Neural Radiance Fields

24.11.31.

Mingyu Kim

24.11.01

Repository

https://github.com/MingyuKim87/day6.git

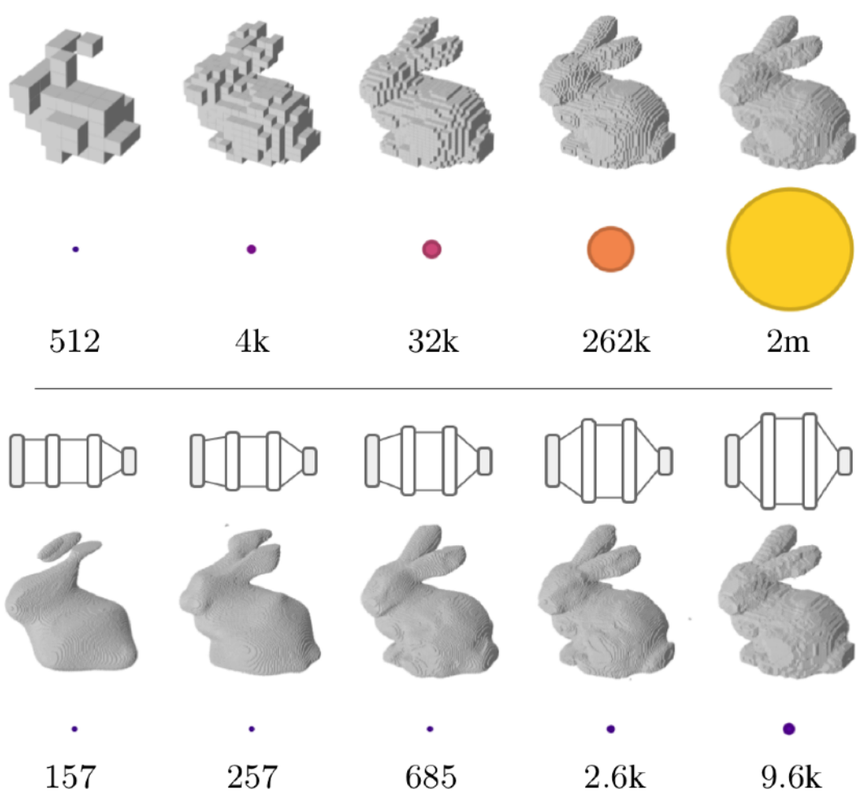
Dataset :

https://drive.google.com/drive/folders/1xcoBIj4ZyGbeDzVXRnqYr9hWupfdImI8?usp=sharing

Contents

Coordinate Networks (Implicit Neural Representation)

- INRs aims to make continuous fields using given specific signals.
  - Do not require additional storages (only store a checkpoint file, network parameters)
    - The previous methods require huge storages as increasing resolutions.
  - Can synthesize unseen signals such as super-resolution, interpolating frames.



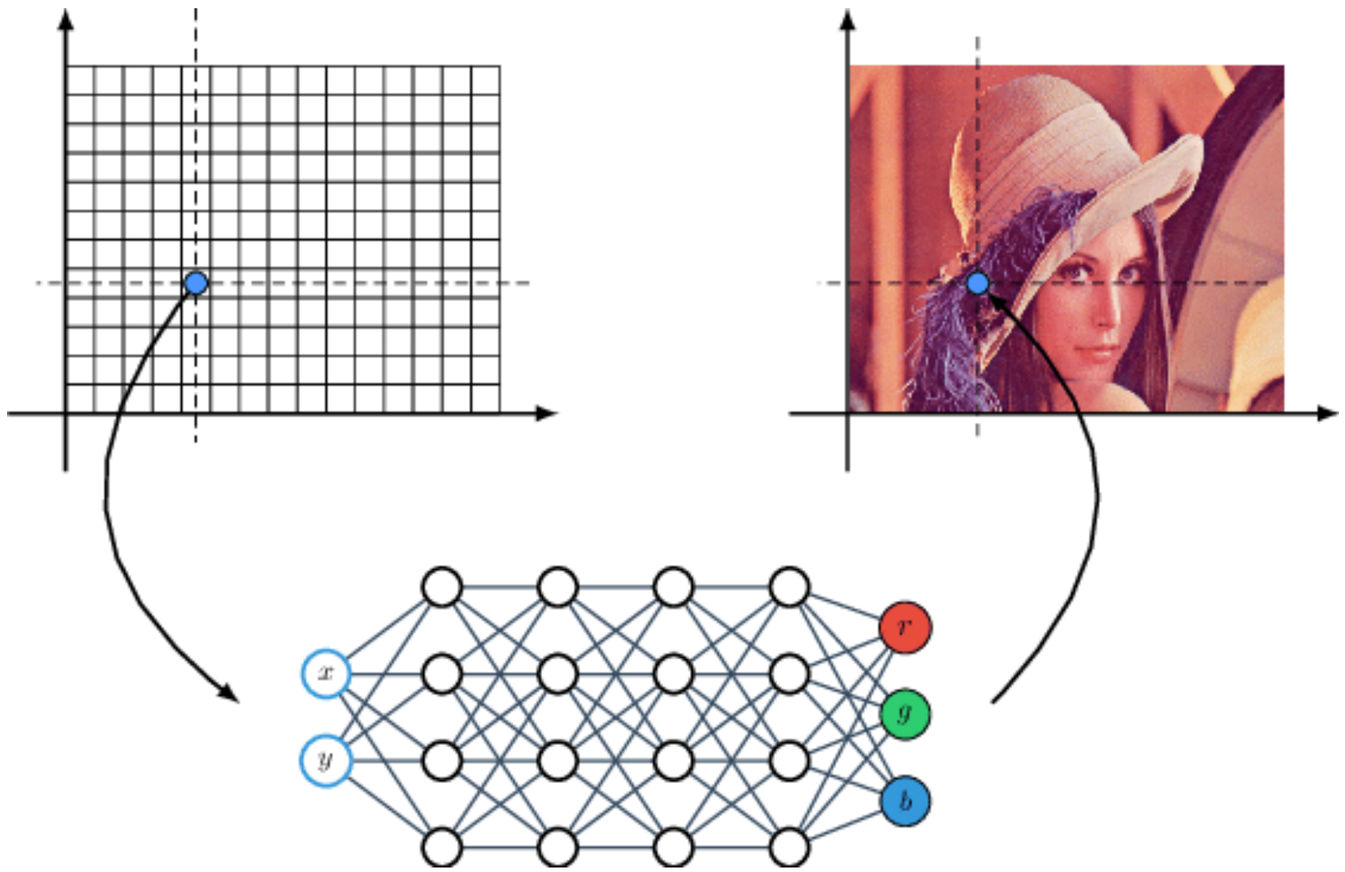
RESOLUTION	DEPTH	WIDTH	ω_0	n_{PARAMS}
8^3	4	6	8	157
16^3	4	8	12	257
32^3	5	12	12	685
64^3	7	20	14	2621
128^3	10	32	50	9665

Table 5. Hyperparameter values used for Figure 2.

Dupont, Emilien, et al. "From data to functa: Your data point is a function and you can treat it like one." *International Conference on Machine Learning*. PMLR, 2022.

Coordinate Networks (Implicit Neural Representation)

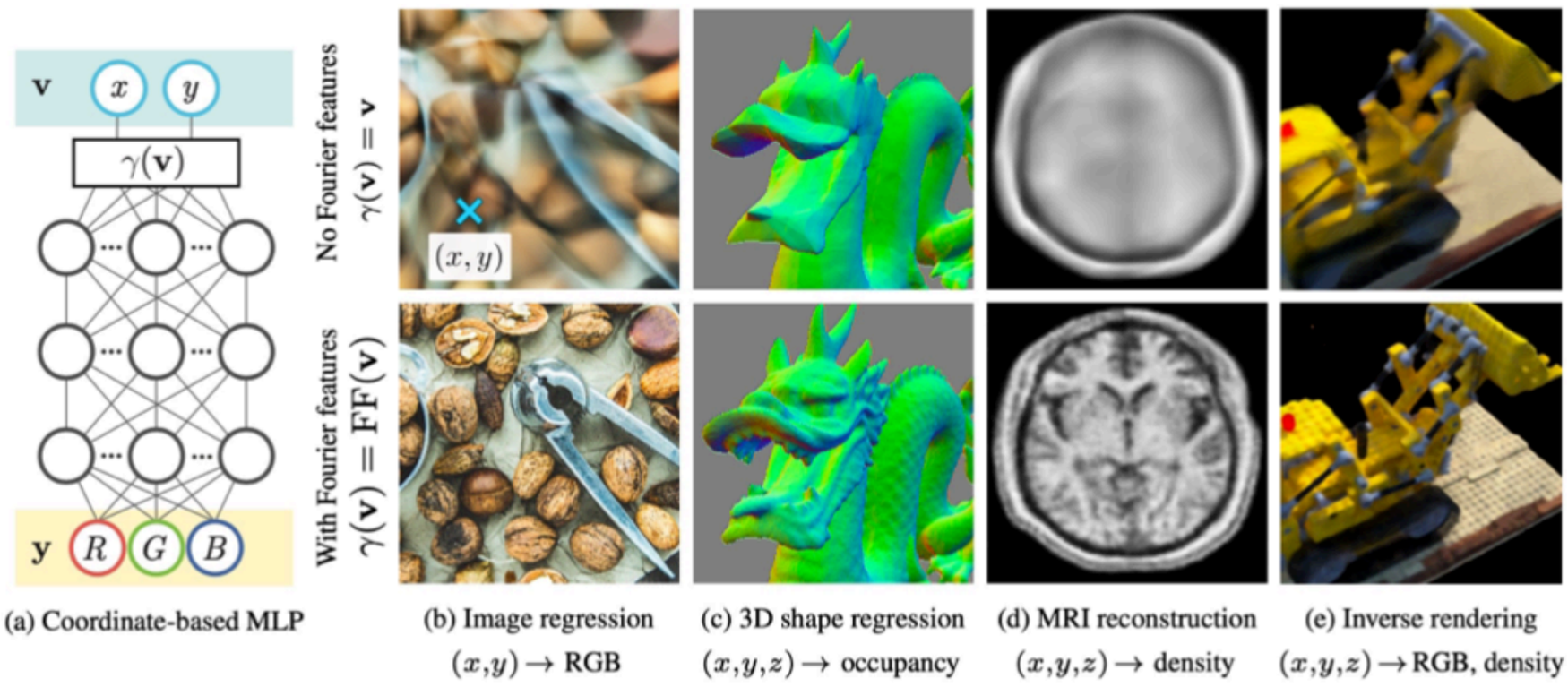
- Generating high-resolutional data is not a big hurdle because all computation is conducted by a feed-forward operation, whereas it requires a large batch size.
  - A batch size will be determined as the number of pixels (Image) or the number of pixels × frames.



Dupont, Emilien, et al. "From data to functa: Your data point is a function and you can treat it like one." *International Conference on Machine Learning*. PMLR, 2022.

FFN : Fourier Feature Networks (NeurIPS2020)

- FFN : Random Fourier feature mapping before neural network is sufficient
  - Rather than applying since activation to all layers
  - Feature matrix is not trained (deterministic feature mapping)



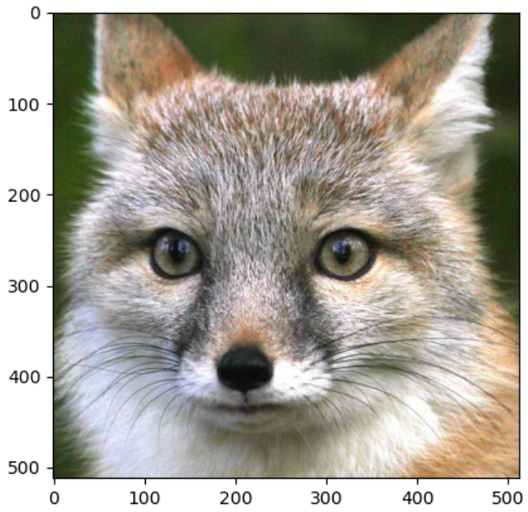
(a) Coordinate-based MLP | (b) Image regression $(x,y) \rightarrow$ RGB | (c) 3D shape regression $(x,y,z) \rightarrow$ occupancy | (d) MRI reconstruction $(x,y,z) \rightarrow$ density | (e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

$$\gamma(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]^{\mathrm{T}}, \quad \mathbf{B}_{ij} \sim \mathcal{N}(0, \sigma^2)$$

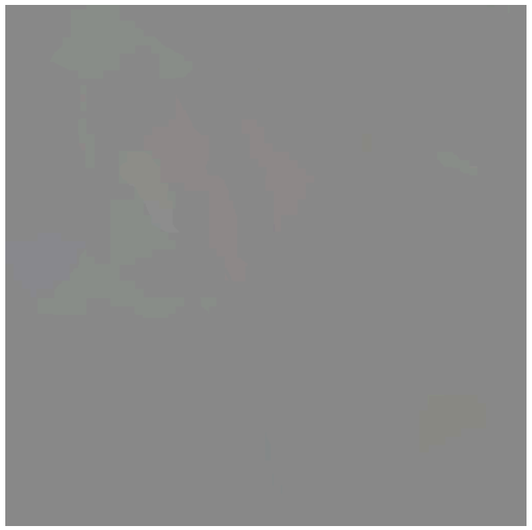
Tancik, Matthew, et al. "Fourier features let networks learn high frequency functions in low dimensional domains." *Advances in Neural Information Processing Systems* 33 (2020): 7537-7547.

FFN : Fourier Feature Networks (CVPR2020)

- FFN : Random Fourier feature mapping before neural network is sufficient
  - $\gamma(v) = \{\sin(2\pi\alpha\beta x), \cos(2\pi\alpha\beta x)\} \quad where \quad \beta \sim \mathcal{N}(0,1) \quad |\beta| = 256$

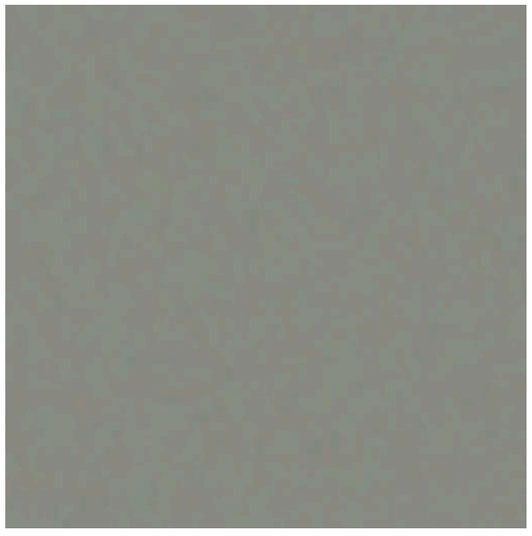


No mapping

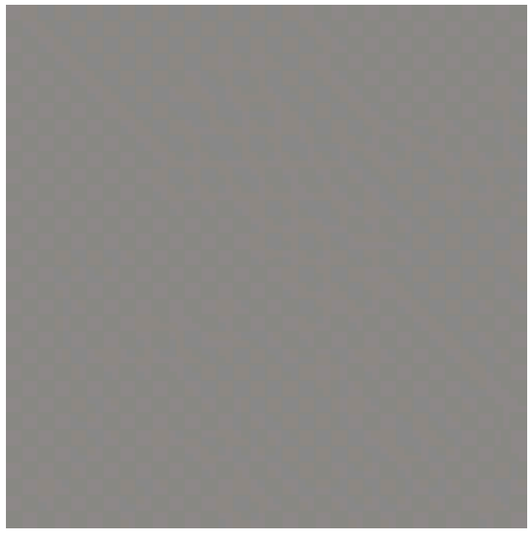


$\gamma(v) = \{\sin(x), \cos(x)\}$

$\alpha = 1$



$\alpha = 10$

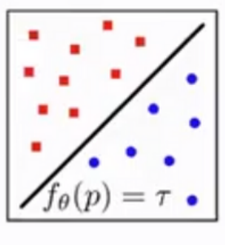


$\alpha = 100$

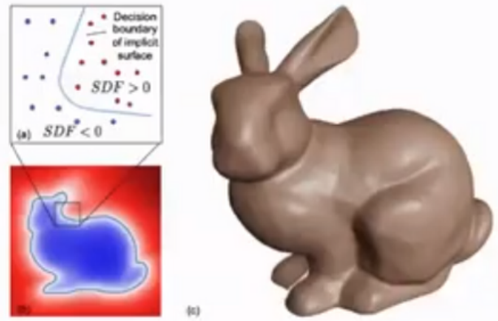
3D Shape Representation via INRs

- Efficient representation for 3D shape
  - Training : Sparse 3D signals (points and its corresponding target values)
  - Testing : New coordinates (x, y, z) → 3D shape information

Occupancy Networks
(Mescheder et al. 2019)
$(x, y, z) \rightarrow$ occupancy

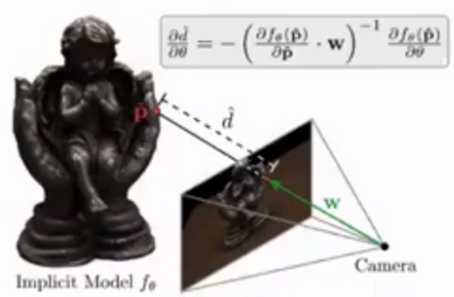


DeepSDF
(Park et al. 2019)
$(x, y, z) \rightarrow$ distance



Scene Representation Networks
(Sitzmann et al. 2019)
$(x, y, z) \rightarrow$ latent vec. (color, dist.)

Differentiable Volumetric Rendering
(Niemeyer et al. 2020)
$(x, y, z) \rightarrow$ color, occ.



Matthew Tancik: Neural Radiance Fields for View Synthesis: https://youtu.be/dPWLybp4LL0

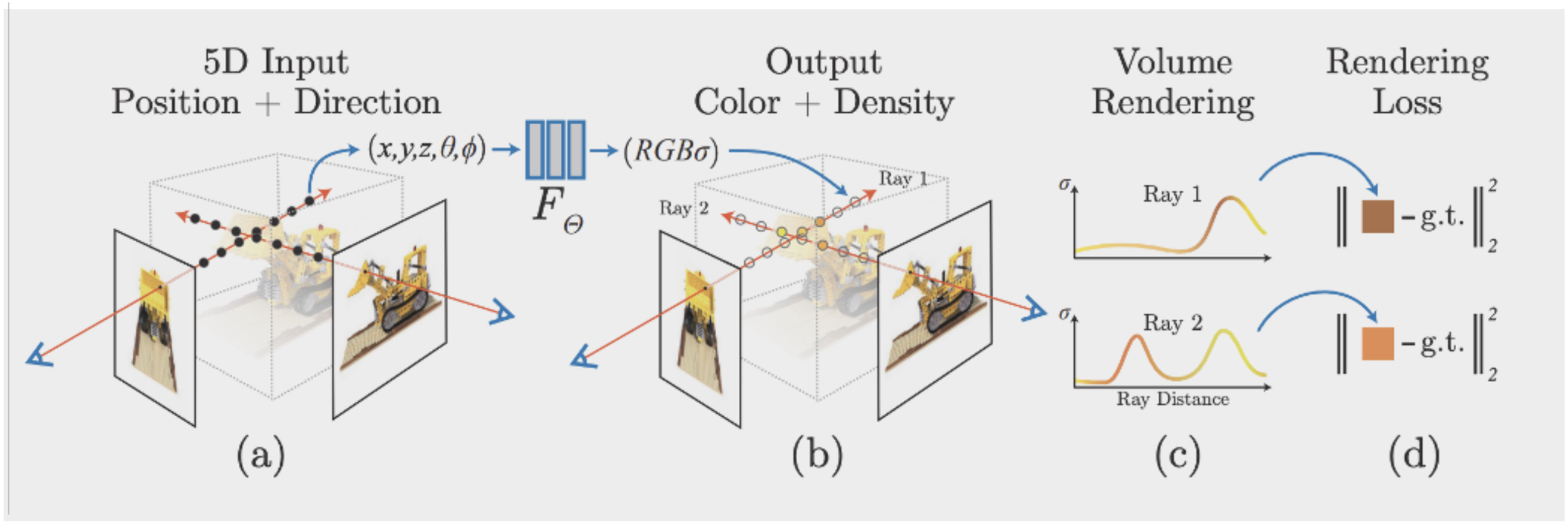
Neural Radiance Fields (NeRFs)

- Introduction
  - Neural Radiance fields utilize 3D coordinate networks via **Volume Rendering**

https://youtu.be/JuH79E8rdKc

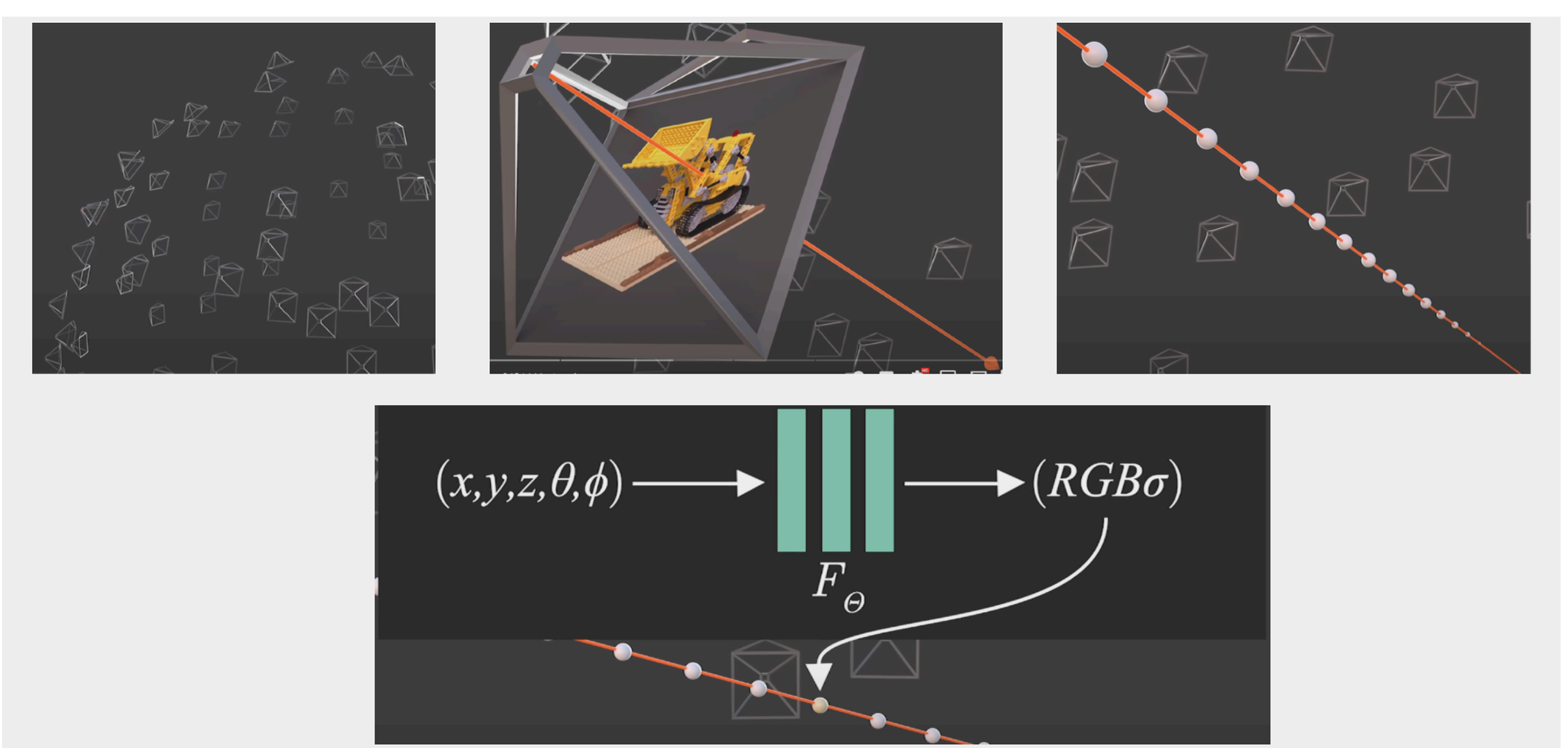
Neural Radiance Fields (NeRFs)

- Neural Radiance Fields employs coordinate networks
  - (a) : Cast (discretize) samples in the 5D coordinates (position, direction) along camera rays
  - (b) : Feeding those points into an MLP to produce a color (c) and volume density (σ),
  - (c) : Using volume rendering techniques to composite these values into an image.
  - (d) : This rendering function is differentiable, so we can optimize our scene representation by minimizing synthesized pixels and ground truth pixels.



Neural Radiance Fields (NeRFs)

- Visual description of Volume Rendering
  - We render the color of any ray passing through the scene using classical volume rendering.
  - The volume density σ(x) can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location x.

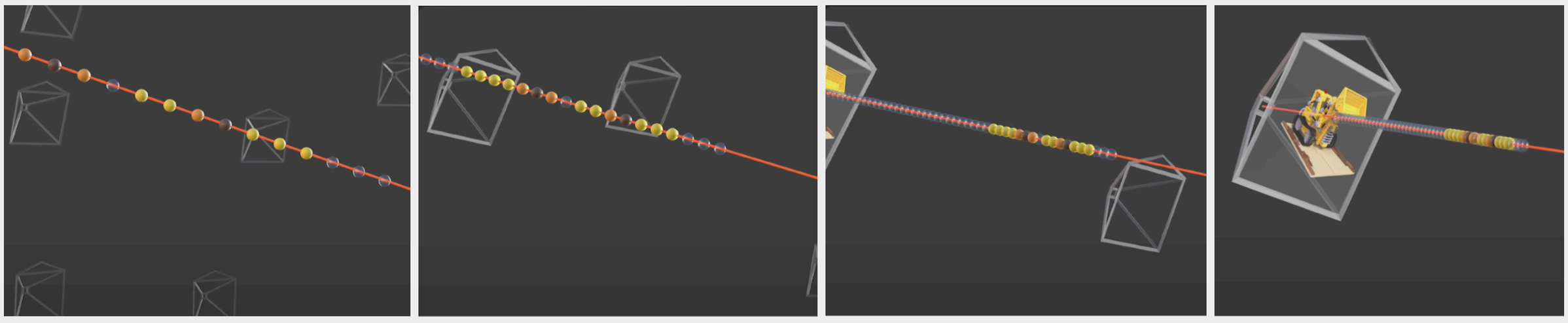


Differentiable Rendering (Volume Rendering)

- The expected color $\hat{C}(r)$ of camera ray r(t) = o + td with near and far bounds t_n and t_n.
- Using deterministic quadrature, it effectively MLP queried at a fixed discrete set of points in 3D spaces.

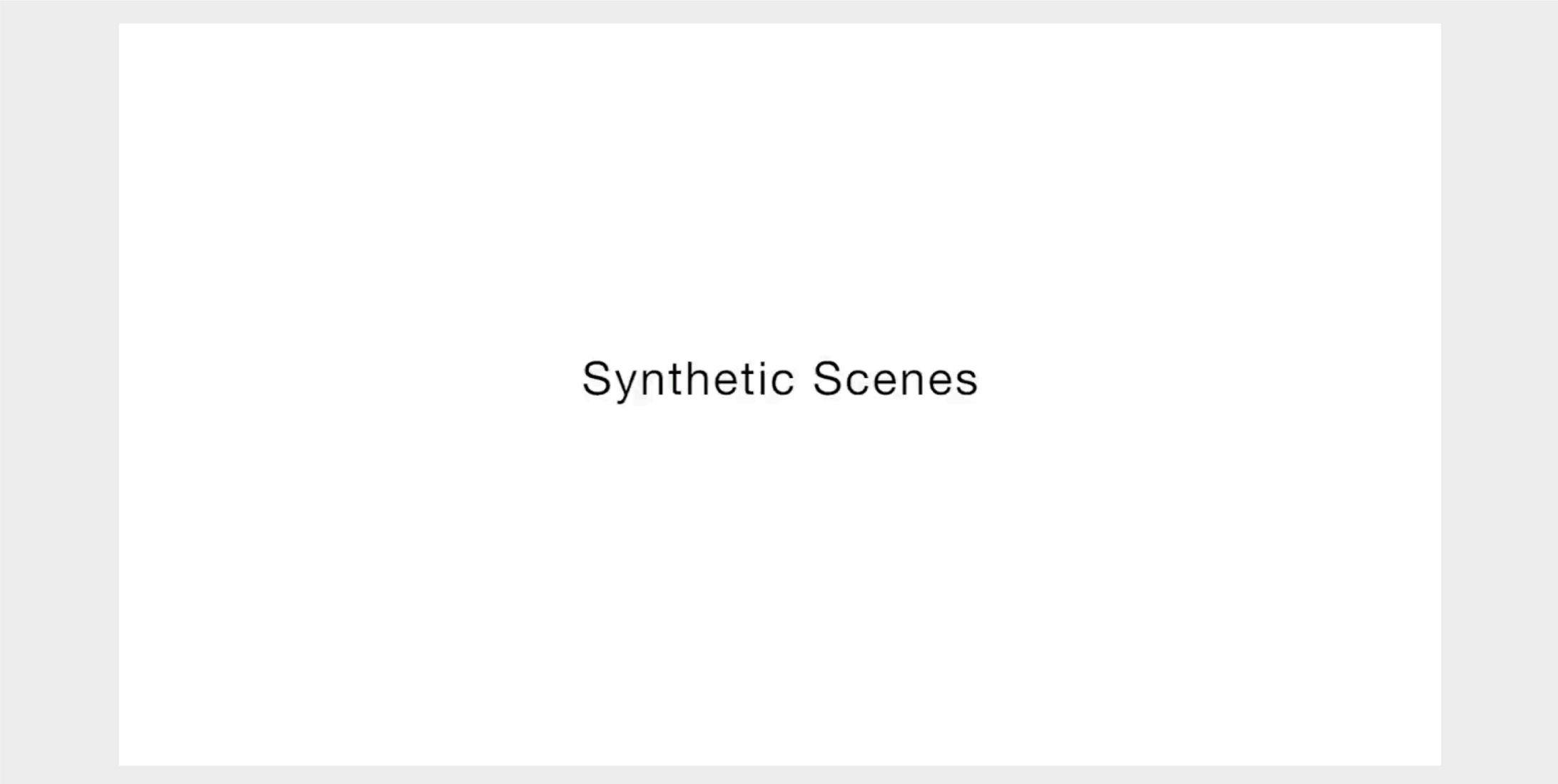
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right).$$

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



Neural Radiance Fields (NeRFs)

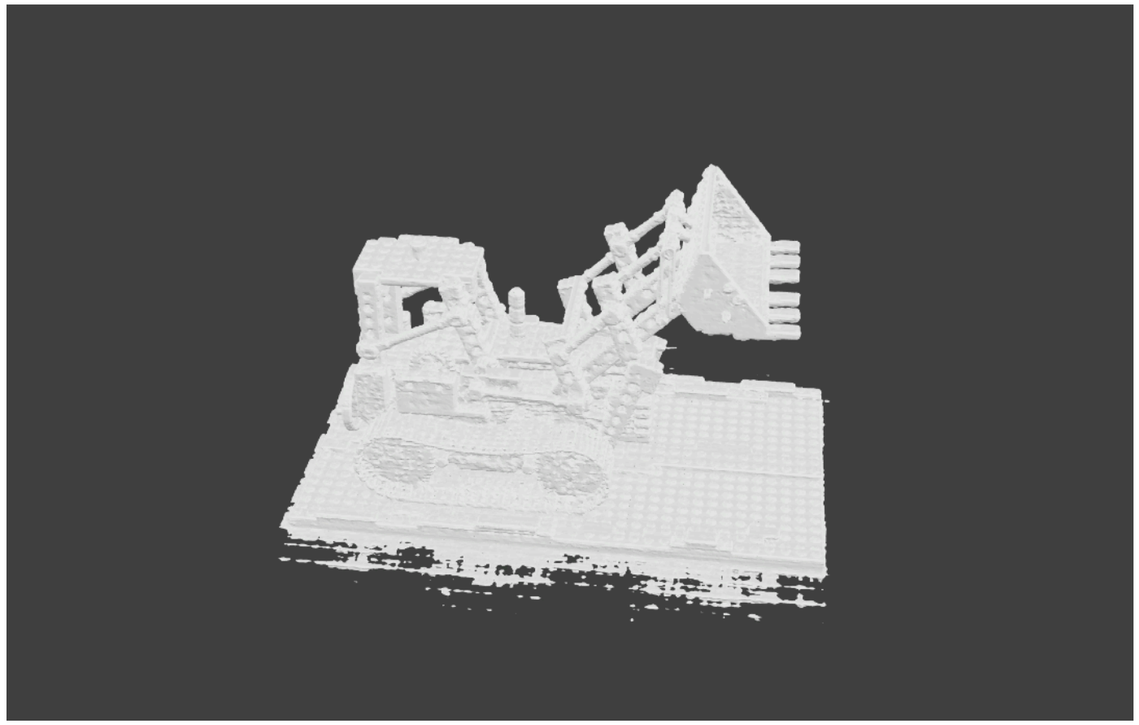
- Experimental Results
  - At that time, the NeRF achieves the best performance compared of other coordinate networks.



https://youtu.be/JuH79E8rdKc

Neural Radiance Fields (NeRFs)

- The output of volume rendering
  - NeRF not only generates novel-view synthesis (images) but also provides volume information for objects, including mesh information.
  - While the reconstructed surface may have some minor irregularities or be somewhat wiggly, the overall quality of the mesh is good, especially considering that it doesn't directly use 3D information.



https://youtu.be/JuH79E8rdKc

Enhancing Performance of NeRFs via Explicit Representation

Drawbacks of vanilla NeRF

- Since NeRF consists of multiple MLP layers, both training and inference can be time-consuming. Mapping c and σ values for specific 3D spaces requires extensive training:
  - Training: Approximately 20 hours per a scene.
    - Not support parallelism as MLP embedding necessitates sequential operations.

- To address issues, explicit parameterization methods have been developed since 2022.

1. Neural Sparse Voxel Fields (NSVF, NeurIPS2020) : Voxel Grid Features + Small MLPs
2. Direct Voxel Grid Optimization (DVGO, CVPR2022) : Voxel Grid without MLPs
3. Instant Neural Graphic Primitive (SIGGRAPH2022) : Hash function for Voxel Grid
4. TensoRF (ECCV2022) : Multi-planes encoding
5. K-Planes / HexPlane (CVPR2023) : Multi-planes encoding including time axis

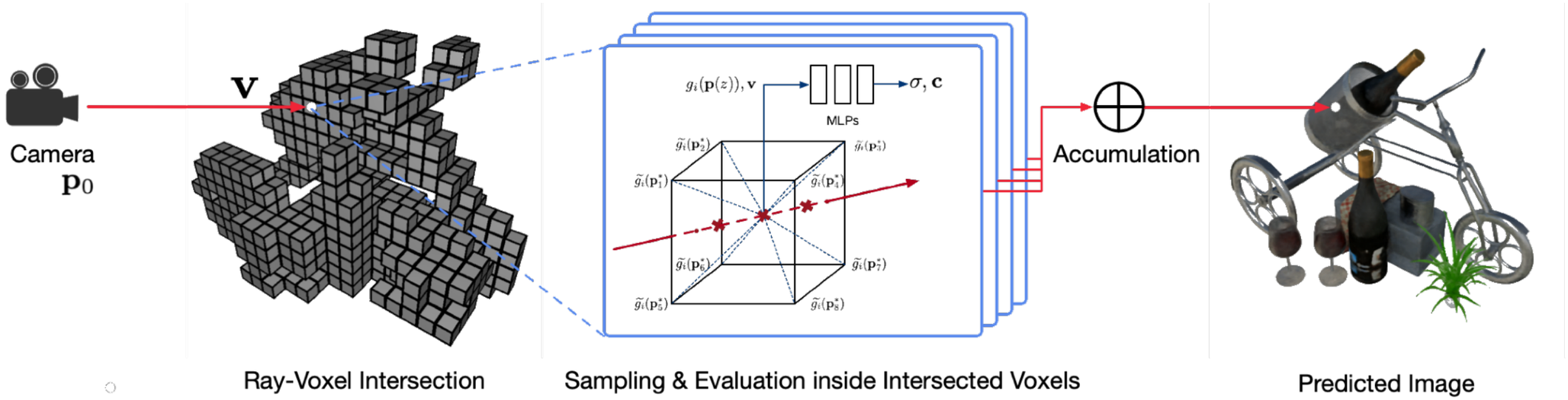
Neural Sparse Voxel Fields (NeurIPS2020)

✓ This model differs from MLP-based NeRF by creating a Voxel grid feature that corresponds to spatial space, enhancing the quality of generated images through training.

- Pioneering study under NeRF training pipeline: Voxel feature grids..

✓ This study addresses memory issues by pruning spatially empty regions post-training, improving upon traditional voxel-grid features.

- Note that this model still relies on multi-layer MLPs after spatial representation, which means it doesn't significantly speed up rendering and training times



Neural Sparse Voxel Fields (NeurIPS2020)

- This method compares of MLP-based NeRFs models and Voxel-grid method at that time.

✓ The baselines often lack high-frequency details, resulting in some blurry regions.

- However, this approach succeeds in generating clean and authentic-looking images. Regarding rendering speed, it doesn't outperform Neural Volume due to the multiple MLP layers, which limit the acceleration of rendering speeds.

Wine Holder

SRN (Sitzmann et al. 2019)
(Rendering speed: 1.10 s/frame)

NSVF
(Rendering speed: 1.68 s/frame)

Visual Results

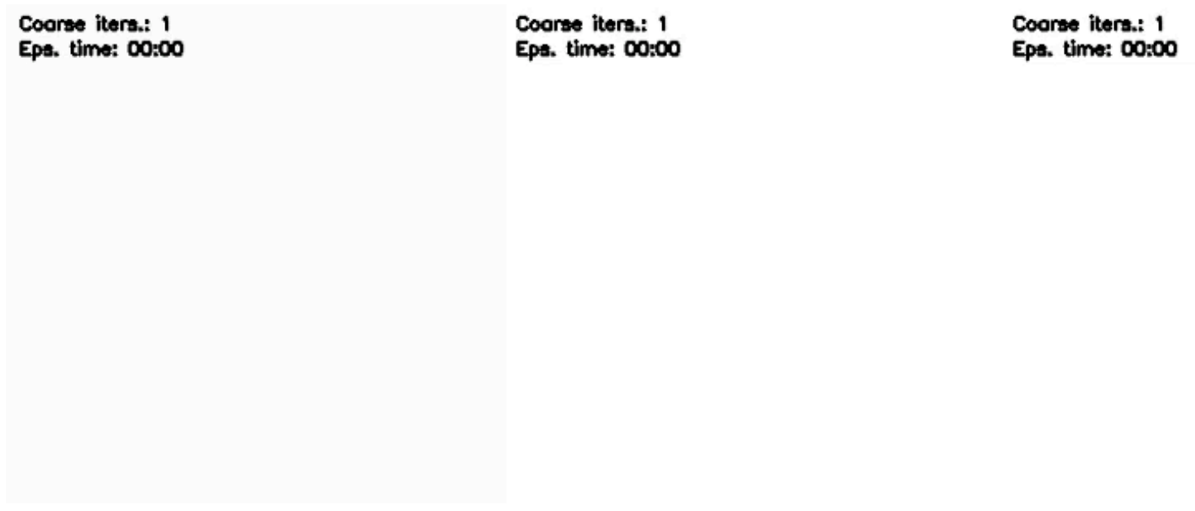
Models	Synthetic-NeRF PSNR	SSIM	LPIPS
SRN	22.26	0.846	0.170
NV	26.05	0.893	0.160
NeRF	31.01	0.947	0.081
NSVF^0	**31.75**	**0.954**	0.048
NSVF	31.74	0.953	**0.047**

Numerical Comparison

Direct Voxel Grid Optimization (CVPR2022)

- The previous work faces challenging in fast inference due to the use of MLP layers.
- Direct Voxel Grid Optimization addresses this issue by relying only on voxel grid features to predict $\hat{c}(r)$ and σ, resulting in significantly faster training and inference times
  - Fine Training: $\leq$ 15min / Coarse Training: $\leq$ 5min
- However, this model assumed the availability of substantial amount of training datapoints.

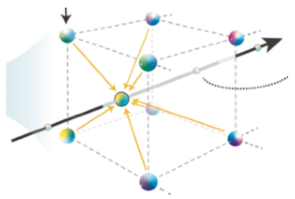
$$\alpha^{(\text{post})} = \underline{\text{activate}}\left(\underline{\text{interp}}\left(x, \text{Voxel Grid}\right)\right)$$



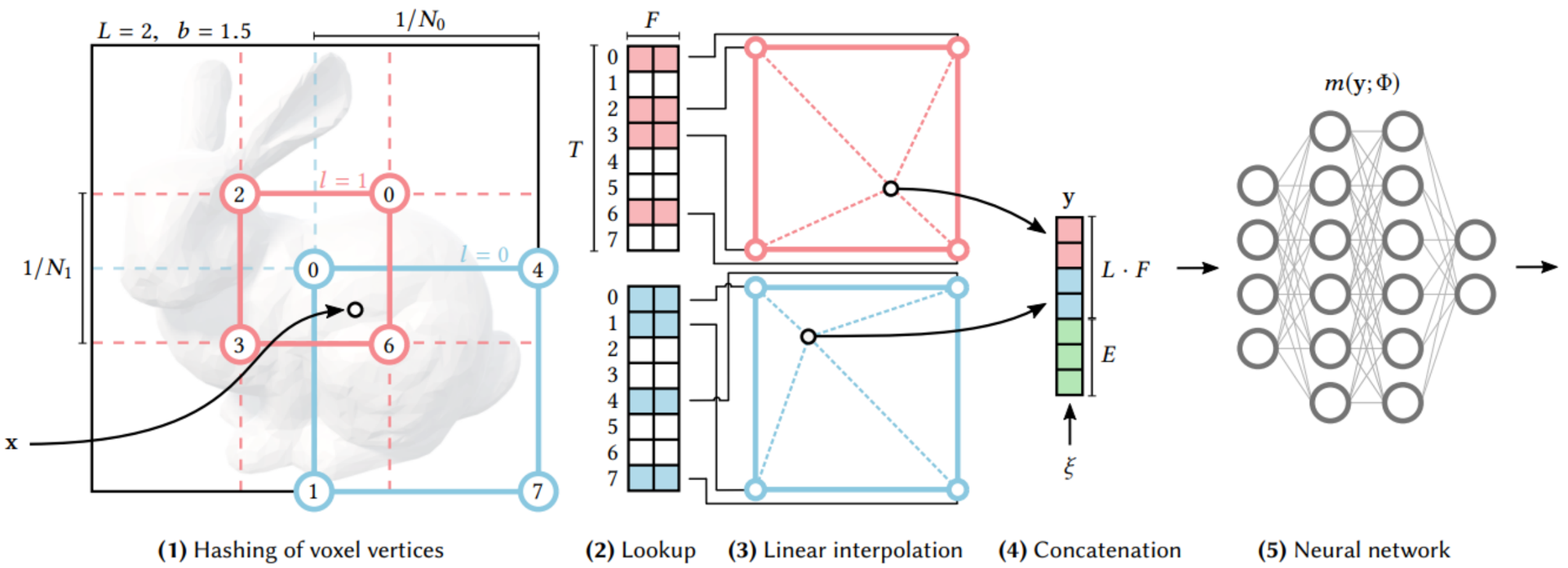
Visual experiment results

Instant Neural Graphics Primitives (SIGGRAPH2022)

- The hash function enables mapping each hash to duplicated feature grids, reducing memory consumption. This approach effectively increases capacity by incorporating multi-resolution feature grids, allowing for finer detail compared to previous methods.
- The hashing technique doesn't demand intensive learning and is straightforward to use for synthesizing data points like voxel grid. However, note that the training data must be sufficiently dense.



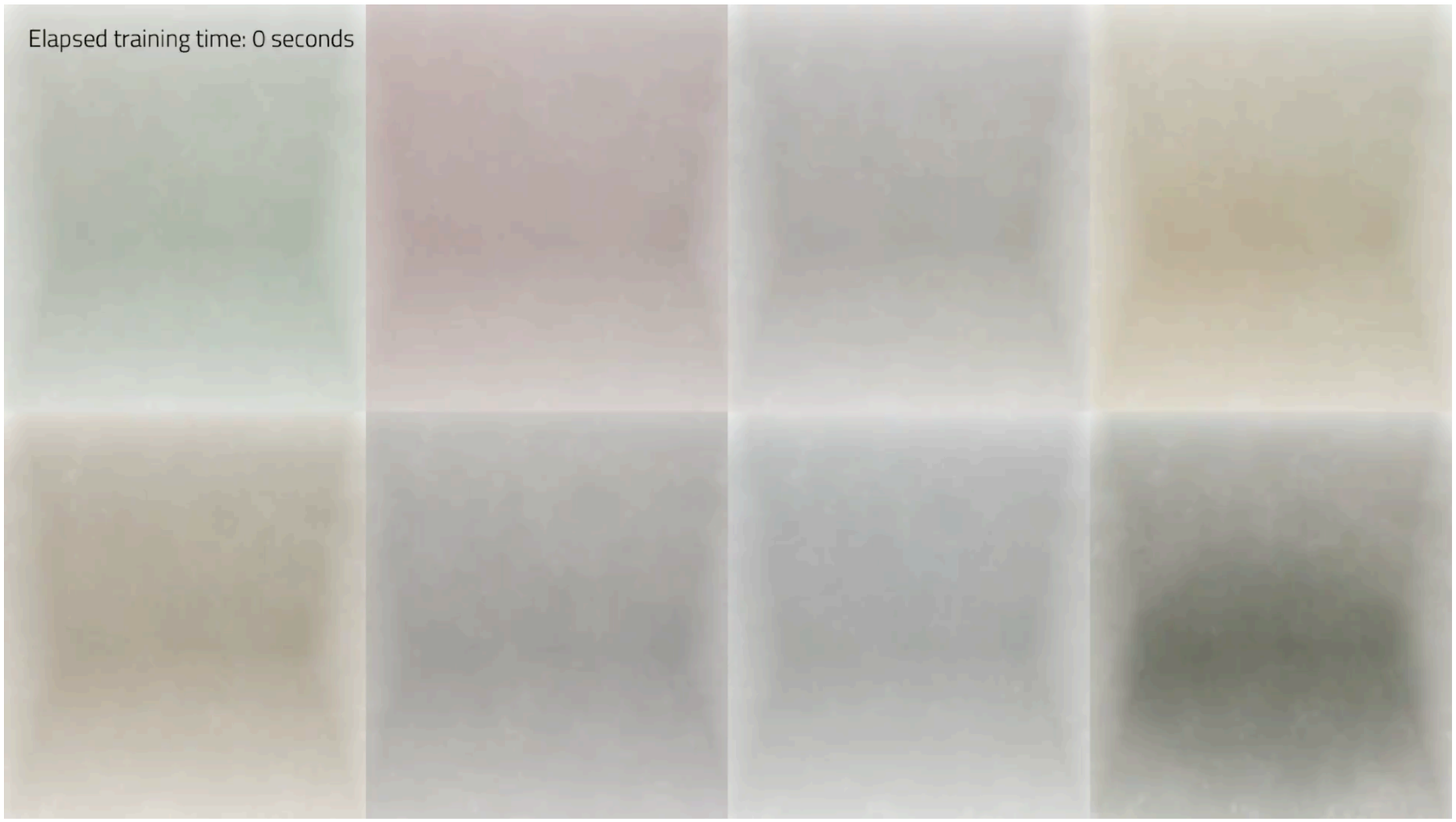
Parametric Encoding
(Linear interpolation)



Müller, Thomas, et al. "Instant neural graphics primitives with a multiresolution hash encoding." *arXiv preprint arXiv:2201.05989* (2022).

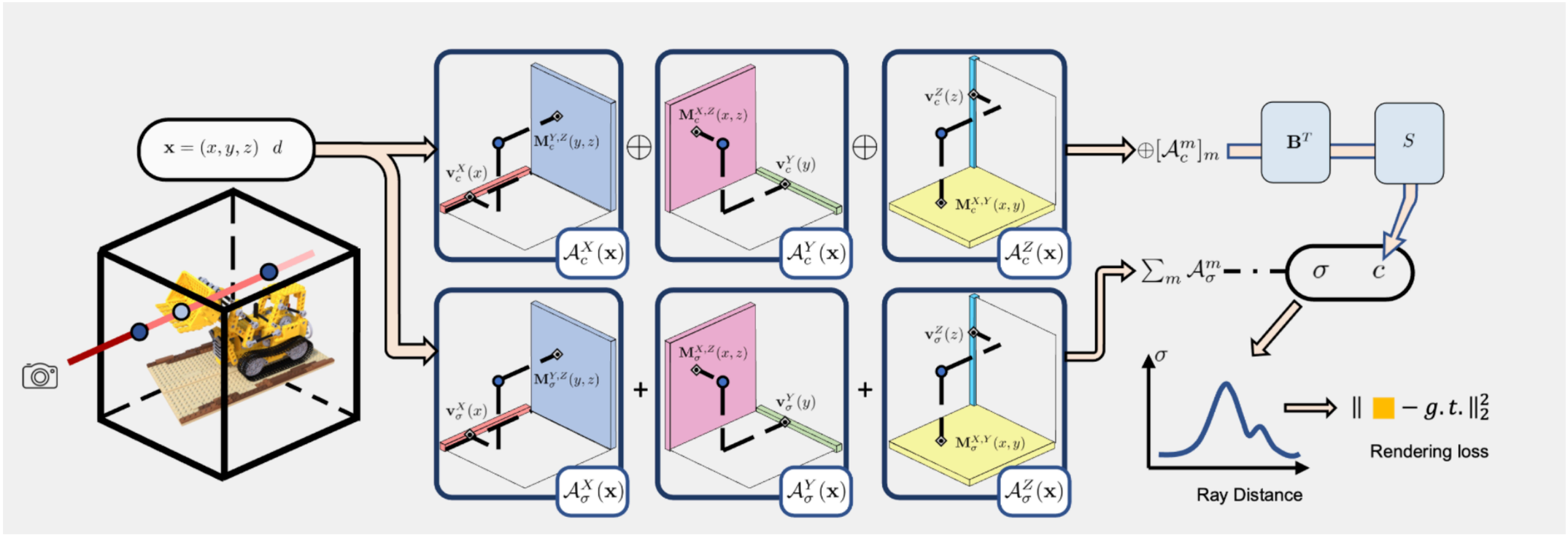
Instant Neural Graphics Primitive (SIGGRAPH2022)

- This model is entirely implemented using CUDA, resulting in significantly faster training and inference compared to other methods.
- However, it's worth mentioning that making modifications to the algorithm can be quite challenging.



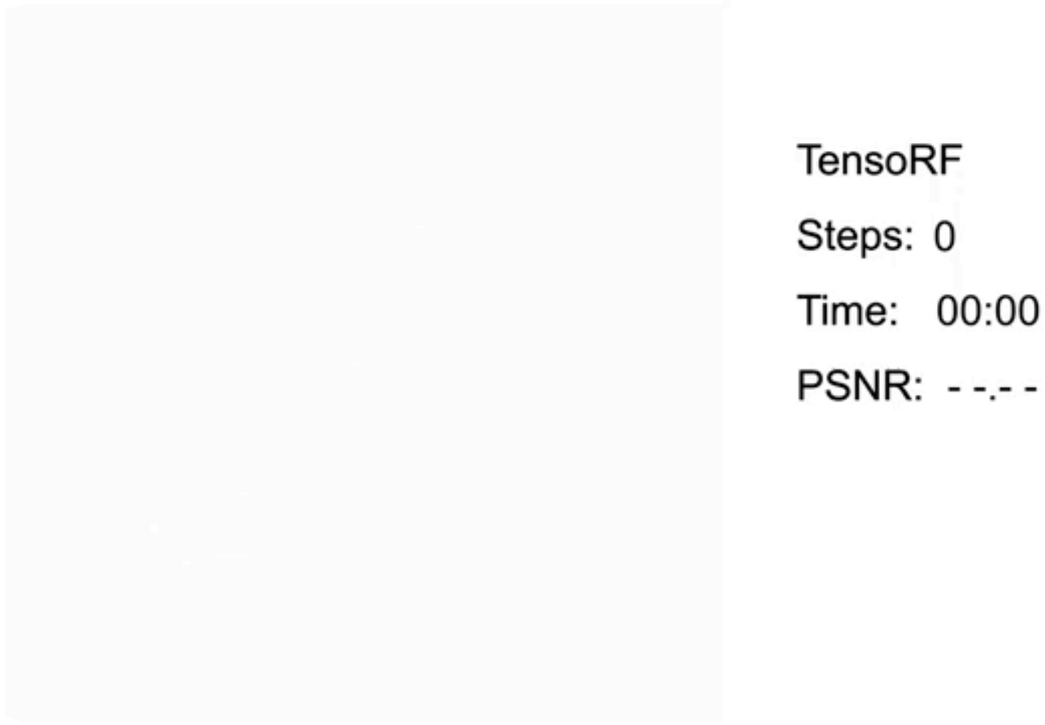
TensoRF (ECCV2022)

- Voxel-grid and hash-grid methods inherently involve a substantial number of parameters, primarily due to their reliance on features in 3D spaces (Curse of Dimensionality).
- In contrast, TensoRF adopts a different approach by leveraging spatial features in 2D spaces (Planes), and then these features are interpolated using 3D points projected onto the feature planes.
- This results in a model with fewer parameters, while still achieving comparable



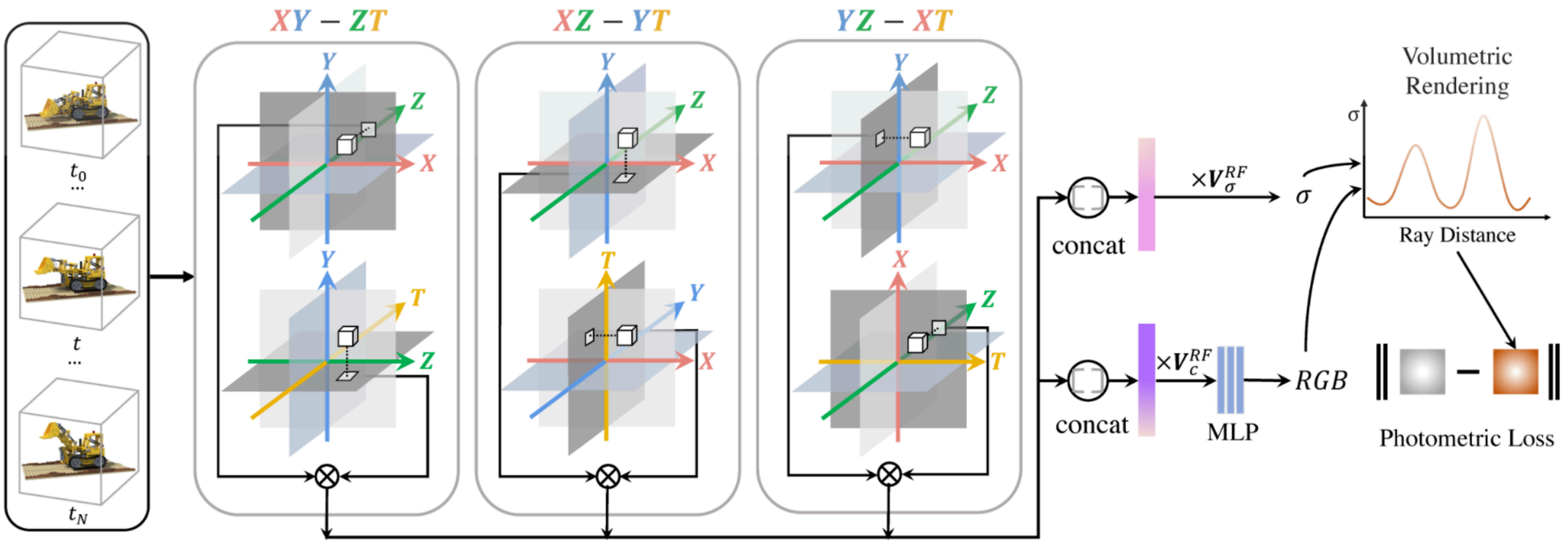
TensoRF (ECCV2022)

- This model offers fast training, even though it doesn't rely on CUDA implementation. All operations are conducted within auto-differentiable frameworks like PyTorch, making it easy to understand how it works and modify algorithms.
- While the number of parameters is lower than Voxel-grid approaches, it's worth noting that this model consumes more memory during training.



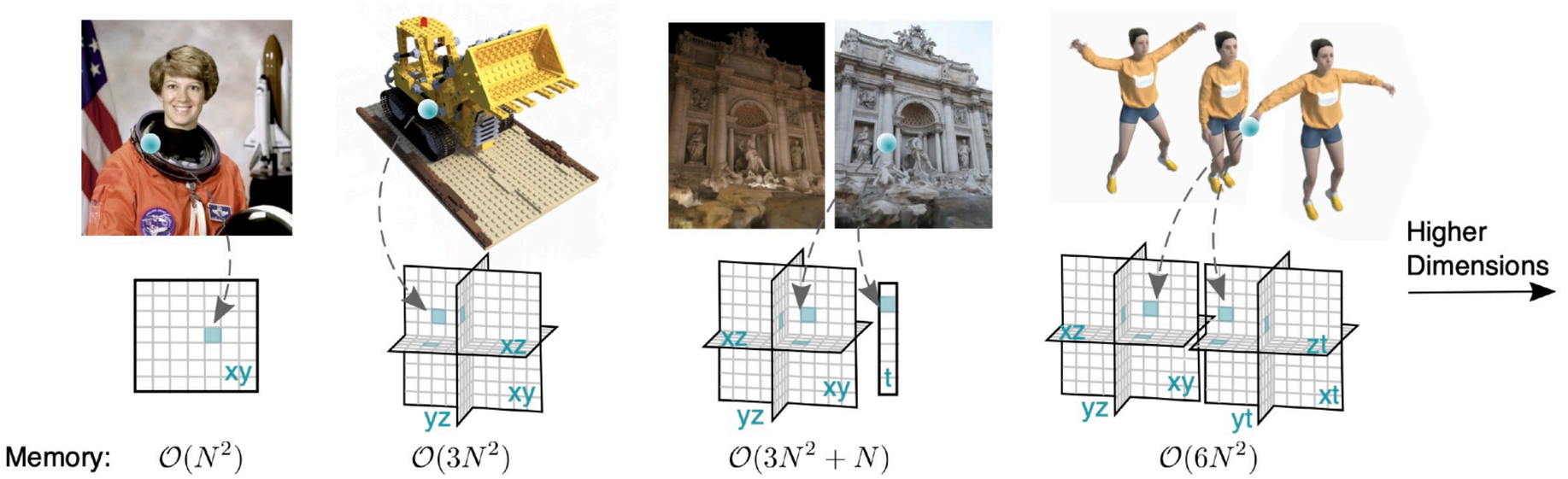
HexPlane (CVPR2023)

- In contrast to TensoRF, which primarily uses Tri-planes, HexPlane utilizes pairs of planes that include spatial features and spatial-temporal feature planes.
  - e.g) *XY* plane + *ZT* planes
- HexPlane introduces six planes, including the time-axis, which efficiently learns dynamic NeRF and achieves fast training.



K-Planes (CVPR2023)

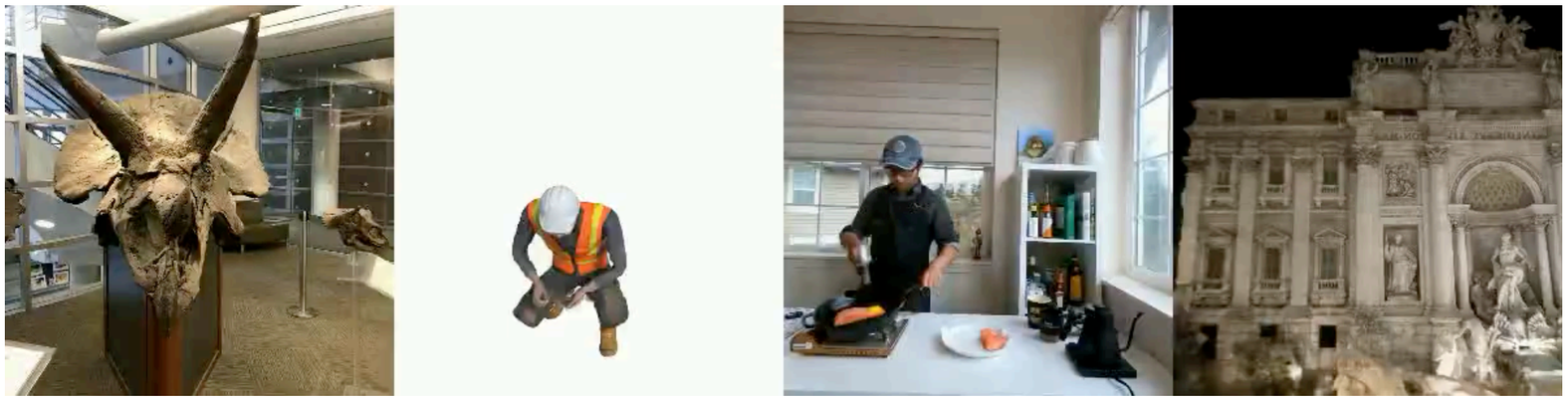
- While HexPlane focuses specifically on Dynamic NeRF, K-Planes introduces a more generalized concept through multi-plane encoding.
- K-Planes can handle not only static NeRF with varying appearances, such as transitions from daytime to nighttime but also dynamic NeRF.
- The core concept is similar to HexPlane, but K-Planes demonstrates different embedding types depending on each specific case.



K-Planes & HexPlane (CVPR2023)

- In the case of dynamic NeRF (4D NeRF), these methods include regularization techniques focused on image denoising regularization. (To be discussed in detail)
- Plane features can be thought of as image planes. Image denoising methods help counter overfitting to time-sparsity because time-dependent information is relatively scarcer compared to spatial information.
- Current MLP-based method requires over 1400 GPU hours of training for a single view, while our method finishes it within 10 hours with the same quality, which is over 100x

HexPlane result



K-Plane results

Summary of this chapter

- The development of NeRF and its representation:

1. **Implicit learning with positioning encoding (Vanilla NeRF)** : It offers a flexible, model-agnostic representation. However, it comes with a drawback of requiring substantial training time and being less compatible with parallelism.

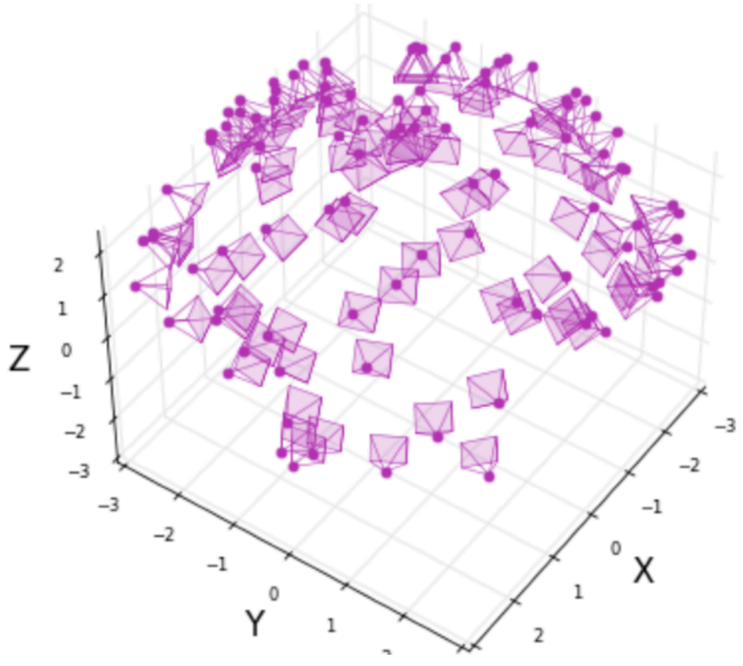
2. **Neural Voxel-Grid (DVGO, iNGP)** : Those open explicit spatial features to accelerate training and inference time while improving performance. However, they consume a significant amount of memory due to the curse of dimensionality; ; $\mathcal{O}(n^d)$

3. **Multi-Plane Encoding (TensoRF, K-Planes)** : These models do not only achieves state-of-art performance in both static and dynamic NeRFs (3&4 D), but also they do without excessive memory, thanks to their plane features in 2D spaces; $\mathcal{O}(kn^2)$

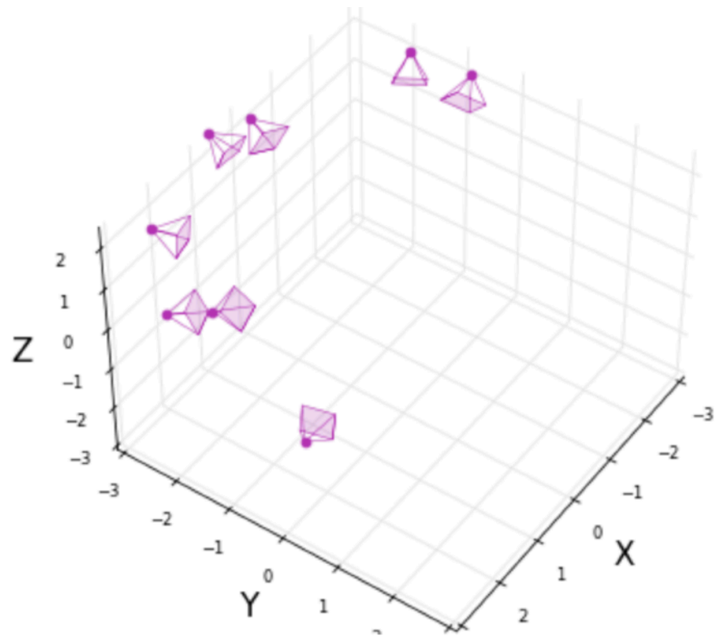
NeRF in the sparse-input regimes

How those work in sparse-inputs?

- The {Lego, Hotdog, …} dataset, also known as the NeRF_Synthetic dataset, assumes a training set with 100 views.
  - However, this concept doesn't align closely with practical usage, as in real-world scenarios, we typically require at least 100 shots for accurate representation.
- In sparse-input scenarios, we aim to use only 8 views, which presents a significant challenge.



Training camera poses : 100

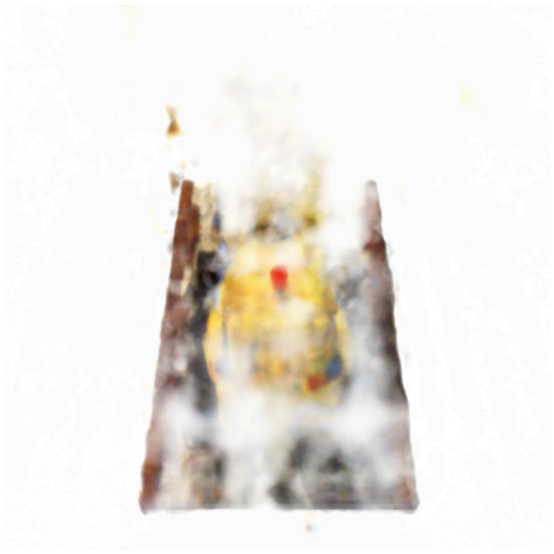


Training camera poses : 8

Simplified_NeRF (Naive approach)

- In vanilla NeRF, the high-frequency positioning encoding can lead to artifacts that harm performance.
- As a simple approach to address this issue, we can drop the highest frequency of positioning encoding. While this can help reduce floating artifacts, it may not precisely capture high-frequency details in the data.

Model	Lego	Drums	Ship	Avg. of all scenes
Simplified NeRF	12.45	14.188	18.648	19.22

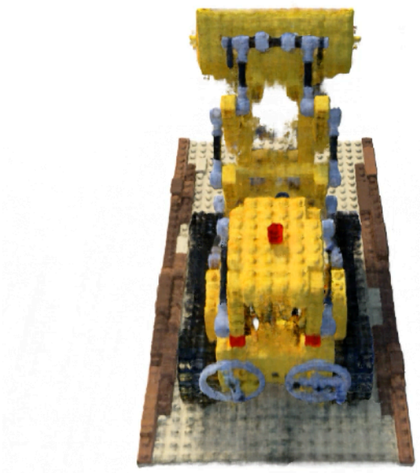


Simplified NeRF

FreeNeRF (CVPR2023)

- Carefully managing high-frequency embeddings is essential for achieving comparable results. This method adopts a gradual activation of embeddings from low-frequency to high-frequency during training.
- Specifically, at the initial step, high-frequency embeddings are masked. As training progresses, this method gradually activates the high-frequency components. This approach helps prevent overfitting when training high-frequency embeddings for training data points.

Model	Lego	Drums	Ship	Avg. of all scenes
Free NeRF	24.12	19.99	23.01	23.40

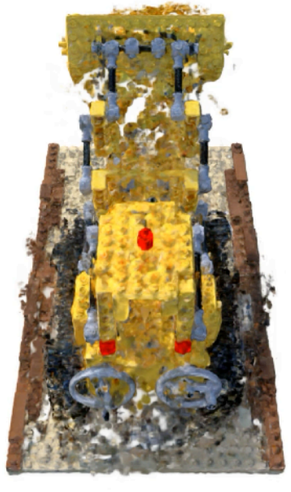


FreeNeRF

DVGO / iNGP (Voxel-grid approaches)

- Explicit representation methods, like Voxel-grid approaches, work by explicitly representing spatial information. They can be sensitive to the number of training data points, potentially leading to overfitting when sparse input data is provided.
  - Voxel-grid approaches rely on features in 3D spaces, and overfitting becomes evident when dealing with sparse-input data.
- While well-known regularization techniques can help improve performance to some extent, they may not result in a significant performance enhancement in these scenarios.

Model	Lego	Drums	Ship	Avg. of all
DVGO	20.85	16.54	18.17	20.57

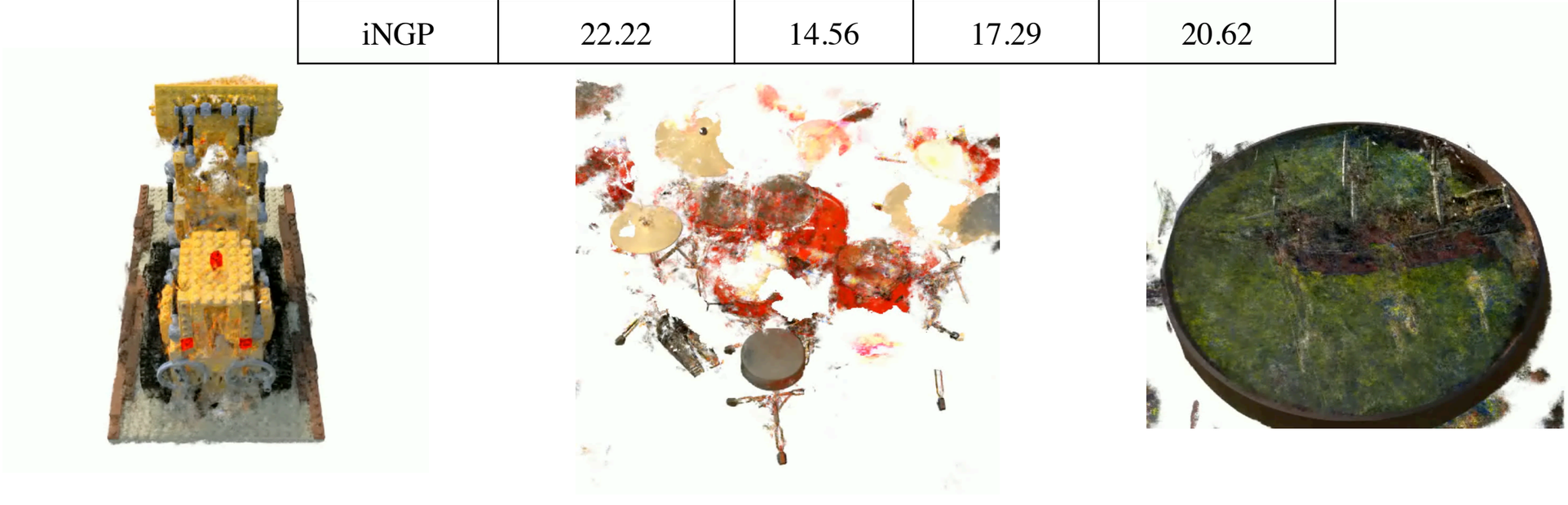


DVGO

DVGO / iNGP (Voxel-grid approaches)

- Explicit representation methods, like Voxel-grid approaches, work by explicitly representing spatial information. They can be sensitive to the number of training data points, potentially leading to overfitting when sparse input data is provided.
  - Voxel-grid approaches rely on features in 3D spaces, and overfitting becomes evident when dealing with sparse-input data.
- While well-known regularization techniques can help improve performance to some extent, they may not result in a significant performance enhancement in these scenarios.

Model	Lego	Drums	Ship	Avg. of all
iNGP	22.22	14.56	17.29	20.62



iNGP

TensoRF (ECCV2022)

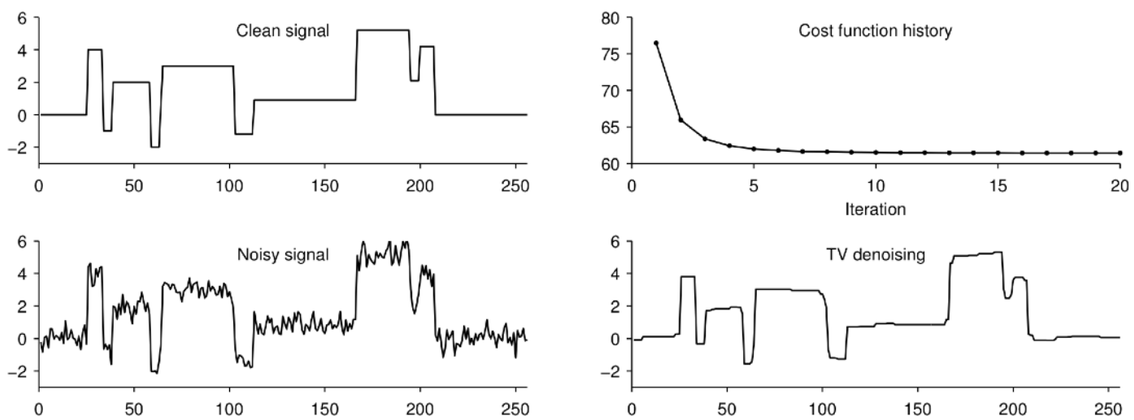
- TensoRF utilizes 2-dimensional feature spaces with the projection of 3D points onto these feature spaces, which enhances its stability in sparse-input regimes.
- Notably, even without the assistance of external regularization, vanilla TensoRF achieves performance comparable to FreeNeRF. Furthermore, TensoRF boasts faster training and inference times compared to FreeNeRF.

Model	Lego	Drums	Ship	Avg. of all scenes
TensoRF	26.28	15.94	20.29	23.15

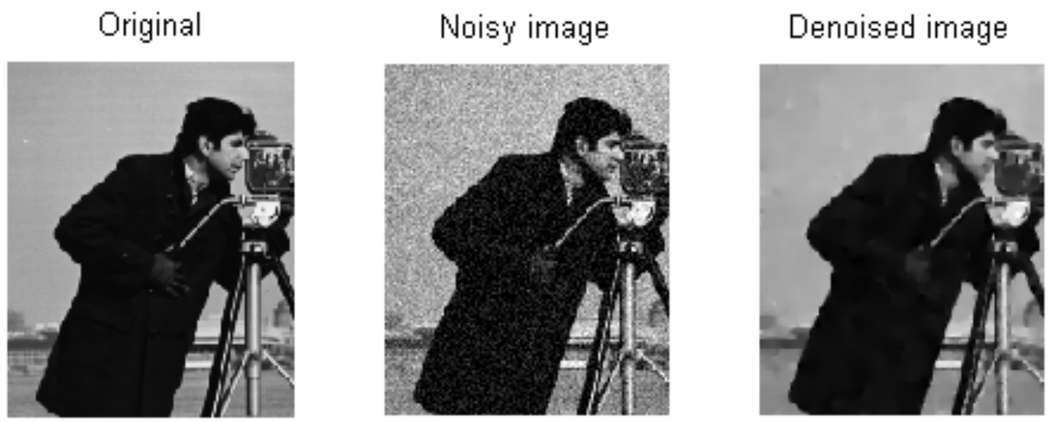
K-Planes (CVPR2023)

- K-Planes utilizes a regularization, "Total variational (TV) loss" for stable training.
- The Total Variation (TV) loss is commonly employed for denoising signals, such as noisy images. This loss helps remove noise while preserving true signals.
- Technically, this loss is written as follows:

$$\mathscr{L}_l(P) = \sum_c \sum_{hw} \left(\left\| P^c_{h+1,w} - P^c_{h,w} \right\|_2^2 + \left\| P^c_{h,w+1} - P^c_{h,w} \right\|_2^2 \right).$$



1D Denoising example



2D Image denoising

https://en.wikipedia.org/wiki/Total_variation_denoising

K-Planes (CVPR2023)

- K-Planes leverages the Total Variation (TV) loss for effectively removing floating artifacts. During training, this method minimizes both the photometric loss and denoising regularization on plane features (TV loss) simultaneously.
  - This method employs a weight ratio of {1:0.001} between two terms.
- This approach leads to a significant improvement in performance, surpassing even the capabilities of FreeNeRF.

Model	Lego	Drums	Ship	Avg. of all scenes
TensoRF	26.52	20.43	21.34	24.24

K-Planes

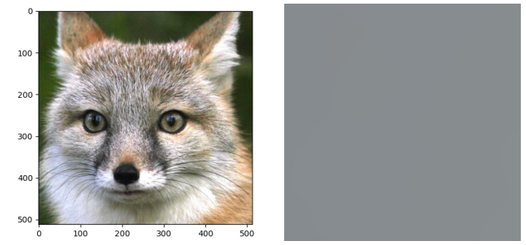
TensoRF & K-Planes

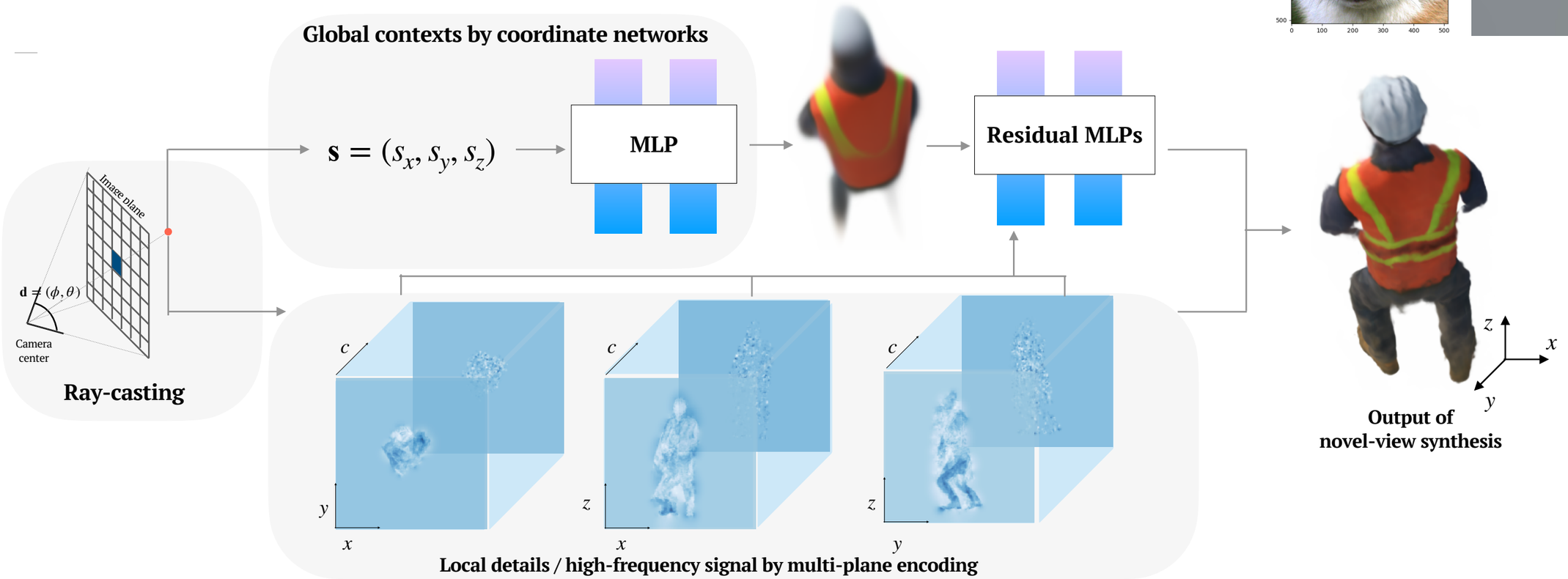
- Although TV (Total Variation) loss is effective at removing floating artifacts and improving performance by increasing weights for TV regularization.
- It can also introduce a different type of artifacts that appear authentic but were not present in the training data.

TensorRefine (NeurIPS-workshop 2023)

- We found that (low-frequency positional encoding) coordinate network has relatively less influence when trained them simultaneously, so that we develop curriculum training strategy; a coordinate network is trained first, and followed by sequential training of multiple-encoding.

- This strategy is quite effective to train complicated scenes such as `drums`, as well as fast motion scene `standup`.





TensorRefine

- This method is not significantly affected by the denoising regularization parameter, primarily because of the foundation provided by the low-frequency embedding from the coordinate network.
- Consequently, it excels in few-shot tasks and achieves the best performance in such scenarios.

Model	Lego	Drums	Ship	Avg. of all scenes
TensorRefine	26.73	19.55	24.27	24.56

TensorRefine

TensorRefine

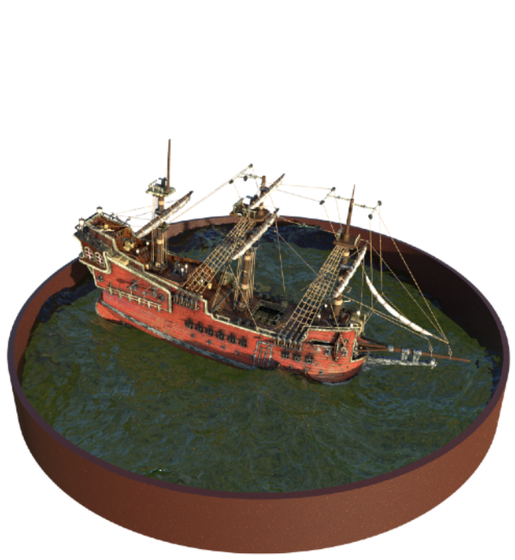
- Even though high regularization weights exposed, this method does not depict artificial artifacts that look authentic, but not present in the training data.
- Furthermore, this method is more stable against varying denoising regularization, if this parameter varies, this method less influence compared of TensoRF and K-Planes
  - When excessive denoising weights, the images simply appear faded or less vibrant.

Table 3: Average PSNR across all scenes varying denoising regularization λ_1. The hyphen indicates not converged

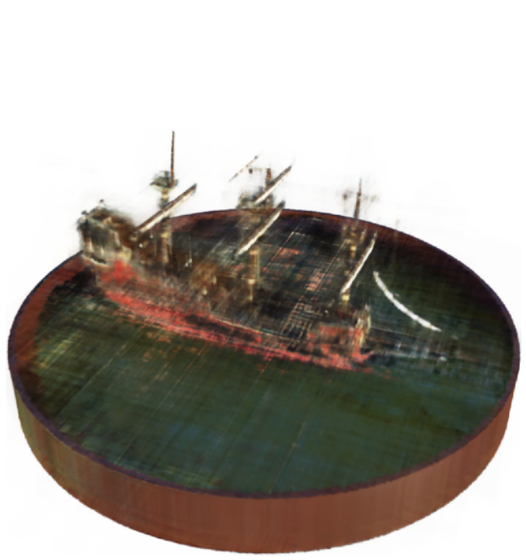
λ_1	Static NeRF (8 views)			D-NeRF (25 views)		
	TensoRF	K-Planes	ours	HexPlane	K-Planes	ours
0.0001	24.10	24.31	23.68	22.83	24.32	24.67
0.001	24.98	24.28	24.47	23.86	24.01	25.38
0.01	-	24.28	24.55	24.15	24.02	25.74
0.1	-	23.64	24.23	23.46	23.55	25.84
1.0	-	22.05	22.99	21.95	22.62	25.42

Sensitivity Analysis of Regularization

- Furthermore, this method is more stable against varying denoising regularization, if this parameter varies, this method less influence compared of TensoRF and K-Planes
  - When excessive denoising weights, the images simply appear faded or less vibrant.

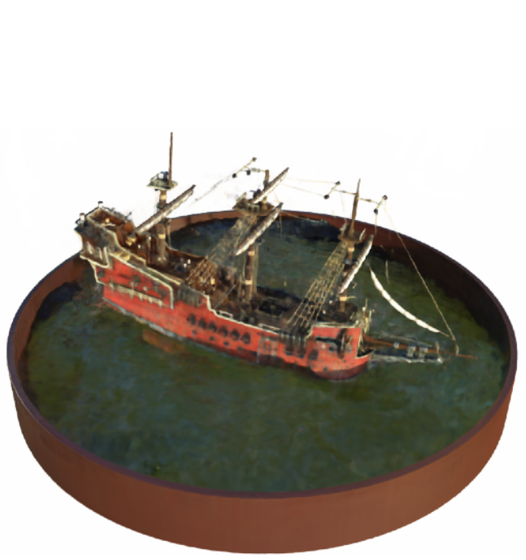


Ground Truth



TensoRF with $\lambda_1 = 1.0$

K-Planes with $\lambda_1 = 1.0$



TensorRefine
$\lambda_1 = 1.0$

Quantitative Results (1)

- Dataset : NeRF Synthetic (Training data: 8 Views / Test data : 200 views)
  - In scenarios like {Lego, Hotdog, Mic}, multi-plane encoding approaches demonstrate competitive performance compared to other methods.
  - In scenes with a mix of low and high-frequency details, such as {Drums, Ship}, these methods may struggle to depict accurately. Nevertheless, TensorRefine addresses these challenges by incorporating coordinate networks and multi-plane encoding, resulting in improved performance.

Models	**PSNR** ↑								**Avg. PSNR** ↑	**Avg. SSIM** ↑	**Avg. LPIPS** ↓
	chair	drums	ficus	hotdog	lego	materials	mic	ship			
Simplified_NeRF	20.35	14.19	<u>21.63</u>	22.57	12.45	18.98	24.95	18.65	19.22	0.827	0.265
DietNeRF	21.32	14.16	13.08	11.64	16.12	12.20	24.70	19.34	16.57	0.746	0.333
HALO	24.77	18.67	21.42	10.22	22.41	21.00	24.94	21.67	20.64	0.844	0.200
FreeNeRF	26.08	<u>19.99</u>	18.43	<u>28.91</u>	24.12	<u>21.74</u>	24.89	<u>23.01</u>	23.40	0.877	0.121
DVGO	22.35	16.54	19.03	24.73	20.85	18.50	24.37	18.17	20.57	0.829	0.145
VGOS	22.10	18.57	19.08	24.74	20.90	18.42	24.18	18.16	20.77	0.838	0.143
iNGP	24.76	14.56	20.68	24.11	22.22	15.16	26.19	17.29	20.62	0.828	0.184
TensoRF	26.23	15.94	21.37	28.47	26.28	20.22	26.39	20.29	23.15	0.864	0.129
K-Planes	<u>27.30</u>	**20.43**	**23.82**	27.58	<u>26.52</u>	19.66	**27.30**	21.34	<u>24.24</u>	**0.897**	**0.085**
Ours	**28.02**	19.55	20.30	**29.25**	**26.73**	**21.93**	<u>26.42</u>	**24.27**	**24.56**	<u>0.896</u>	<u>0.092</u>

Quantitative Results (2)

Optimization and Statistical Inference LAB

- We define stability as the minimal performance discrepancy between test viewpoints adjacent to and not adjacent to the training views.
  - Variance of PSNR across all test viewpoints in the static NeRF dataset.
- FreeNeRF, which uses MLP and sinusoidal encoding, records the lowest variance among baselines. While K-Planes reduces instability than these methods, its variances still do not reach the level of ours. Quantitatively, ours achieves comparable results to FreeNeRF.

	Variance (↓)			
	chair	**lego**	**ship**	**Total**
FreeNeRF	5.07	**6.42**	6.48	**17.31**
iNGP	8.43	7.78	6.03	23.95
TensoRF	10.88	10.27	5.71	23.22
K-Planes	10.74	10.76	11.48	19.61
Ours	**3.82**	8.72	**6.01**	18.23

Variance of PSNR in each scene (↓)

Summary

- In summary, the existing NeRF models tend to overfit in the positioning encoding, making it challenging to train in sparse data situations. To address this, it has been observed that adjusting the frequency of positioning encoding during training can help mitigate this issue.

- Additionally, it has been found that the 2D multi-plane method is more robust than 3D Voxel grids. The introduction of regularization in this context can yield higher performance compared to the original FreeNeRF model.

- However, these methods face challenges in obtaining comprehensive information due to the interpolation of local features. Therefore, combining coordinate-based networks without positioning encoding and multi-plane encoding proved to achieve the best performance.

Q&A