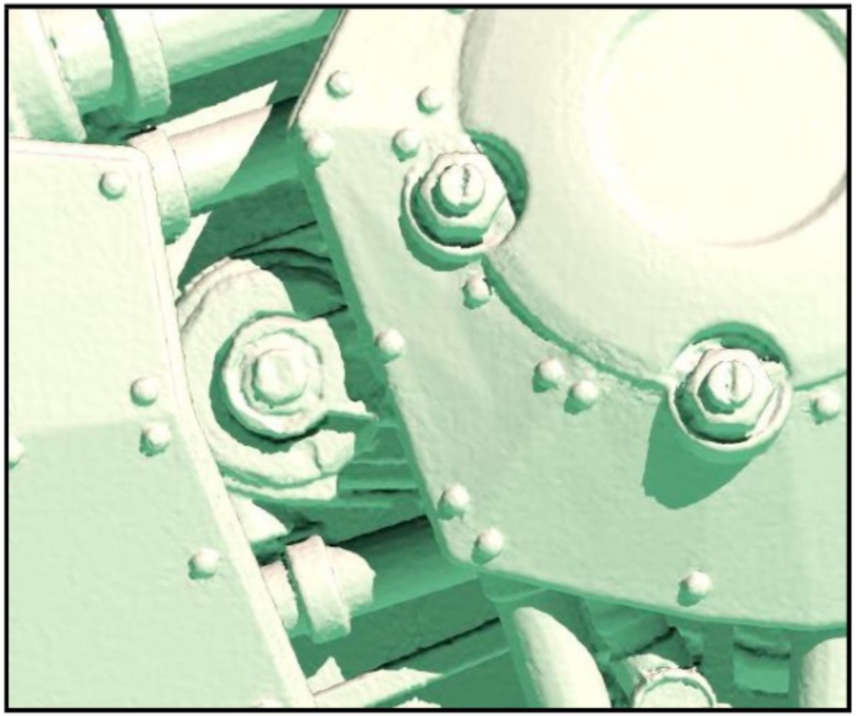
Reduction may be favorable when neural network is significantly more expensive than encoding



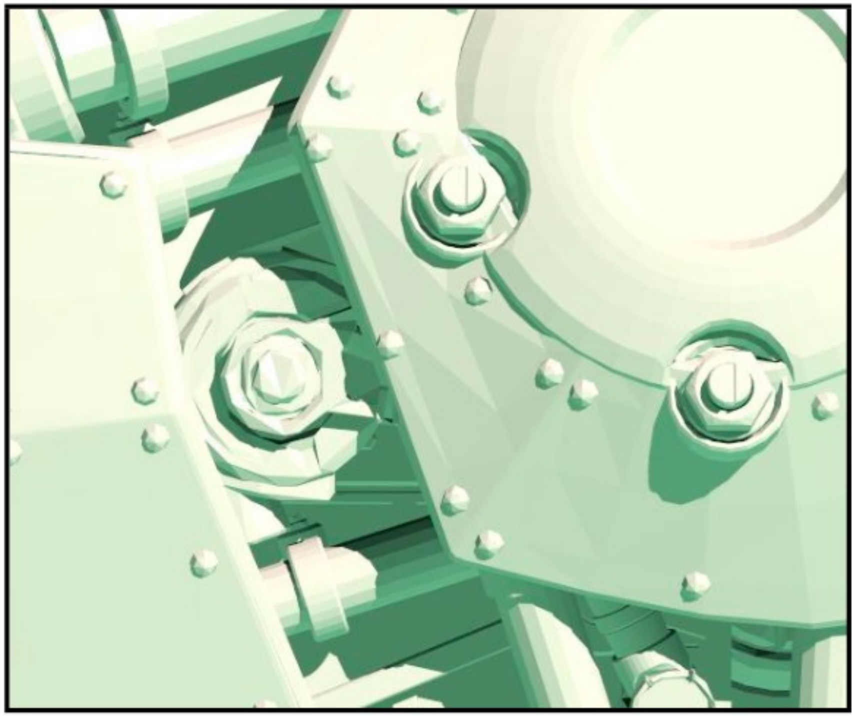
Discussion and Future Work

Microstructure due to hash collisions

Hash encoding



NGLOD



Summary

- Automatically focuses on relevant detail

Summary

- Automatically focuses on relevant detail
- Independent of task

Summary

- Automatically focuses on relevant detail
- Independent of task
- Overhead allows online training and inference

Summary

- Automatically focuses on relevant detail
- Independent of task
- Overhead allows online training and inference
- Speeding up NeRF by several orders of magnitude

Summary

- Automatically focuses on relevant detail
- Independent of task
- Overhead allows online training and inference
- Speeding up NeRF by several orders of magnitude
- Matches performance of concurrent non-neural 3D reconstruction techniques

Summary

- Automatically focuses on relevant detail
- Independent of task
- Overhead allows online training and inference
- Speeding up NeRF by several orders of magnitude
- Matches performance of concurrent non-neural 3D reconstruction techniques
- Single-GPU training times are within reach for many graphics applications

Q&A

Any Questions?

Thank you!

Introduction

Adaptivity

- Coarse Resolution – 1:1 mapping
- Fine Resolution - Hash Table
- No structural Updates to data structure

Introduction

Adaptivity

- Coarse Resolution – 1:1 mapping
- Fine Resolution - Hash Table
- No structural Updates to data structure

Efficiency

- Hash Tabel lookups are O(1)
- Avoiding execution divergence and serial pointer-chasing
- Resolutions may be queried in parallel

Introduction

Adaptivity

- Coarse Resolution – 1:1 mapping
- Fine Resolution - Hash Table
- No structural Updates to data structure

Efficiency

- Hash Tabel lookups are O(1)
- Avoiding execution divergence and serial pointer-chasing
- Resolutions may be queried in parallel

Independent from Task

Multiresolution Hash Encoding

1. Scale Input x
   1. $b := \exp(\frac{\ln N_{max} - \ln N_{min}}{L - 1})$
   2. $N_L := \lfloor N_{min} * b^l \rfloor$
   3. $x * N_l$

2. Round down and up
   1. $\lfloor x_l \rfloor = \lfloor x * N_l \rfloor$
   2. $\lceil x_l \rceil = \lceil x * N_l \rceil$

3. Span voxel with 2^d integer vertices

4. Map each corner to an entry in respective feature vector array

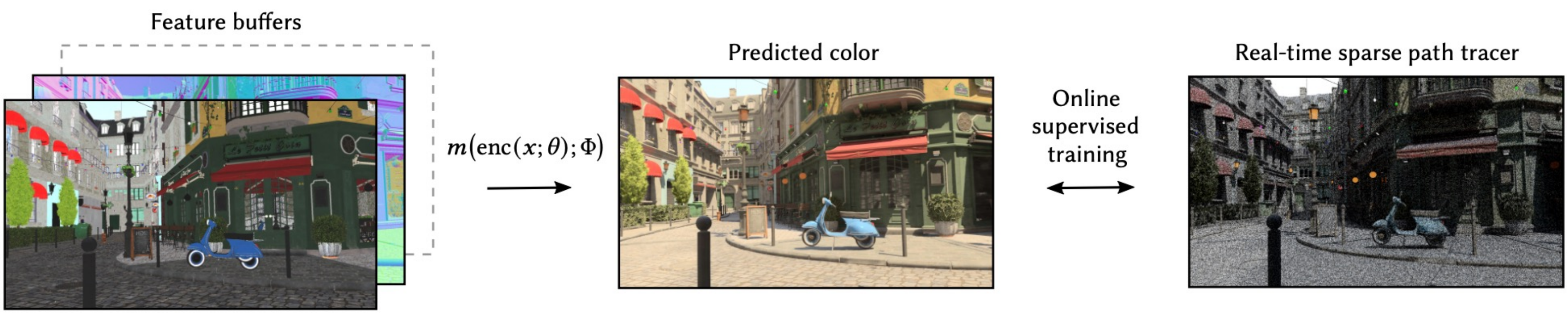
5. Spatial Hash Function
   1. $h(x) = (\bigoplus_{i=1}^{d} x_i \pi_i \ mod\ T)$

Multiresolution Hash Encoding

Number of trainable encoding parameters θ bounded by L*T*F

- L resolution levels
- T feature vectors per level
- F dimensional feature vectors

Experiments
Neural Radiance Caching



Experiments
Neural Radiance and Density Fields (NeRF)

Model Architecture:

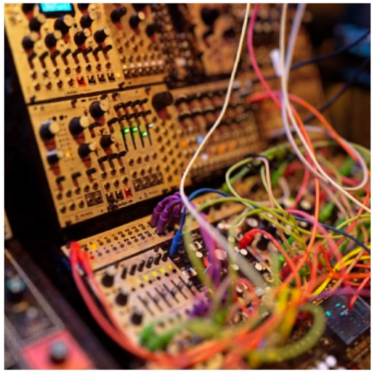
Density MLP: hash encoded position mapped to 16 output values

Color MLP: adds view-dependent color variation

Accelerated ray marching:

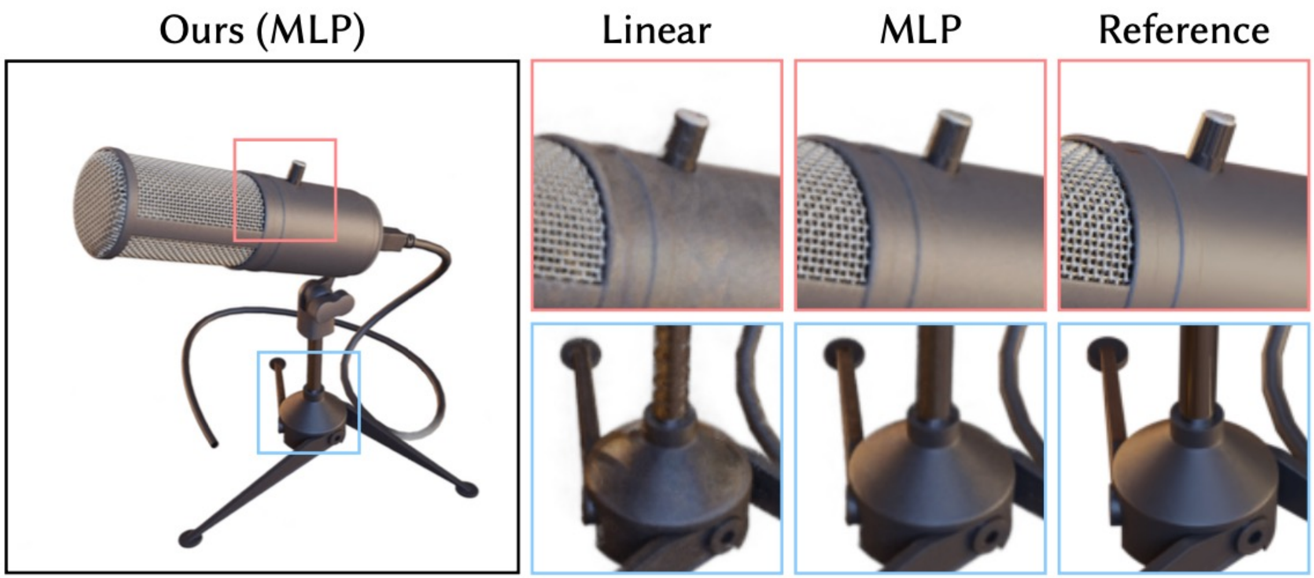
Maintain occupancy grid that coarsely marks empty vs. non-empty space

Additionally cascade it and distribute samples exponentially

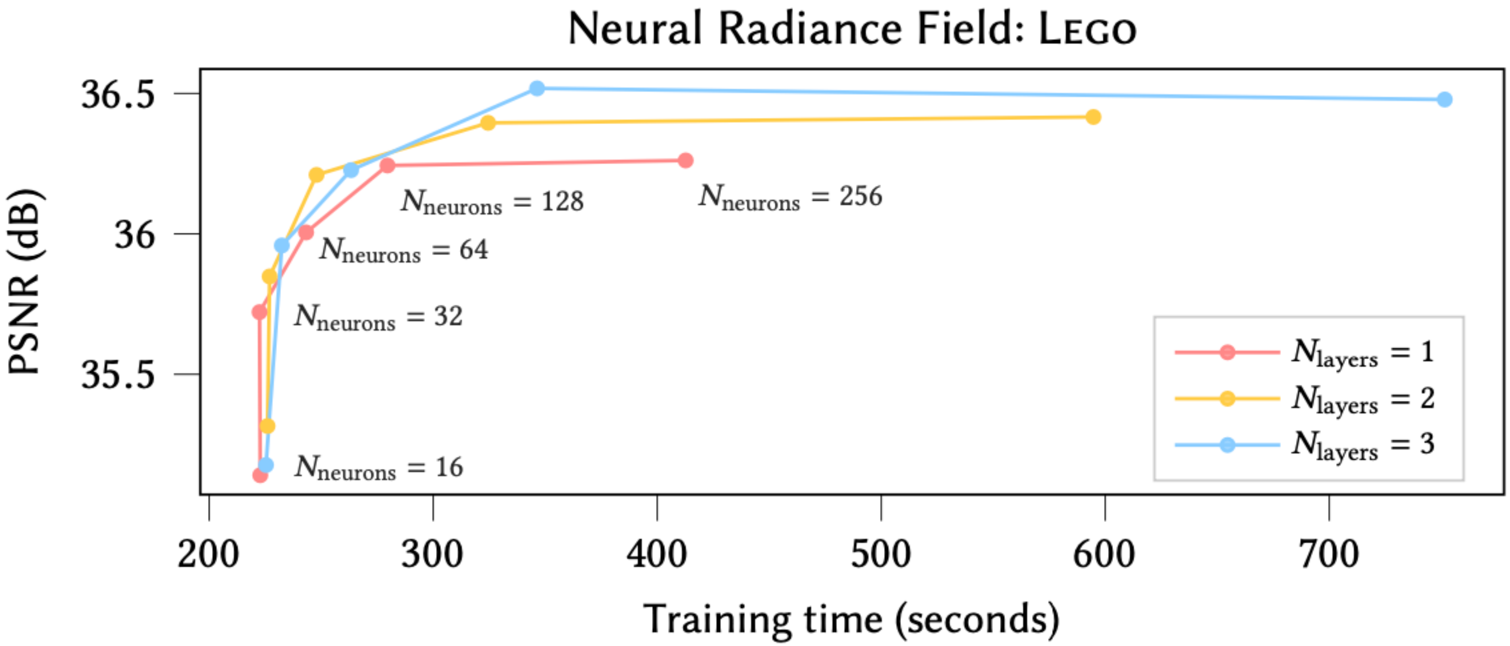


Experiments
Neural Radiance and Density Fields (NeRF)

Comparison with direct voxel lookups



NeRF Model Architecture



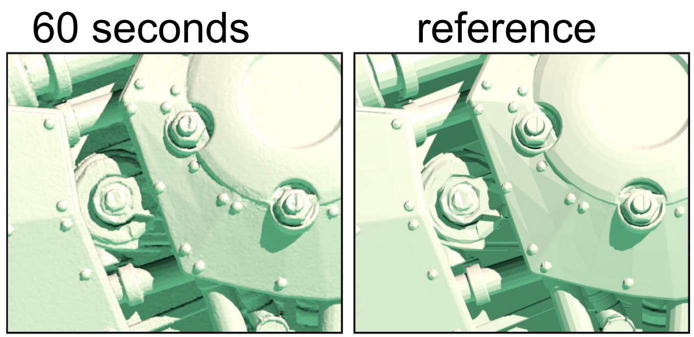
Discussion and Future Work

Choice of hash function

- PCG32 RNG, with superior statistical properties
- Order LSBs of $\mathbb{Z}^d$ by space-filling curve and only hashing higher bits
- Treat hash function as tiling of space into dense grids

Discussion and Future Work

Microstructure due to hash collisions



Other applications

Heterogenous volumetric density fields

Implementation

Performance Considerations

Hash tables evaluated level by level to optimally use GPU‘s caches

Performance on tested hardware constant for $T <= 2^{19}$

Architecture

MLP with two hidden layers with a width of 64 neurons, ReLU activation and linear output layer

N_{max} is set to:

- 2048 x scene size for NeRF and SDF
- Half of gigapixel image width
- 2^{19} for radiance caching

Implementation

Initialization

Weights are initialized according to Glorot and Bengio to provide reasonable scaling of activations and their gradients

Hash table entries initialized using $\mathcal{U}(-10^{-4}, 10^{-4})$ to provide randomnes

Training

Trained by applying Adam with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-15}$

Weak L2 regularization to prevent divergence

Gigapixel and NeRF: L_2 Loss

SDF: MAPE

NRC: luminance-relative L_2 Loss

Learning rate of 10^{-4} for SDF and 10^{-2} otherwise

Implementation

Non-spatial input dimensions

Auxiliary dimensions such as view direction and material parameters (light field)

One-blob encoding [Müller et al. 2019] is used in radiance caching

Spherical Harminocs basis in NeRF