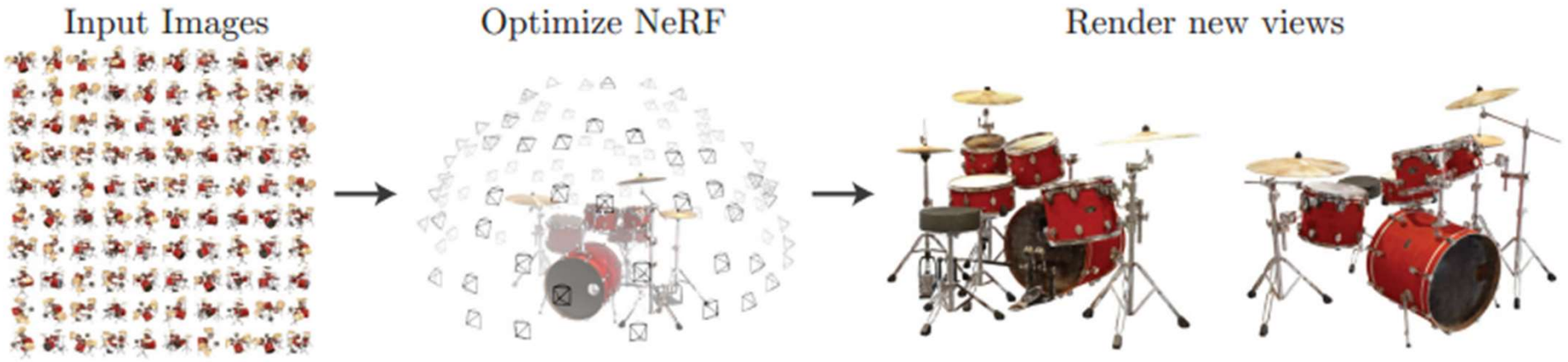
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020 Oral - Best Paper Honorable Mention

Ben Mildenhall[1⋆] Pratul P. Srinivasan[1⋆] Matthew Tancik[1⋆]
Jonathan T. Barron[2] Ravi Ramamoorthi[3] Ren Ng[1]

[1]UC Berkeley [2]Google Research [3]UC San Diego

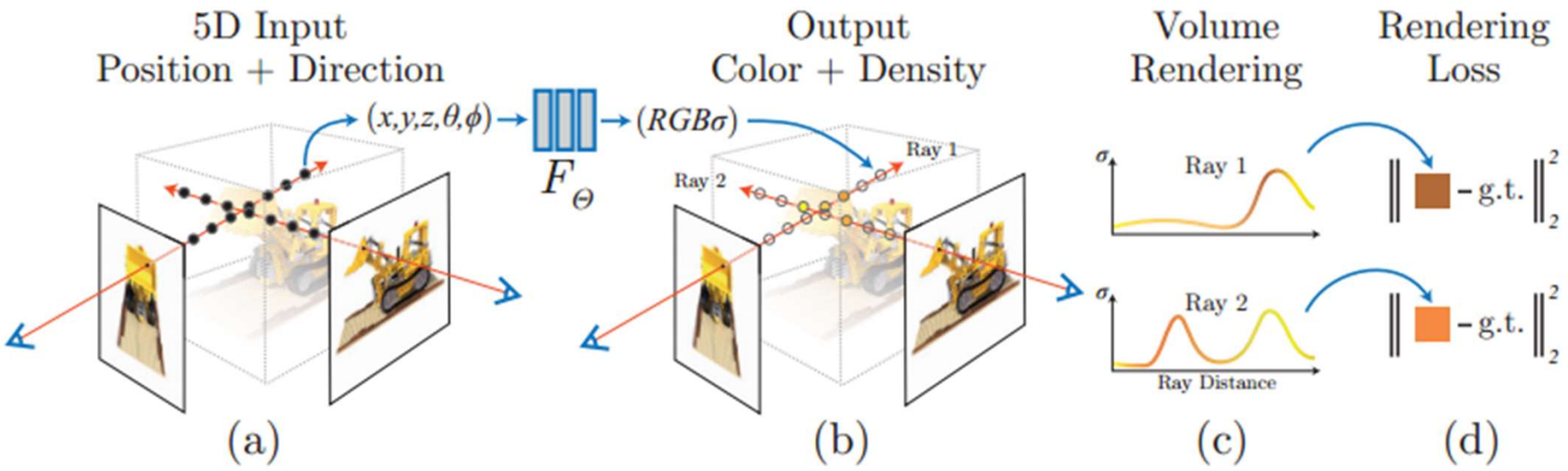
Introduction



A function:**5D coordinate (x, y, z, θ, φ)** to **a single volume density σ & view-dependent RGB color**

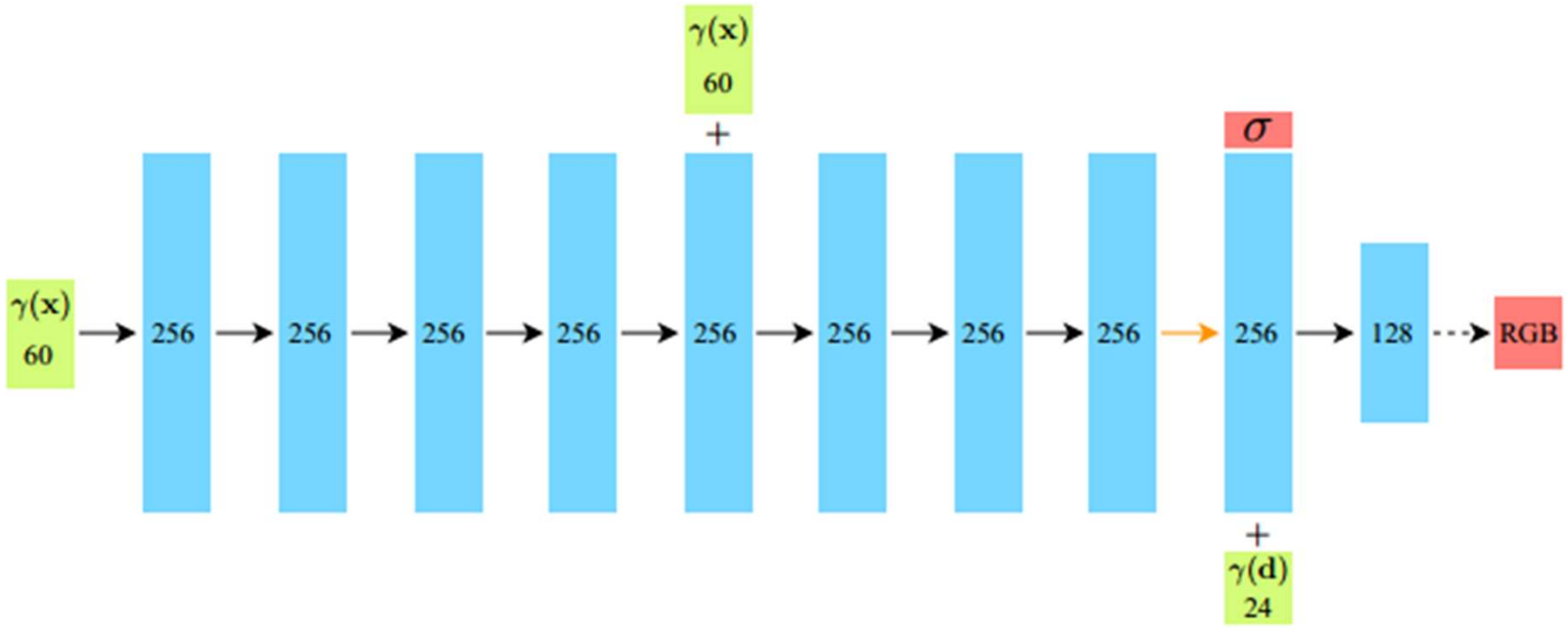
Represented by a **MLP**:a deep fully-connected neural network without any convolutional layers

Introduction

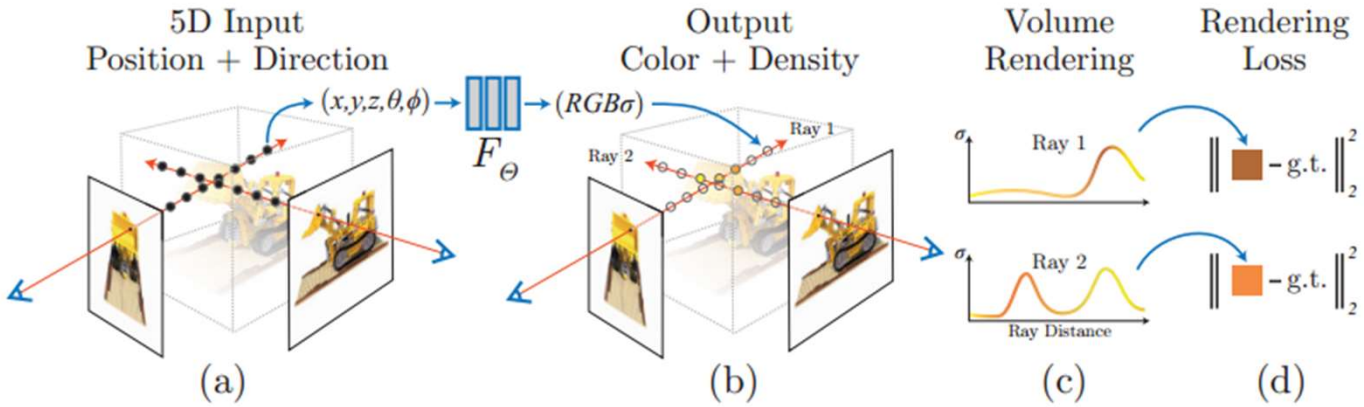


1) march camera rays through the scene to generate a sampled set of 3D points
2) use those points and their corresponding2D viewing directions as input to the neural network to produce an output set of colors and densities
3) use classical volume rendering techniques to accumulate those colors and densities into a 2D image.

Neural Radiance Field Scene Representation



Volume Rendering with Radiance Fields



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

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

Optimizing a Neural Radiance Field:
Positional encoding

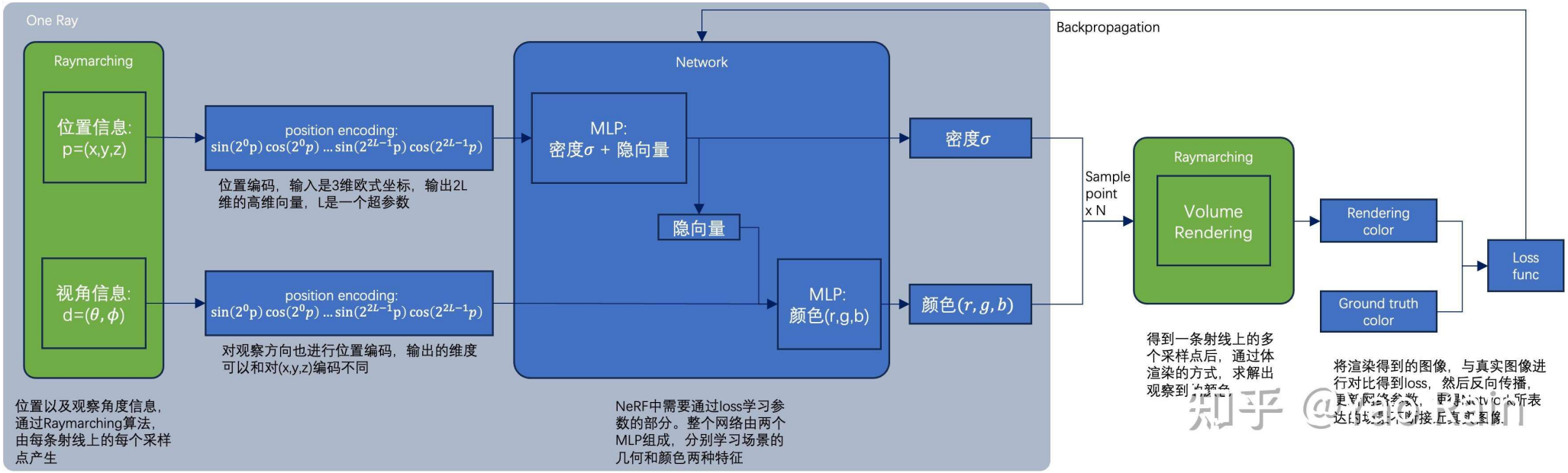
Ground Truth

No Positional Encoding

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

L = 10 for γ(x) and L = 4 for γ(d)

Implementation details



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

Batch size:4096 N_c=64 N_f=128

The optimization for a single scene typically take around 100-300k iterations to converge on a single NVIDIA V100 GPU (about 1-2 days)

Results

Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

Neural Volumes (NV)
Scene Representation Networks (SRN)
Local Light Field Fusion (LLFF)

PSNR:Peak Signal-to-Noise Ratio 峰值信噪比
SSIM:Structure Similarity Index Measure 结构相似性
LPIPS:Learned Perceptual Image Patch Similarity 学习感知图像块相似度

Comparations

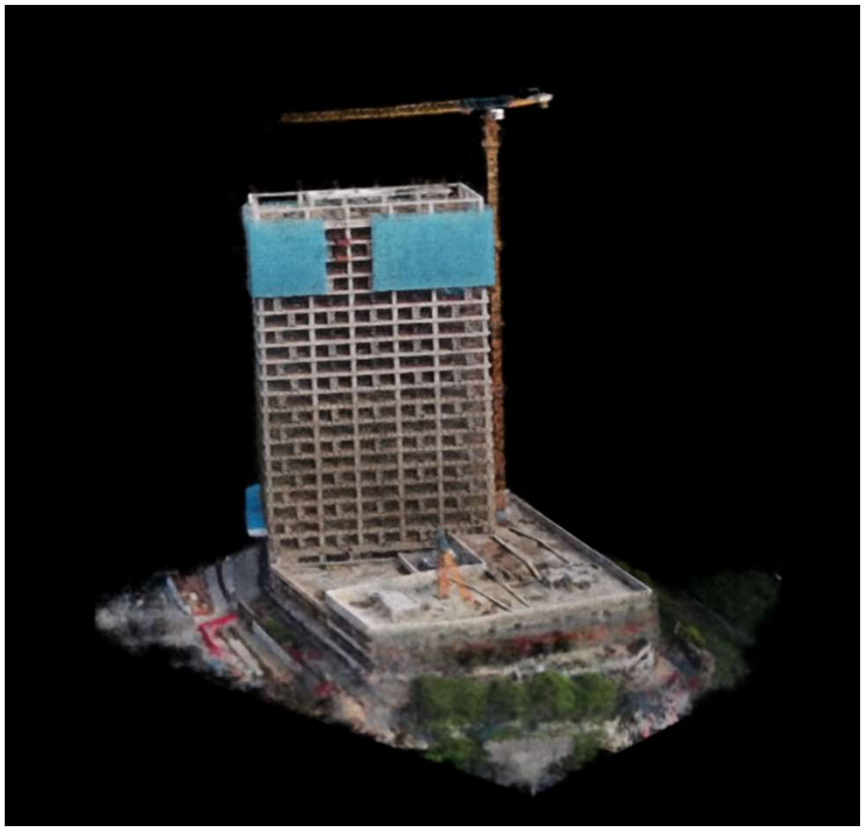
Ablation studies

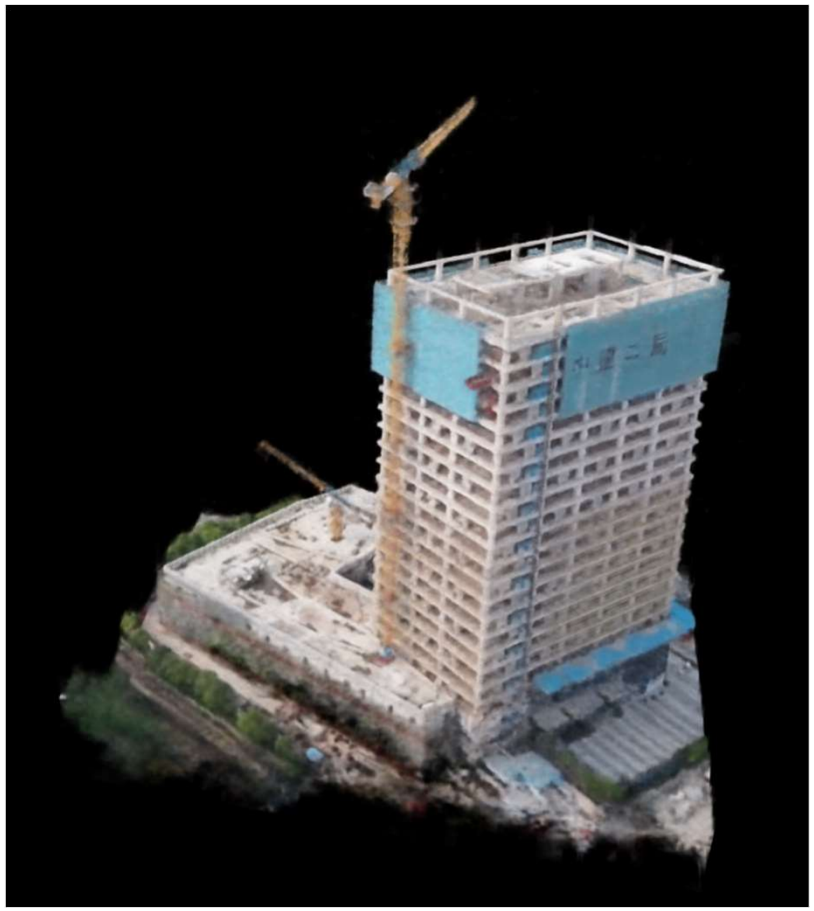
	Input	#Im.	L	(N_c, N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

Table 2: An ablation study of our model. Metrics are averaged over the 8 scenes from our realistic synthetic dataset. See Sec. 6.4 for detailed descriptions.

Instant NGP+NeRF

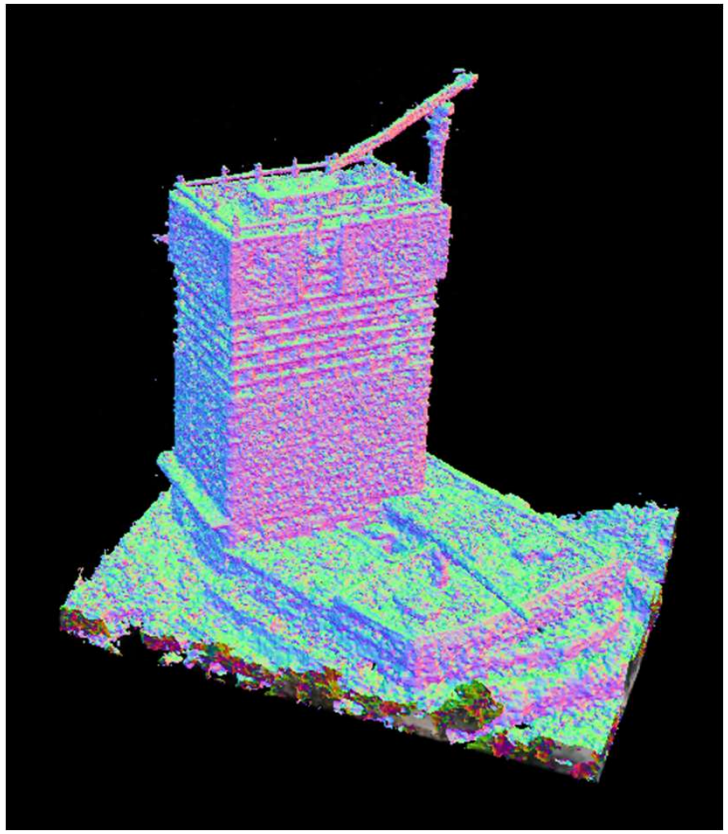
Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

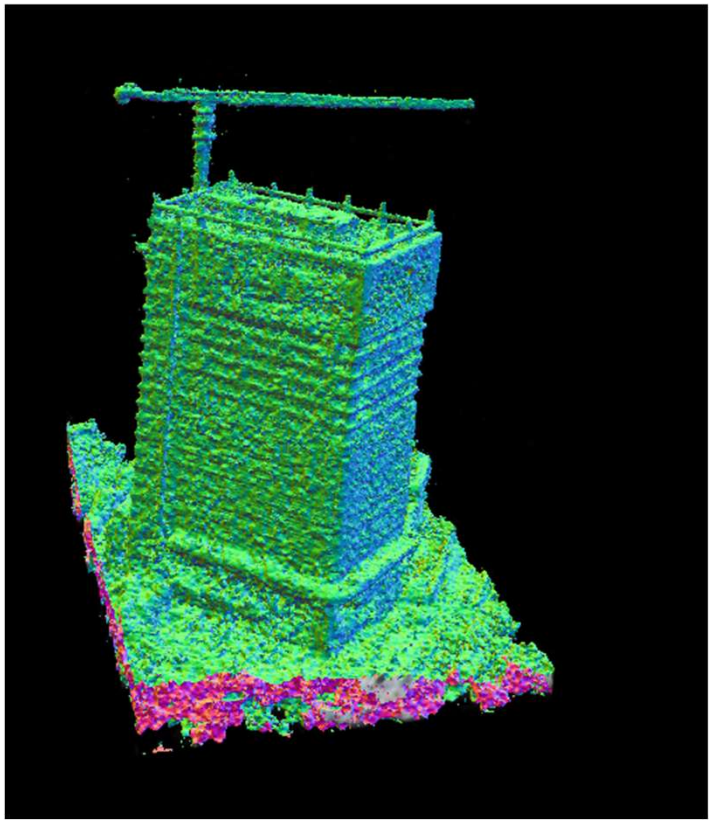




Instant NGP+NeRF

Extract mesh





Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

SIGGRAPH 2022

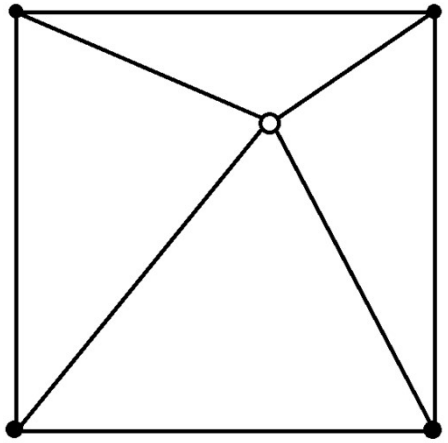
THOMAS MÜLLER, NVIDIA, Switzerland
ALEX EVANS, NVIDIA, United Kingdom
CHRISTOPH SCHIED, NVIDIA, USA
ALEXANDER KELLER, NVIDIA, Germany

Position encoding

- Frequency encoding
- Parametric encodings
- Sparse parametric encodings

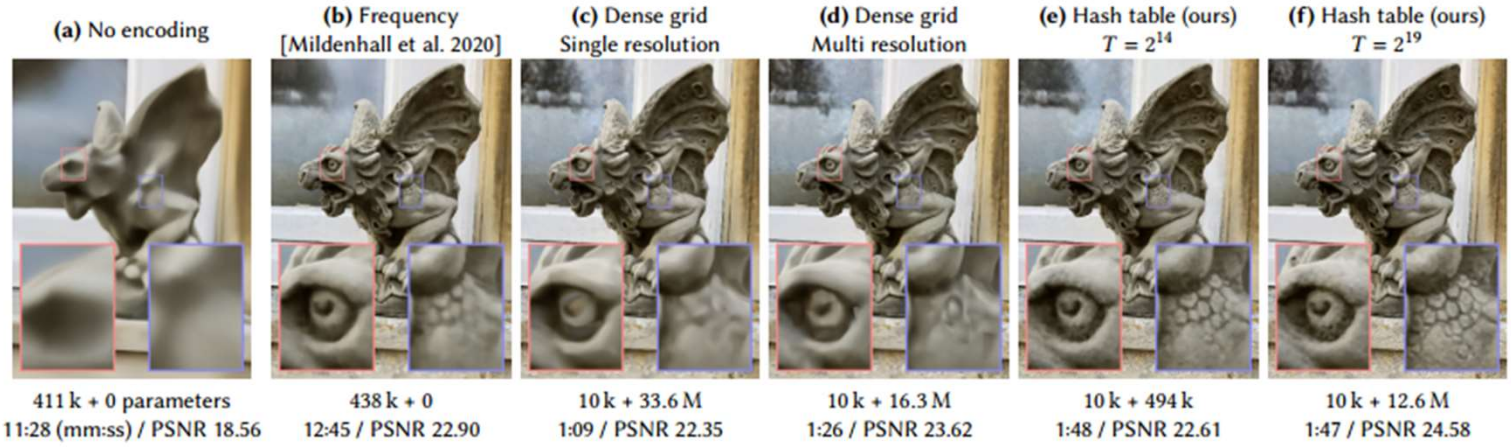
parametric encodings

- arrange additional trainable parameters (beyond weights and biases) in an auxiliary data structure, such as a grid or a tree interpolate these parameters depending on the input vector x
- trades a larger memory footprint for a smaller computational cost
- every weight in the fully connected MLP network must be updated



Sparse parametric encodings

- In parametric encodings, the number of parameters grows as O(N^3), while the visible surface of interest has surface area that grows only as O(N^2). In this example, the grid has resolution 128^3, but only 53 807(2.57%) of its cells touch the visible surface.
- multi-resolution decomposition:features are stored in eight co-located grids with resolutions from 163 to 1733, each containing 2-dimensional feature vectors. These are concatenated to form a 16-dimensional input to the network. Despite having less than half the numberof parameters , the reconstruction quality is similar



Instant-NGP

$$h(\mathbf{x}) = \left(\bigoplus_{i=1}^{d} x_i \pi_i \right) \mod T \qquad \mathbf{w}_l := \mathbf{x}_l - \lfloor \mathbf{x}_l \rfloor$$

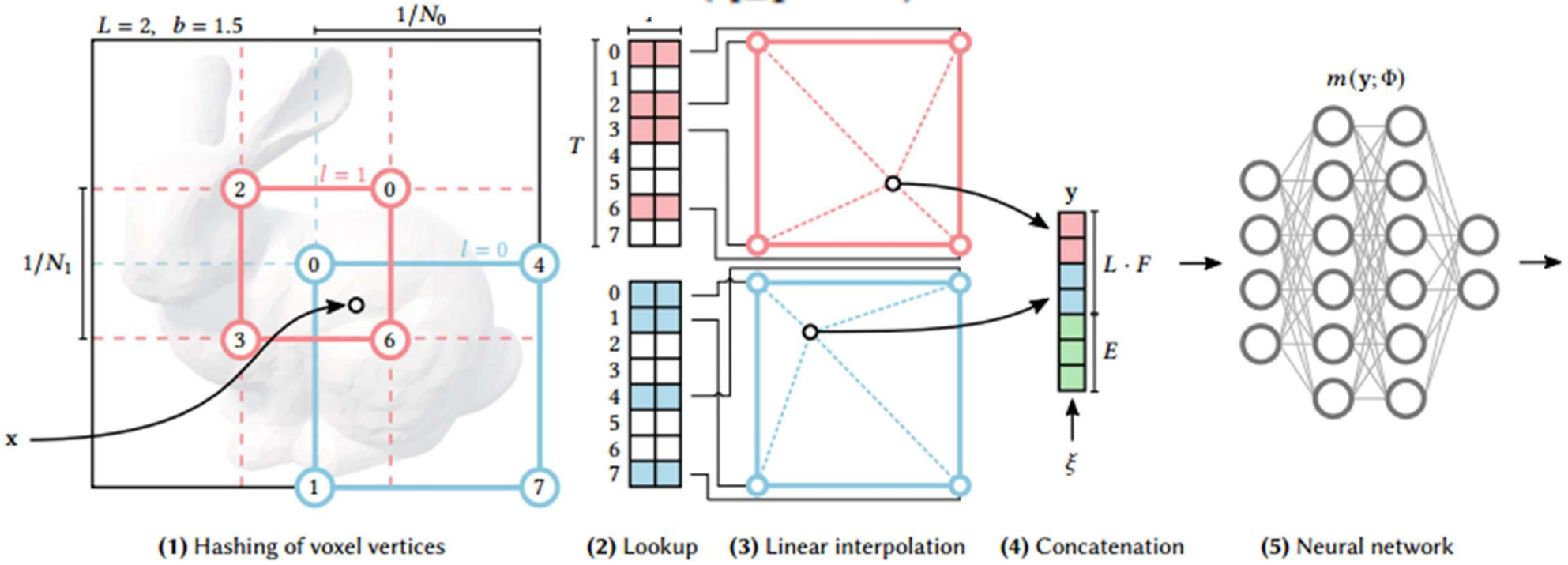


Fig. 3. Illustration of the multiresolution hash encoding in 2D. **(1)** for a given input coordinate $\mathbf{x}$, we find the surrounding voxels at L resolution levels and assign indices to their corners by hashing their integer coordinates. **(2)** for all resulting corner indices, we look up the corresponding F-dimensional feature vectors from the hash tables θ_l and **(3)** linearly interpolate them according to the relative position of $\mathbf{x}$ within the respective l-th voxel. **(4)** we concatenate the result of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$, producing the encoded MLP input $y \in \mathbb{R}^{LF+E}$, which **(5)** is evaluated last. To train the encoding, loss gradients are backpropagated through the MLP **(5)**, the concatenation **(4)**, the linear interpolation **(3)**, and then accumulated in the looked-up feature vectors.

Hyper parameters

Table 1. Hash encoding parameters and their ranges in our results. Only the hash table size T and max. resolution $N_{\max}$ need to be tuned to the task.

Parameter	Symbol	Value
Number of levels	L	16
Max. entries per level (hash table size)	T	2^{14} to 2^{24}
Number of feature dimensions per entry	F	2
Coarsest resolution	$N_{\min}$	16
Finest resolution	$N_{\max}$	512 to 524288

$$N_l := \left\lfloor N_{\min} \cdot b^l \right\rfloor,$$

$$b := \exp\left(\frac{\ln N_{\max} - \ln N_{\min}}{L-1}\right).$$

Different L and F

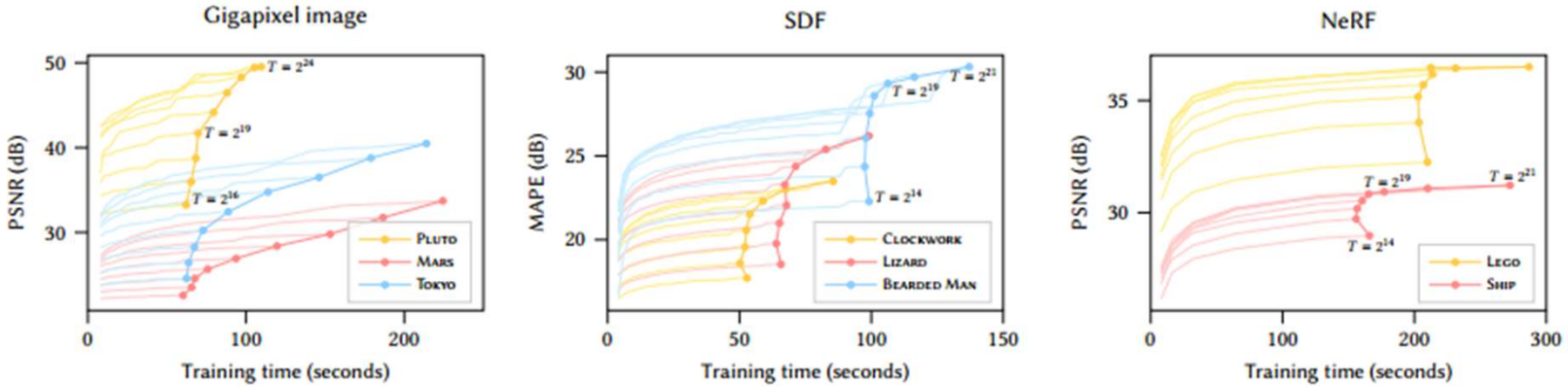
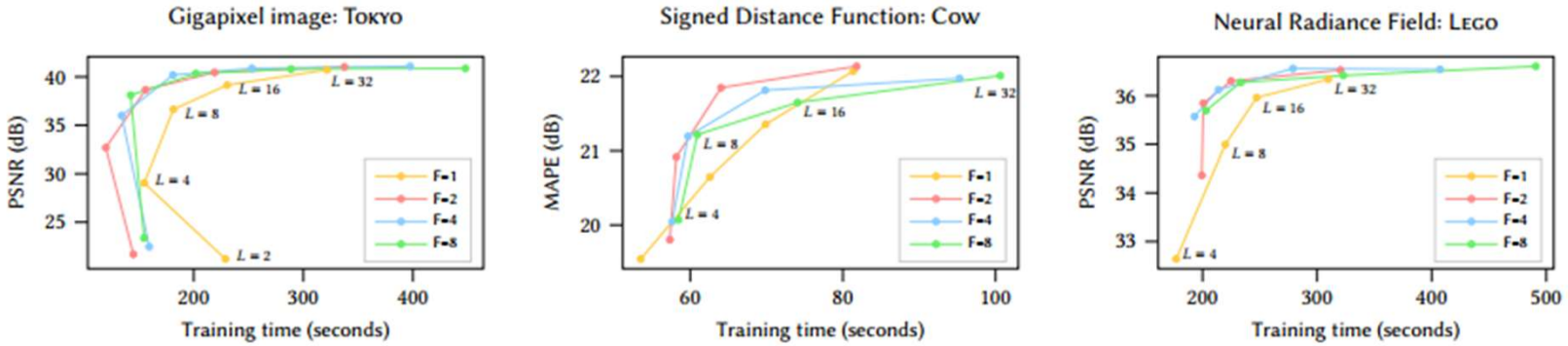


Fig. 4. The main curves plot test error over training time for varying hash table size T which determines the number of trainable encoding parameters. Increasing T improves reconstruction, at the cost of higher memory usage and slower training and inference. A performance cliff is visible at $T > 2^{19}$ where the cache of our RTX 3090 GPU becomes oversubscribed (particularly visible for SDF and NeRF). The plot also shows model convergence over time leading up to the final state. This highlights how high quality results are already obtained after only a few seconds. Jumps in the convergence (most visible towards the end of SDF training) are caused by learning rate decay. For NeRF and Gigapixel image, training finishes after 31 000 steps and for SDF after 11 000 steps.



Different hash table size （T）

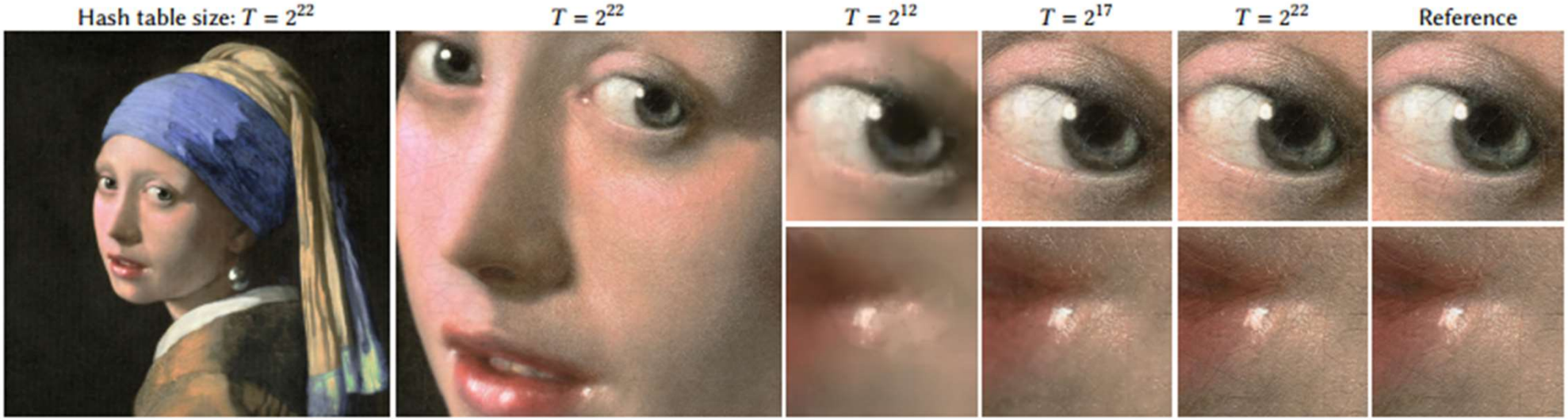
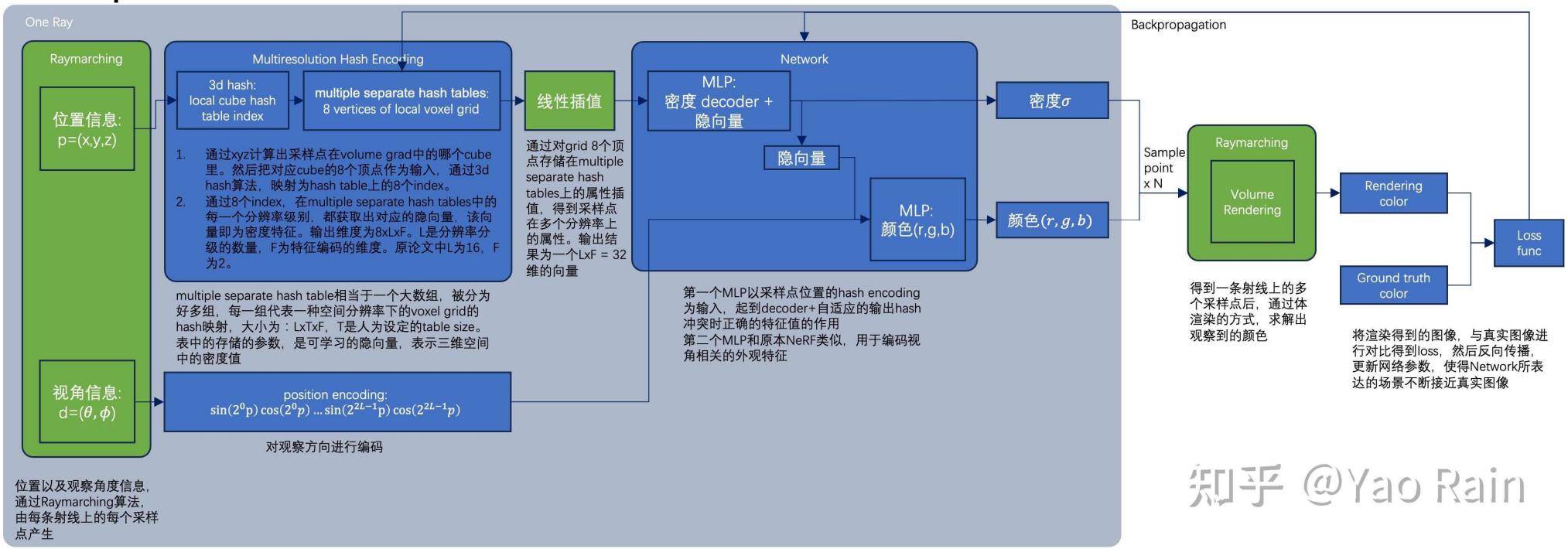
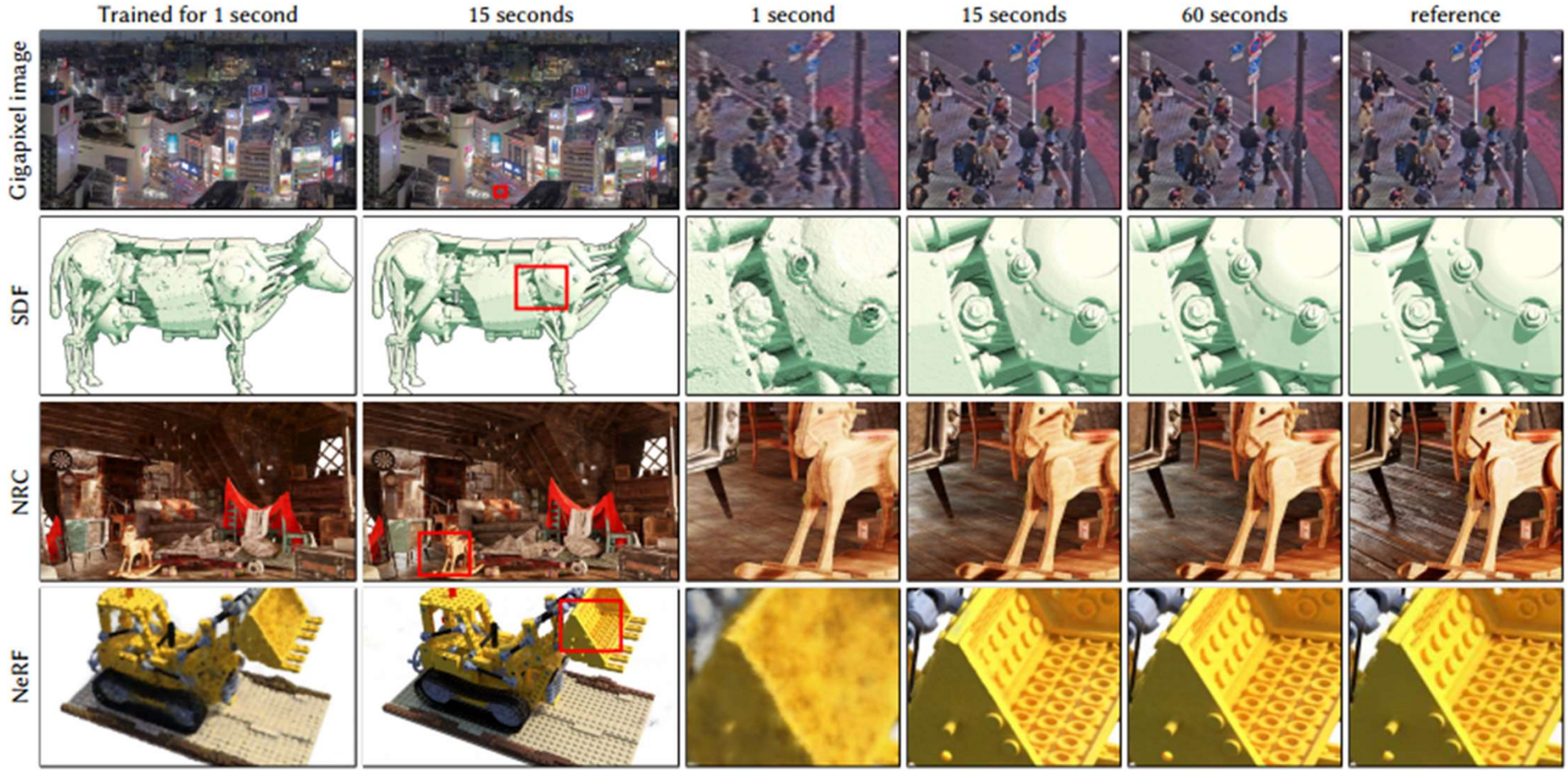


Fig. 6. Approximating an RGB image of resolution 20 000 × 23 466 (469 M RGB pixels) with our multiresolution hash encoding. With hash table sizes T of 2^{12}, 2^{17}, and 2^{22} the models shown have 117 k, 2.7 M, and 47.5 M trainable parameters respectively. With only 3.4% of the degrees of freedom of the input, the last model achieves a reconstruction PSNR of 29.8 dB. "Girl With a Pearl Earring" renovation ©Koorosh Orooj (CC BY-SA 4.0)

Pipeline



Comparations



Comparations

Table 2. Peak signal to noise ratio (PSNR) of our NeRF implementation with multiresolution hash encoding ("Ours: Hash") vs. NeRF [Mildenhall et al. 2020], mip-NeRF [Barron et al. 2021a], and NSVF [Liu et al. 2020], which require ~hours to train (values taken from the respective papers). To demonstrate the comparatively rapid training of our method, we list its results after training for 1 s to 5 min. For each scene, we mark the methods with least error using gold ●, silver ●, and bronze ● medals. To analyze the degree to which our speedup originates from our optimized implementation vs. from our hash encoding, we also report PSNR for a nearly identical version of our implementation, in which the hash encoding has been replaced by the frequency encoding and the MLP correspondingly enlarged to match Mildenhall et al. [2020] ("Ours: Frequency"; details in Appendix D). It approaches NeRF's quality after training for just ~5 min, yet is outperformed by our full method after training for 5 s–15 s, amounting to a 20–60× improvement that can be attributed to the hash encoding.

	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