Photo-realistic Free-viewpoint Rendering with Neural Sparse Voxel Fields

Lingjie Liu

Max Plank Institute for Informatics

Background

Conventional computer graphics modeling and rendering pipeline

- Acquiring a detailed appearance and geometry model
- Global illumination rendering

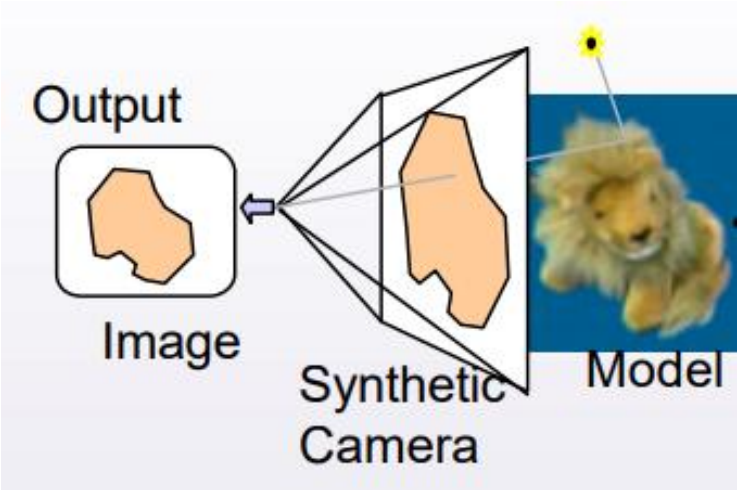


Image from [Cohen et al. 1999]

Background

Photo-realistic rendering of real-world scenes using conventional computer graphics pipeline is difficult.

The quality of existing reconstruction techniques is not good enough to support photo-realistic rendering, especially for the following challenging cases.

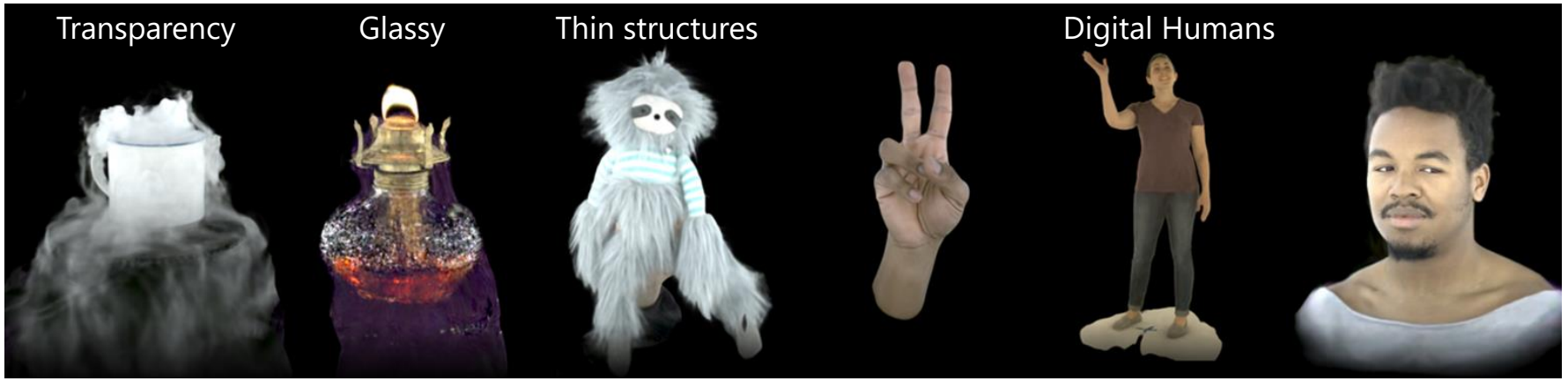


Image from [Lombardi et al. 2019]

Background

Image-based Rendering (IBR) = 3D model + image-based view interpolation

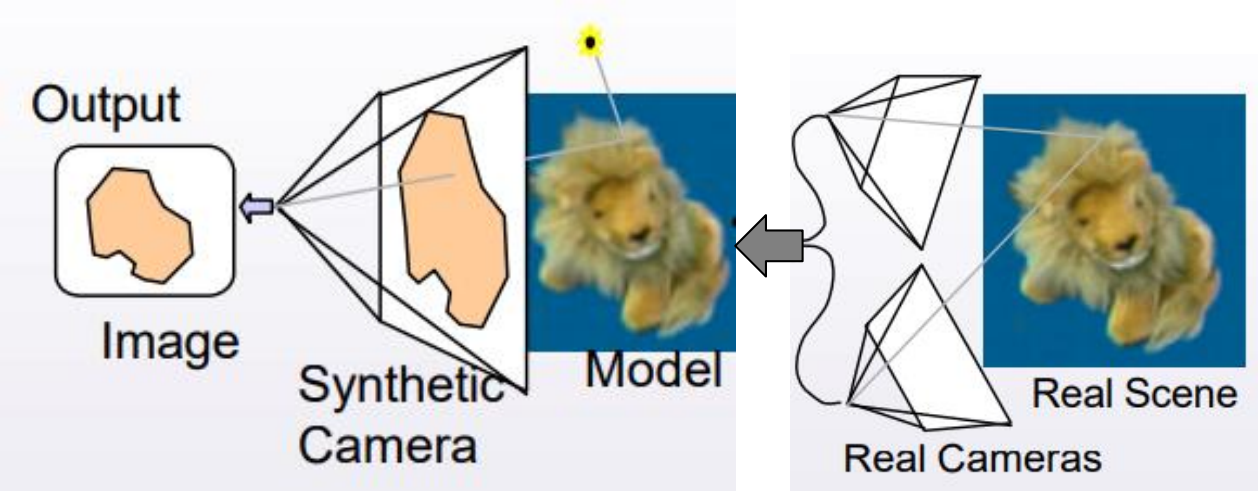


Image from [Cohen et al. 1999]

Limitations: 1) High storage requirements; 2) Limited control over results; 3) Scene-specific.

Background

What is neural rendering?

"Deep neural networks for **image or video generation**

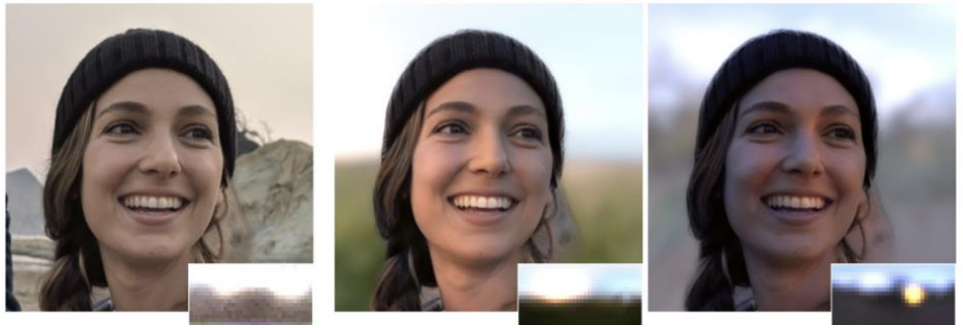
that enable **explicit or implicit control**

of **scene properties**"

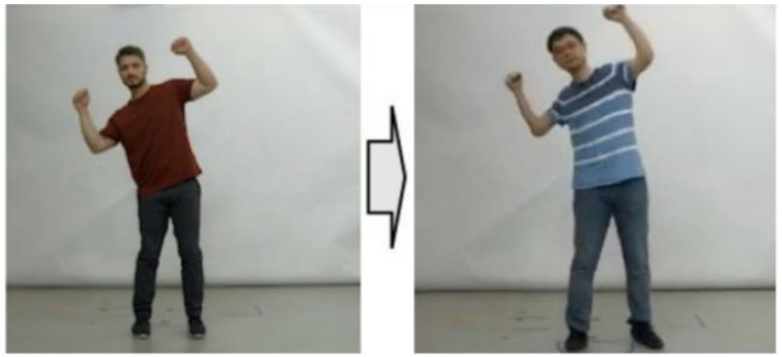
Background

Neural Rendering has various applications

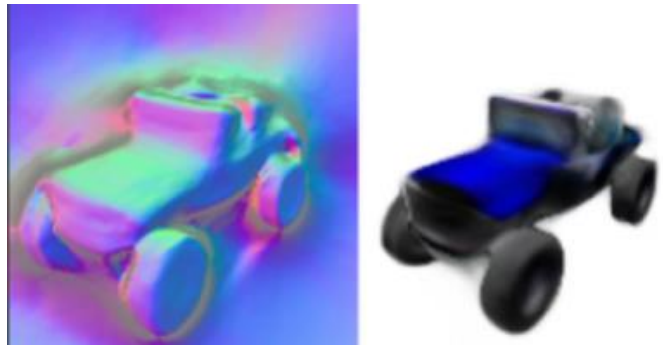
AR / VR



Relighting



Reenactment

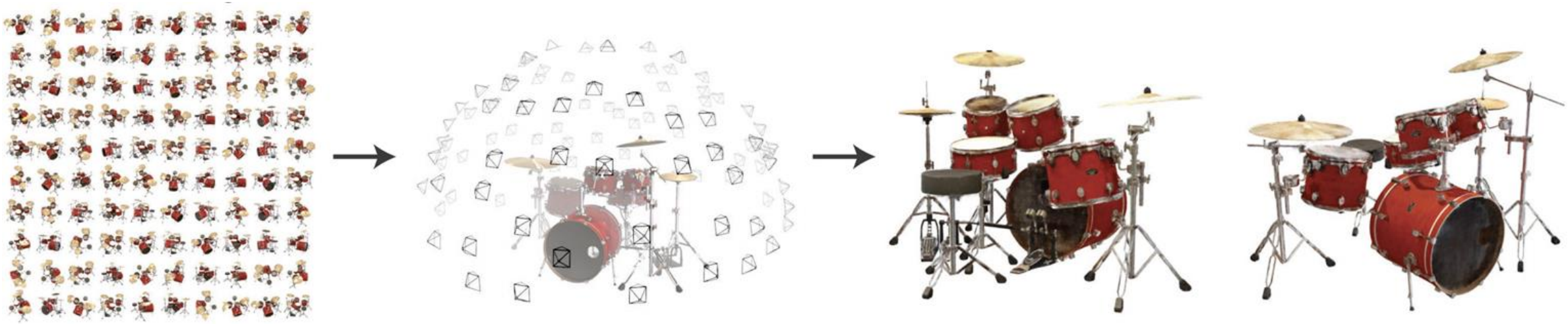


Free-viewpoint Rendering

Background

Neural scene representations and neural rendering for free-viewpoint rendering

- **Scene representation:** mapping every spatial location to a feature representation that describes local geometry and appearance information;
- **Rendering:** synthesizing novel view images using the learnt representations with computer graphics methods.



Input Images

Learned Scene Representation

Synthesized Novel Views

Image from [Mildenhall et al., 2020]

Related Works

Novel view synthesis with a coarse 3D geometry as input

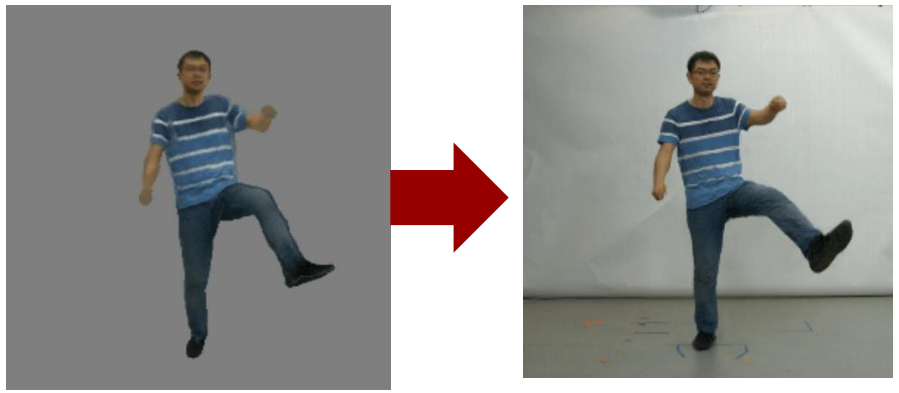
Point cloud:

(a) Input deep buffer (b) Output renderings

[Meshry et al. 2019],
[Martin Brualla et al. 2018],
[Aliev et al. 2019], ...

Image from [Meshry et al. 2019]

Textured meshes:



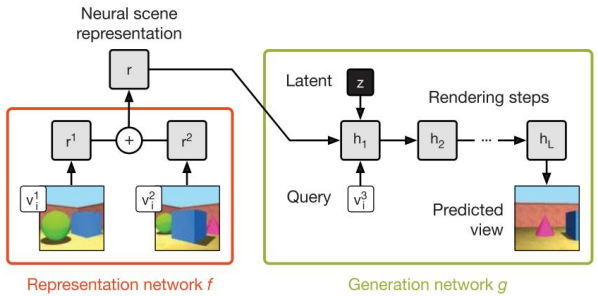
[Thies et al. 2019],
[Kim et al. 2018],
[Liu et al. 2019],
[Liu et al. 2020], ...

Image from [Liu et al. 2020]

Related Works

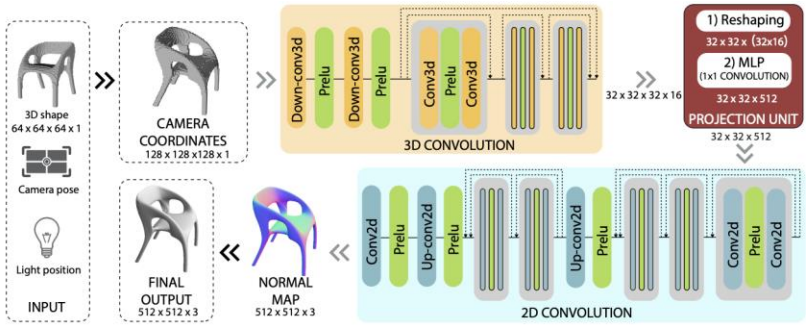
max planck institut informatik

Novel view synthesis without any 3D input

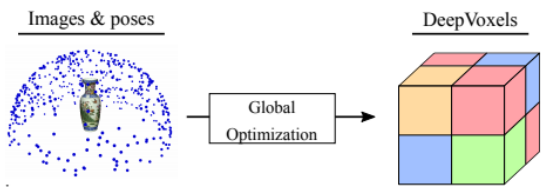


Generative Query Networks
[Eslami et al. 2018]

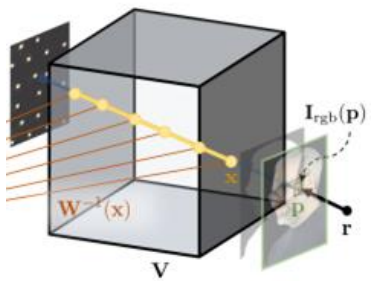
[Flynn et al., 2016; Zhou et al., 2018b; Mildenhall et al. 2019]
Multiplane Images (MPIs)



RenderNet[Nguyen-Phuoc et al. 2018]
Voxel Grids + CNN decoder

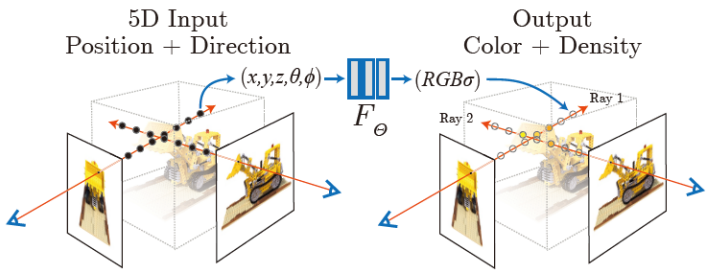


DeepVoxels
[Sitzmann et al. 2019]



Neural Volumes
[Lombardi et al. 2019]

SRN [Sitzmann et al. 2019b]



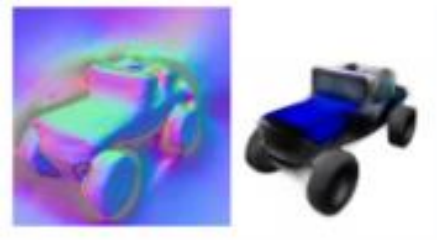
NeRF [Mildenhall et al. 2020]

Voxel Grids + Ray Marching

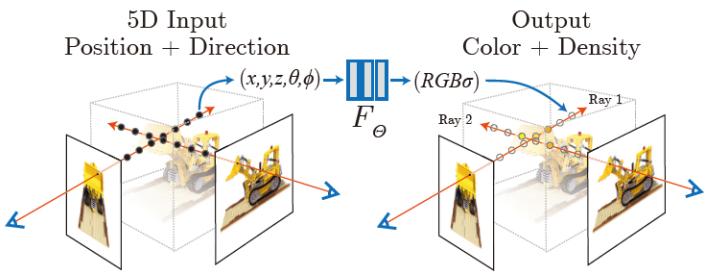
Implicit Fields

Graphics Vision & Video

Related Works

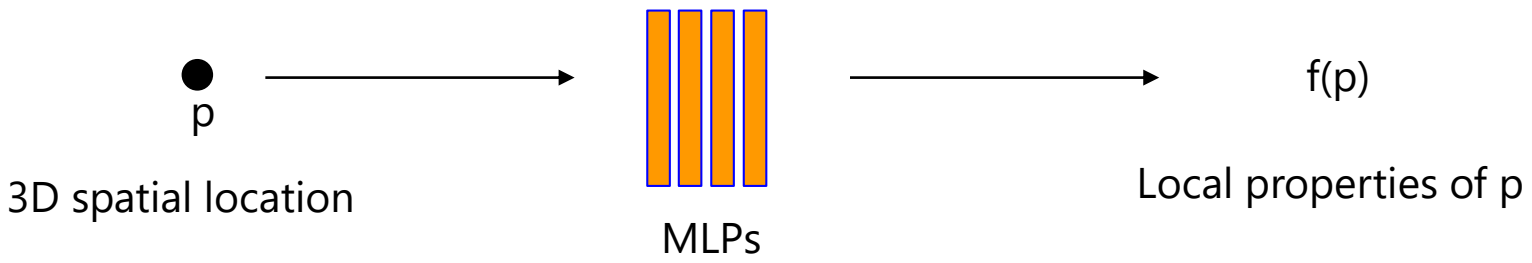


SRN [Sitzmann et al. 2019b]

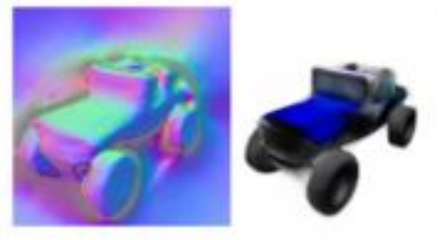


NeRF [Mildenhall et al. 2020]

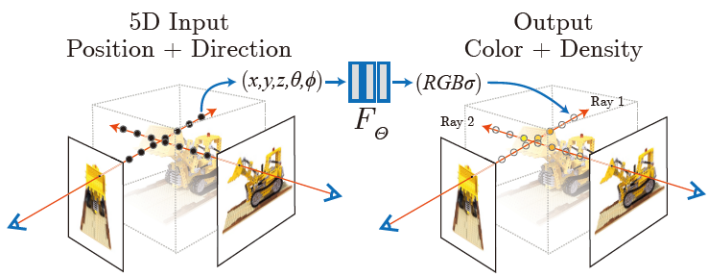
Implicit Fields



Related Works

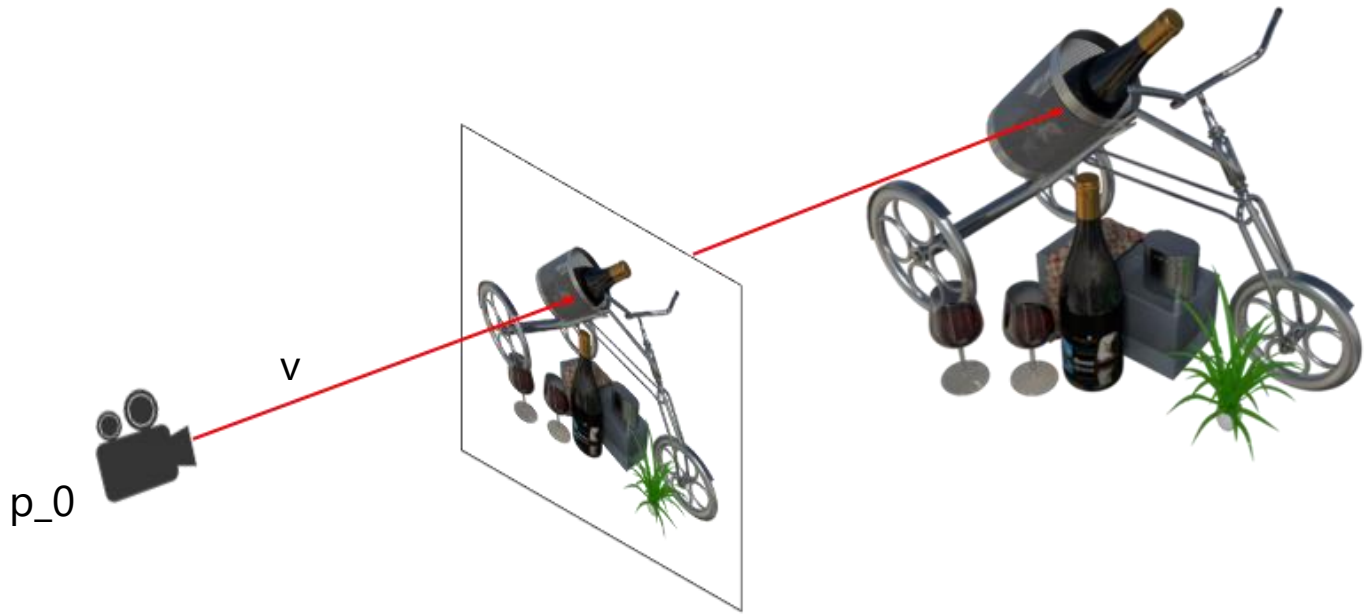


SRN [Sitzmann et al. 2019b]



NeRF [Mildenhall et al. 2020]

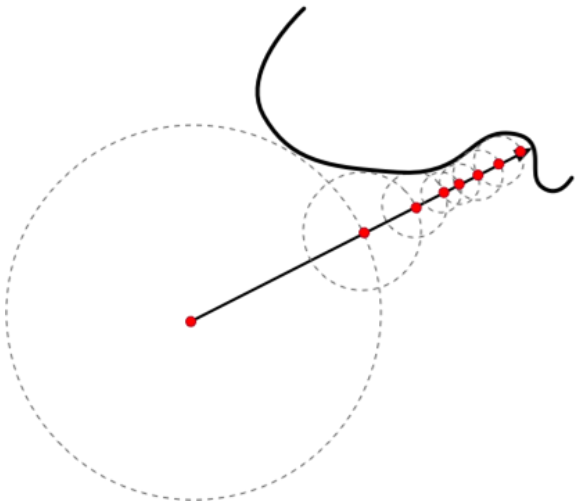
Implicit Fields



Neural Rendering with Implicit Fields

- Surface Rendering vs. Volume Rendering

Surface Rendering, e.g. SRN



Pros: *Fast Inference*
Cons: *Poor synthesis quality (Hard to find the geometry surface accurately)*

Results of SRN:

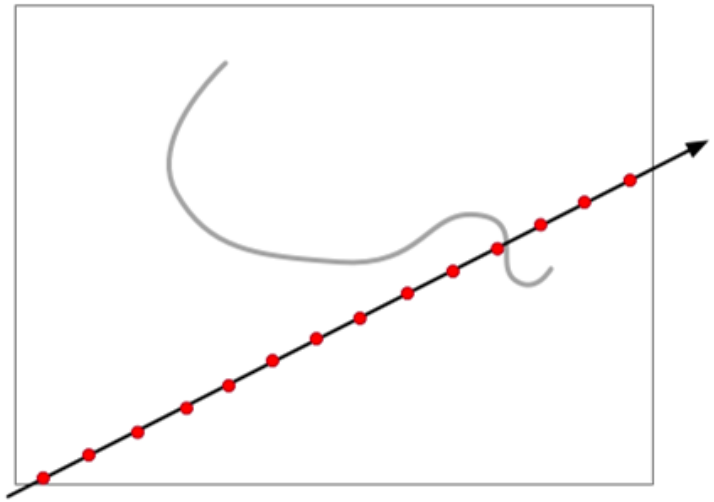
Speed: 4 s / frame
Quality:

- PSNR: 27.57
- SSIM: 0.908
- LPIPS: 0.134

Neural Rendering with Implicit Fields

- Surface Rendering vs. Volume Rendering

Volume Rendering, e.g. NeRF



Pros: *Good synthesis quality if the samples on the ray are dense enough.*
Cons: *Slow Inference*

Results of NeRF:

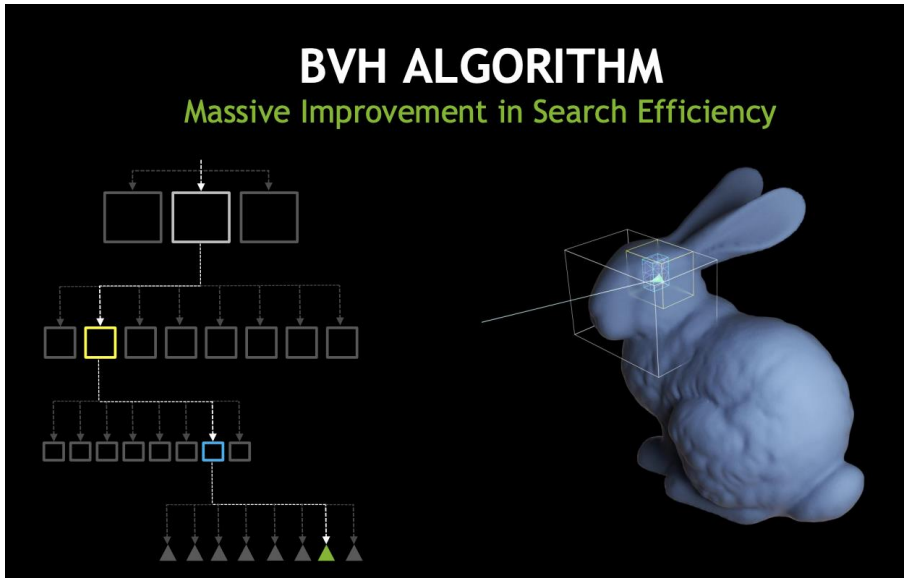
Speed: 100 s / frame
Quality:

- PSNR: 30.29
- SSIM: 0.932
- LPIPS: 0.111

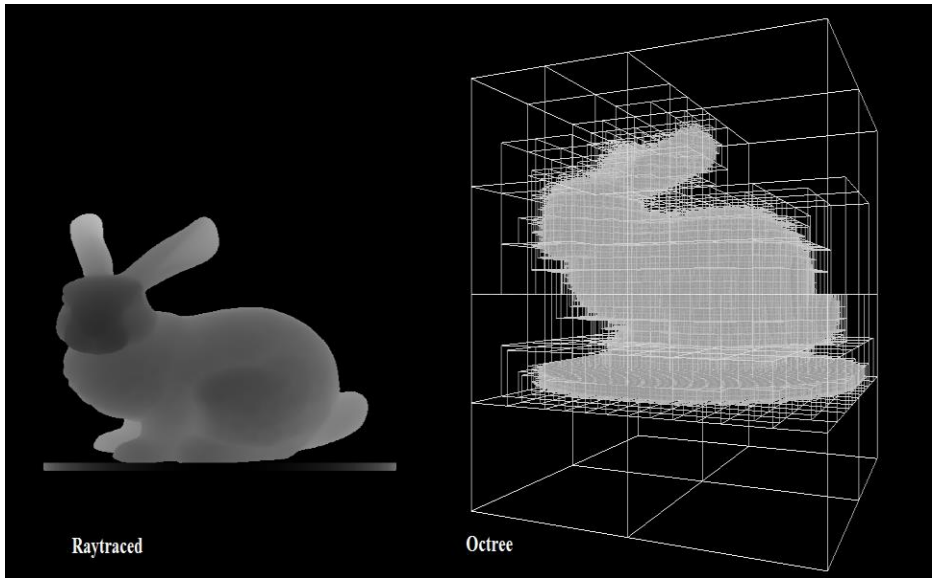
Neural Rendering with Implicit Fields

max planck institut informatik

It is important to prevent sampling of points in empty space without relevant scene content as much as possible.



Bounding Volume Hierarchy



Sparse Voxel Octree

Our Results

Speed: **2.62 s / frame**
v.s. 4s / frame (SRN)
v.s. 100s / frame (NeRF)

Quality:

- PSNR: 33.58
- SSIM: 0.954
- LPIPS: 0.098

Our Results

max planck institut informatik

Family

NeRF (Mildenhall et al. 2020)
(Rendering speed: 100 s/frame)

NSVF
(Rendering speed: 2.62 s/frame)

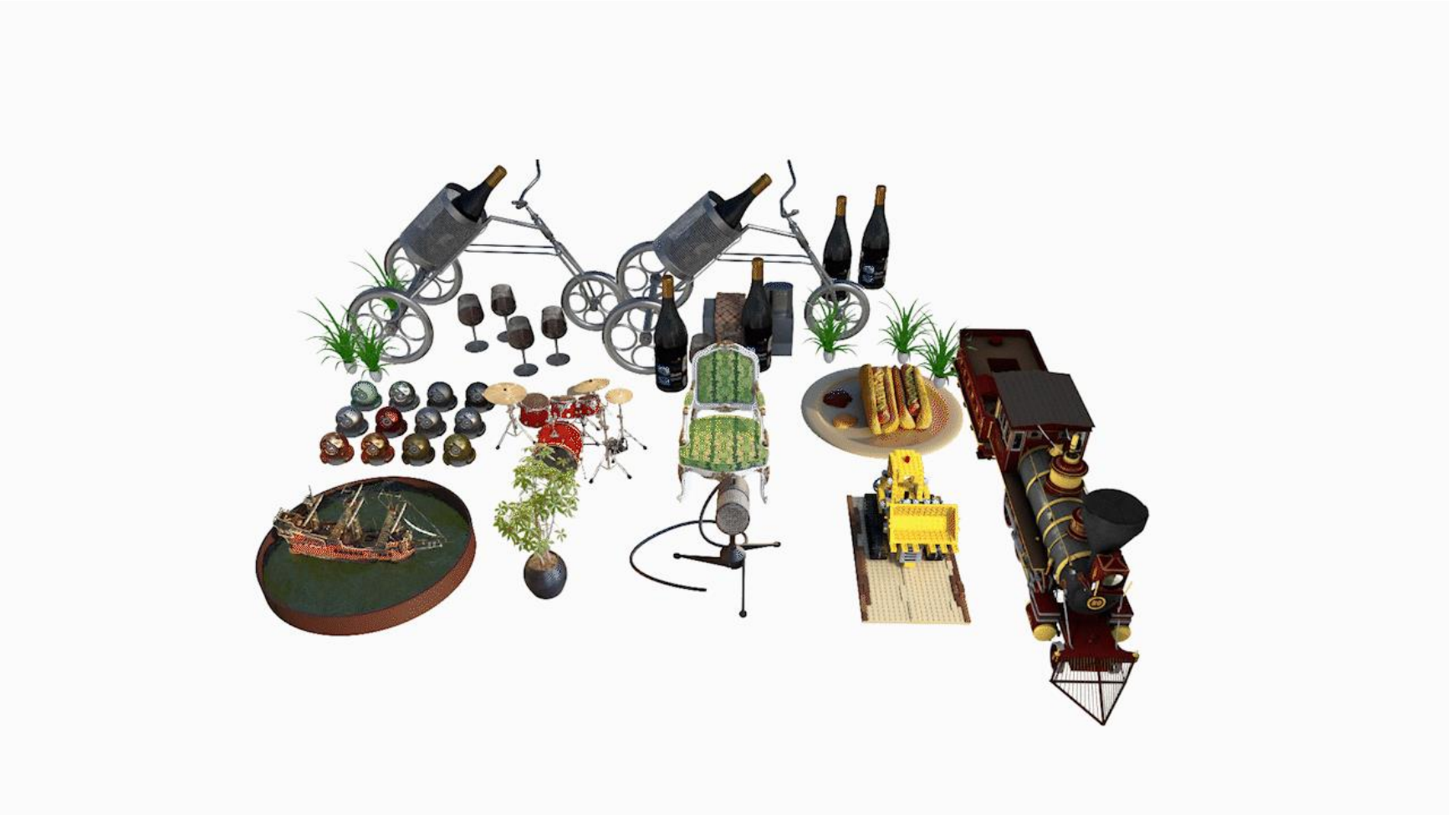
Speed: **2.62 s / frame**
v.s. 4s / frame (SRN)
v.s. 100s / frame (NeRF)

Quality:

- PSNR: 33.58
- SSIM: 0.954
- LPIPS: 0.098

Our Results

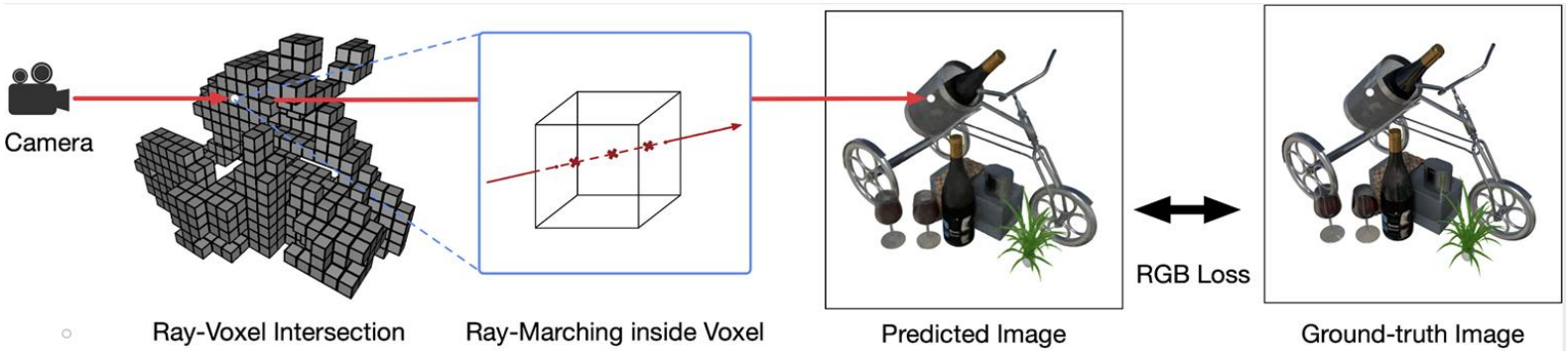
- Multi-object Training for Scene Editing and Scene Composition



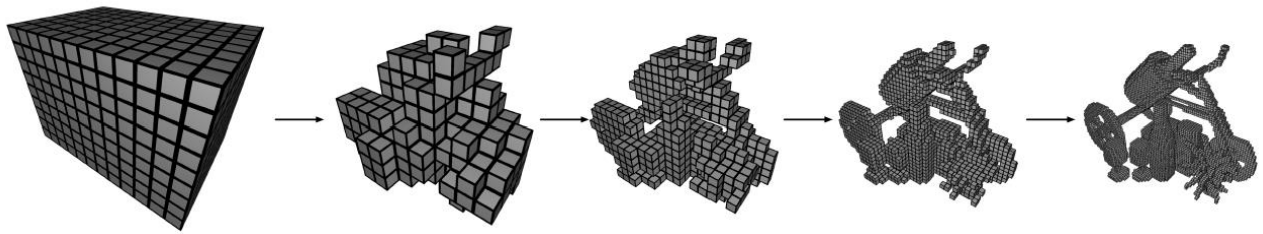
Our Method (NSVF)

Scene Representation - Neural Sparse Voxel Fields (NSVF): a hybrid neural representation for fast and high-quality free-viewpoint rendering.

Volume Rendering with NSVF



Progressive Learning: we learn NSVF with the differentiable volume rendering operation from a set of posed 2D images progressively



Scene Representation - NSVF

The relevant non-empty parts of a scene are contained within a set of sparse bounding voxels :

$$\mathcal{V} = \{V_1 \dots V_K\}$$

The scene is modeled as a set of voxel-bounded implicit functions:

$$F_\theta(\boldsymbol{p}, \boldsymbol{v}) = F_\theta^i(\boldsymbol{g}_i(\boldsymbol{p}), \boldsymbol{v}) \text{ if } \boldsymbol{p} \in V_i$$

spatial location ray direction

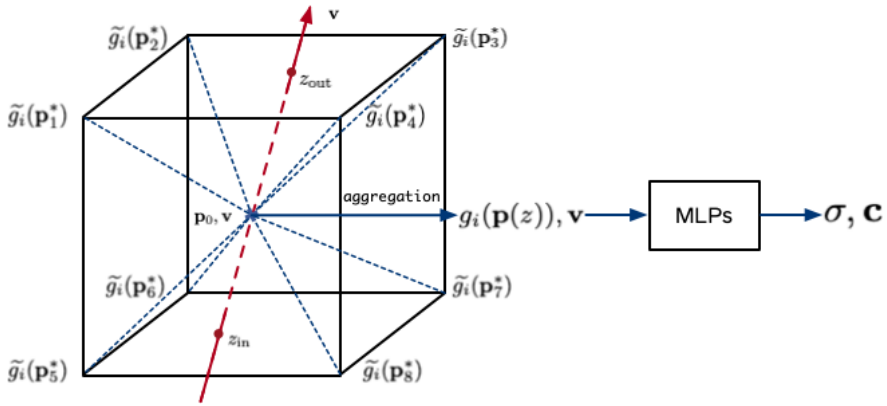
Scene Representation - NSVF

A voxel-bounded implicit field

- For a given point **p** inside voxel Vi, the voxel-bounded implicit field is defined as:

$$F_\theta^i : (\boldsymbol{g}_i(\boldsymbol{p}), \boldsymbol{v}) \rightarrow (\boldsymbol{c}, \sigma), \forall \boldsymbol{p} \in V_i,$$

voxel embedding ray direction color density



- Voxel embedding is defined as:

Trilinear interpolation *Voxel features (e.g. learnable voxel embeddings)*

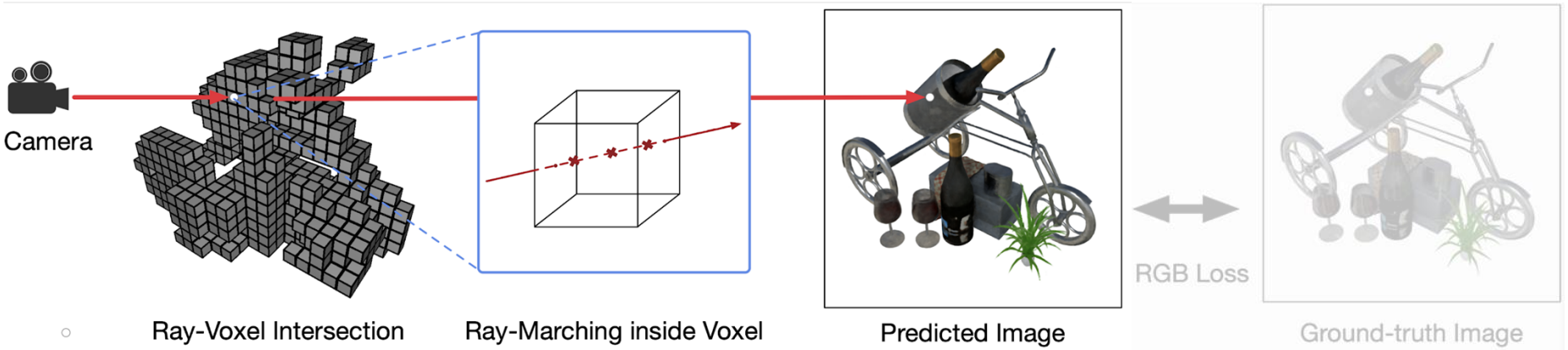
$$g_i(\boldsymbol{p}) = \zeta\left(\chi\left(\widetilde{g}_i(\boldsymbol{p}_1^*), \ldots, \widetilde{g}_i(\boldsymbol{p}_8^*)\right)\right)$$

Post-processing (e.g. Fourier features)

Volume Rendering with NSVF

Rendering NSVF is efficient as it prevents sampling points in the empty space

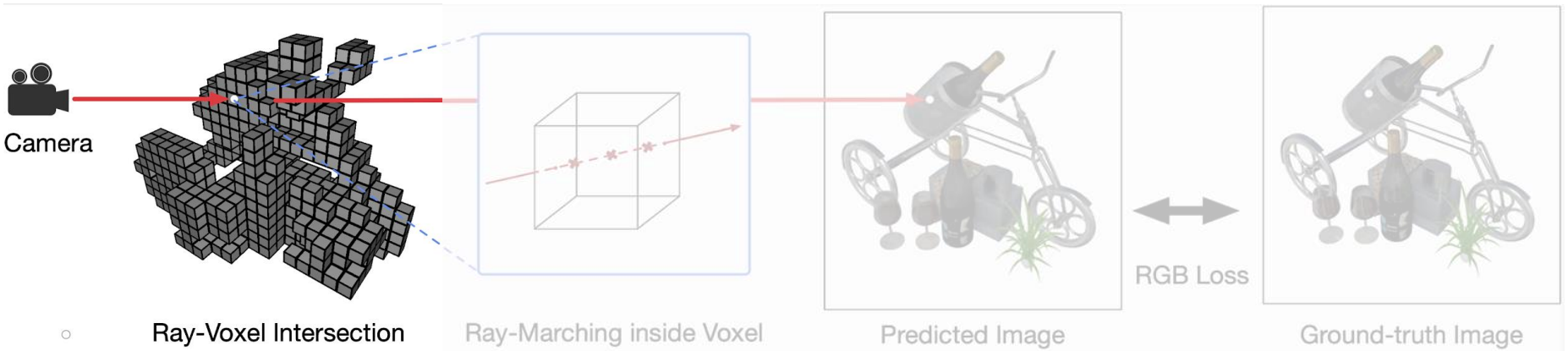
- Ray-voxel Intersection
- Ray-marching inside voxels



Volume Rendering with NSVF

Ray-voxel Intersection

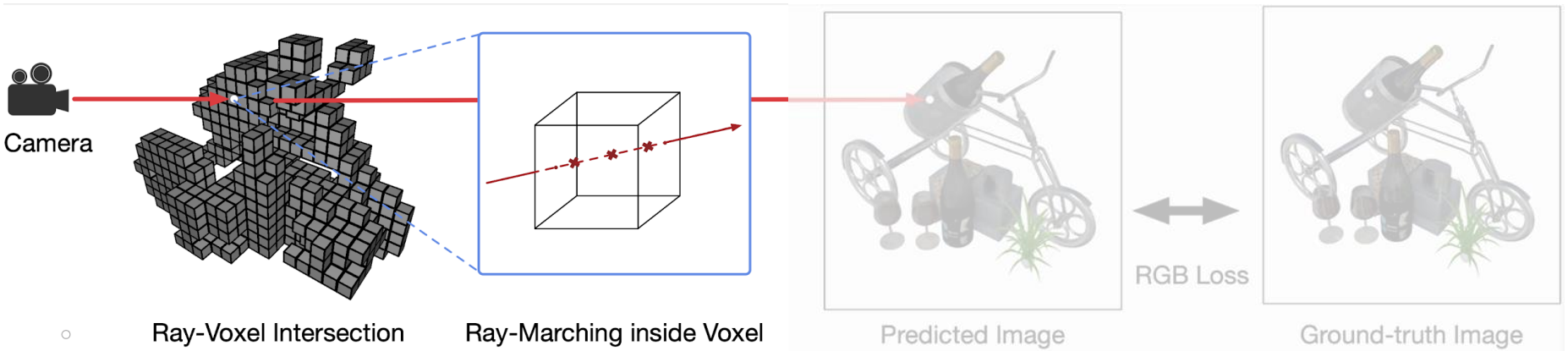
- Apply Axis Aligned Bounding Box (AABB) intersection test [Haines, 1989] for each ray.
- AABB is very efficient for NSVF. It can process millions of ray-voxel intersections in real time.



Volume Rendering with NSVF

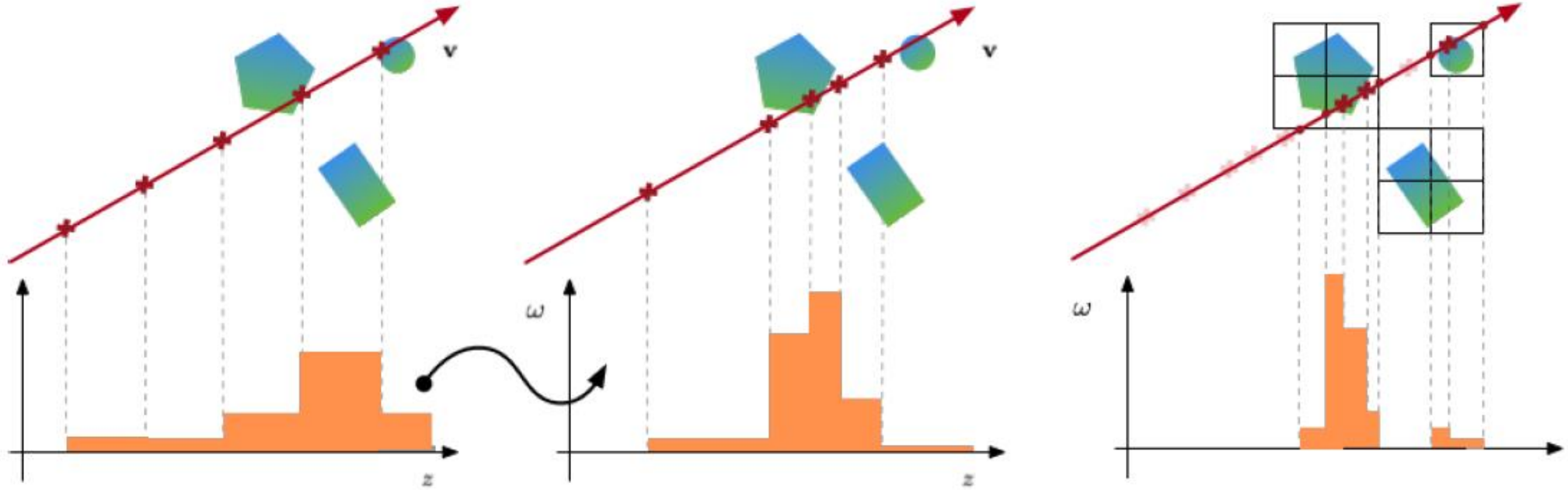
Ray Marching inside Voxels

- Uniformly sample points along the ray inside each intersected voxel, and evaluate NSVF to get the color and density of each sampled point.



Volume Rendering with NSVF

Comparison of Different Sampling Methods



(a) Uniform sampling in the whole space

(b) Importance sampling based on (a)'s result

(c) Sampling with sparse voxels

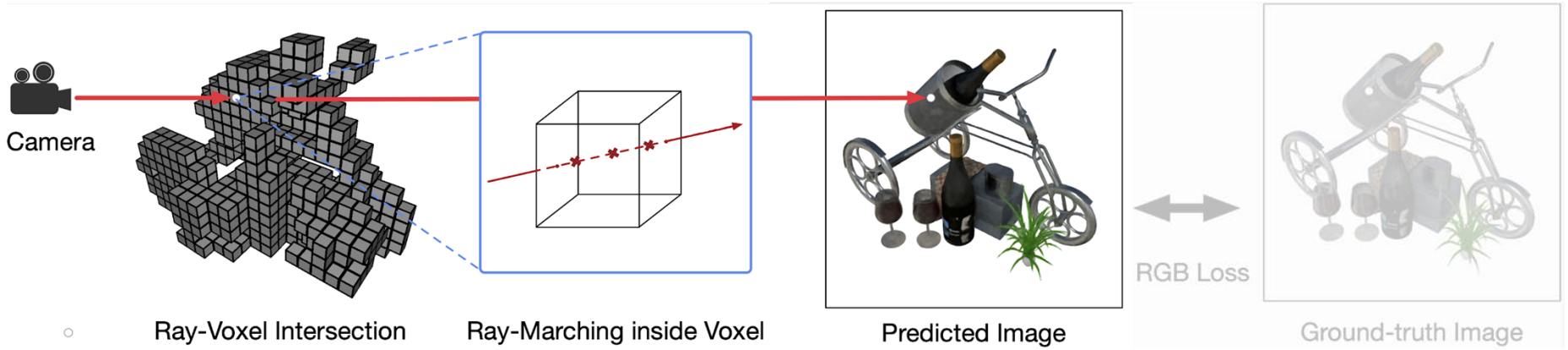
Volume Rendering with NSVF

- Rendering Algorithm

Algorithm 1: Neural Rendering with Sparse Voxel Fields

Input: camera $\boldsymbol{p}_0$, ray direction $\boldsymbol{v}$, step size τ, threshold ϵ, voxels $\mathcal{V} = \{V_1, \dots, V_K\}$, background $\boldsymbol{c}_{\text{bg}}, \theta$
Initialize: transparency $A = 1$, color $\boldsymbol{C} = \boldsymbol{0}$
Ray-voxel Intersection: Return all the intersections of the ray with k intersected voxels, sorted from near to far: $z^{\text{in}}_{t_1}, z^{\text{out}}_{t_1}, \dots, z^{\text{in}}_{t_k}, z^{\text{out}}_{t_k}$, where $\{t_1, \dots, t_k\} \subset \{1 \dots K\}, k < K$;
if $k > 0$ **then**
  Stratified sampling: $z_1, \dots, z_m$ with step size τ, where $z^{\text{in}}_{t_1} \le z_1$ and $z^{\text{out}}_{t_k} \ge z_m$;
  Include voxel boundaries: $\tilde{z}_1, \dots \tilde{z}_{2k+m} \leftarrow \text{sort}\left(z_1, \dots, z_m; z^{\text{in}}_{t_1}, z^{\text{out}}_{t_1}, \dots, z^{\text{in}}_{t_k}, z^{\text{out}}_{t_k}\right)$;
  for $j \leftarrow 1$ **to** $2k + m - 1$ **do**
    Obtain midpoints and intervals: $\hat{z}_j \leftarrow \frac{\tilde{z}_j + \tilde{z}_{j+1}}{2}, \Delta_j \leftarrow \tilde{z}_{j+1} - \tilde{z}_j$;
    if $A > \epsilon$ & $\Delta_j > 0$ & $\boldsymbol{p}(\hat{z}_j) \in V_i \exists i \in \{t_1, \dots, t_k\}$ **then**
      $\alpha \leftarrow \exp\left(-\sigma_\theta\left(g_i(\boldsymbol{p}(\hat{z}_j))\right) \cdot \Delta_j\right), \quad \boldsymbol{c} \leftarrow \boldsymbol{c}_\theta\left(g_i(\boldsymbol{p}(\hat{z}_j)), \boldsymbol{v}\right)$;
      $\boldsymbol{C} \leftarrow \boldsymbol{C} + A \cdot (1 - \alpha) \cdot \boldsymbol{c}, \quad A \leftarrow A \cdot \alpha$
$\boldsymbol{C} \leftarrow \boldsymbol{C} + A \cdot \boldsymbol{c}_{\text{bg}}$;
Return: $\boldsymbol{C}, A$

- Early Termination
  - Avoid taking unnecessary accumulation steps behind the surface;
  - Stop evaluating points earlier when the accumulated transparency **A** drops below a certain threshold ε.

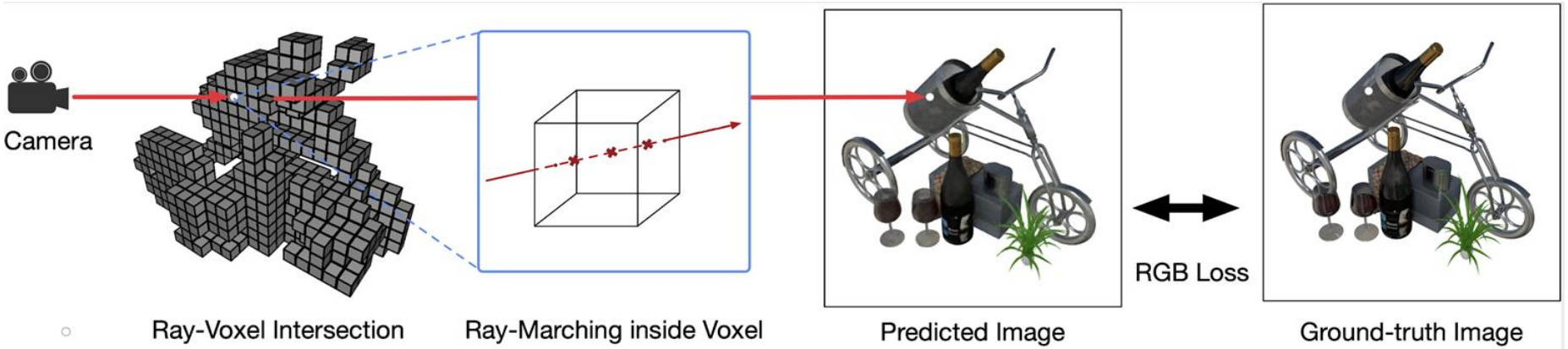


Progressive Learning

- Since our rendering process is differentiable, the model can be trained end-to-end with 2D posed images as input:

$$\mathcal{L} = \sum_{(\boldsymbol{p}_0, \boldsymbol{v}) \in R} \left\| \boldsymbol{C}(\boldsymbol{p}_0, \boldsymbol{v}) - \boldsymbol{C}^*(\boldsymbol{p}_0, \boldsymbol{v}) \right\|_2^2 + \lambda \cdot \Omega\left(A(\boldsymbol{p}_0, \boldsymbol{v})\right)$$

Beta-distribution regularization for transparency.



Progressive Learning

A progressive training strategy to learn NSVF from coarse to fine

- Voxel Initialization
- Self-Pruning
- Progressive Training

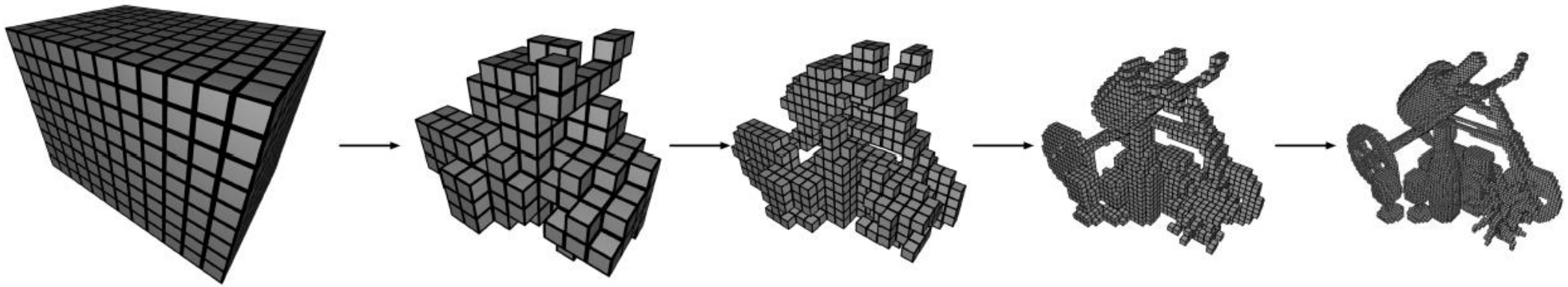
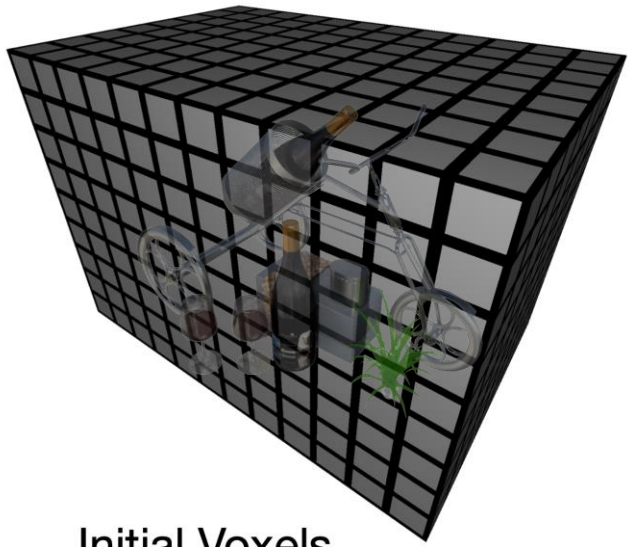


Illustration of self-pruning and progressive training

Progressive Learning

Voxel Initialization

- The initial bounding box roughly encloses the whole scene with sufficient margin. We subdivide the bounding box into ~1000 voxels.
- If a coarse geometry is available, the initial voxels can also be initialized by voxelizing the coarse geometry.



Initial Voxels

Progressive Learning

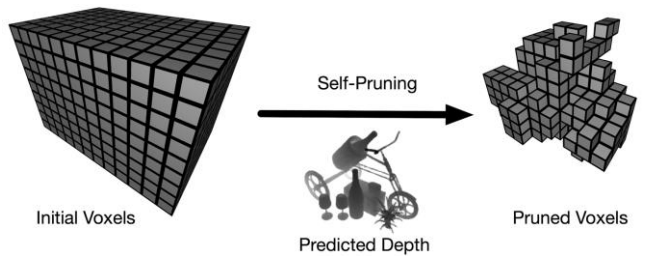
max planck institut informatik

Self-Pruning

- We can improve rendering efficiency by pruning “empty” voxels.
  - Determine whether a voxel is empty or not by checking the maximum predicted density from sampled points inside the voxel.

$$V_i \text{ is pruned } \textbf{if } \min_{j=1\ldots G} \exp(-\sigma(g_i(\boldsymbol{p}_j))) > \gamma, \quad \boldsymbol{p}_j \in V_i, V_i \in \mathcal{V},$$

density



  - Since this pruning process does not rely on other processing modules or input cues, we call it “*self-pruning*”.

Progressive Learning

Progressive Training

- Self-pruning enables us to progressively allocate our resources
- Progressive training:
  - Halve the size of voxels → Split each voxel into 8 sub-voxels
  - Halve the size of ray marching steps
  - The feature representations of the new vertices are initialized via trilinear interpolation of feature representations at the original eight voxel vertices.

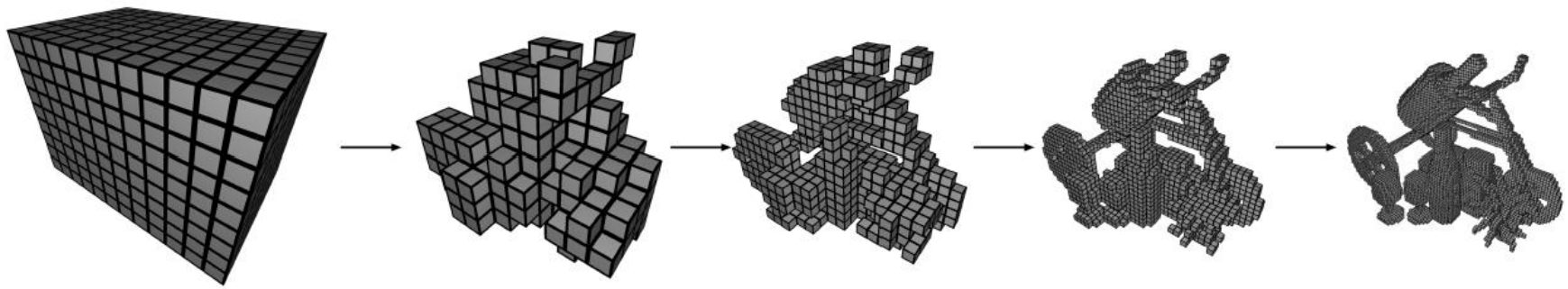


Illustration of self-pruning and progressive training

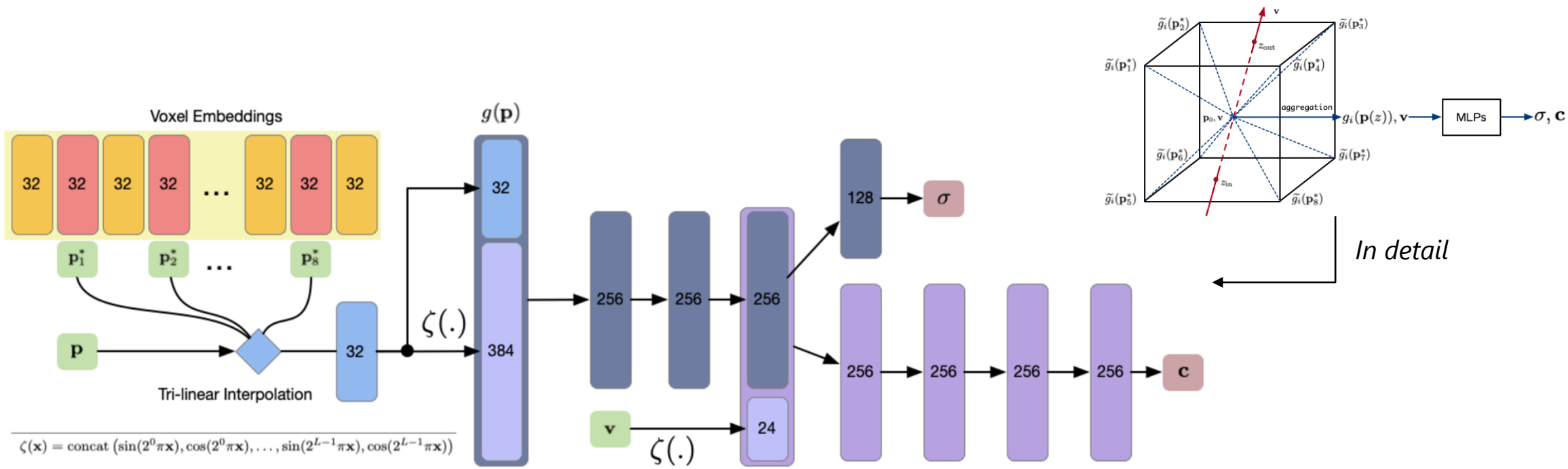
Experimental Settings

- Datasets
  - Synthetic-NeRF
  - Synthetic-NSVF
  - BlendedMVS
  - Tanks & Temple *Real dataset*
  - ScanNet *Large indoor scenes*
  - Maria Sequence *Dynamic sequence of human body*

- Baselines
  - Scene Representation Networks (SRN) [Sitzmann et al. 2019]
  - Neural Volumes (NV) [Lombardi et al. 2019]
  - Neural Radiance Fields (NeRF) [Mildenhall et al. 2020]

Experimental Settings

- Network Architecture
  - In our experiments, we use Fourier transformation as the post-processing function in $g_i(\boldsymbol{p}) = \zeta\left(\chi\left(\widetilde{g}_i(\boldsymbol{p}_1^*), \ldots, \widetilde{g}_i(\boldsymbol{p}_8^*)\right)\right)$, and set maximum frequency L = 6.



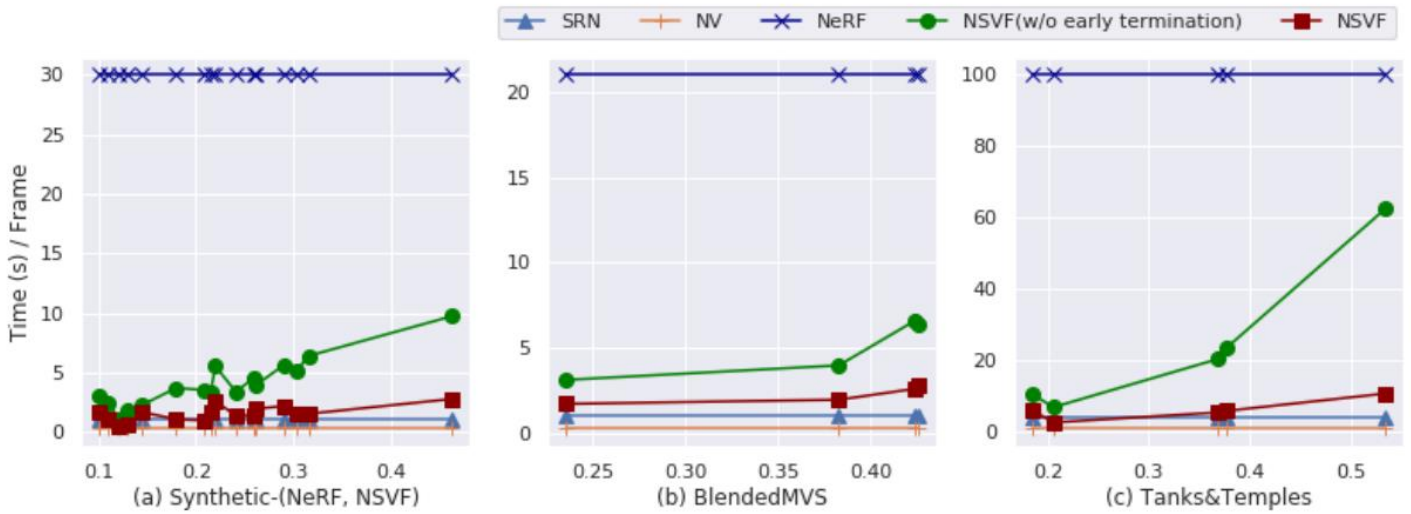
Experimental Settings

- Training
  - 32 images/batch, 2048 rays/image;
  - 8 Nvidia V100 GPUs for 150K updates (~2 days);
  - Perform self-pruning every 2.5K iterations;
  - Progressive training: halve the voxel size and step size at 5K, 25K and 75K iterations.

- Inference
  - Early termination: we set the threshold ε as 0.01 for all the scenes;
  - We evaluate on a single V100 GPU at inference time.

Quantitative Results

Models	Synthetic-NeRF			Synthetic-NSVF			BlendedMVS			Tanks and Temples		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN	22.26	0.846	0.170	24.33	0.882	0.141	20.51	0.770	0.294	24.10	0.847	0.251
NV	26.05	0.893	0.160	25.83	0.892	0.124	23.03	0.793	0.243	23.70	0.834	0.260
NeRF	31.01	0.947	0.081	30.81	0.952	0.043	24.15	0.828	0.192	25.78	0.864	0.198
$NSVF^0$	**31.75**	**0.954**	0.048	**35.18**	**0.979**	**0.015**	26.89	**0.898**	0.114	**28.48**	**0.901**	0.155
NSVF	31.74	0.953	**0.047**	35.13	**0.979**	**0.015**	**26.90**	**0.898**	**0.113**	28.40	0.900	**0.153**



More Results: Synthetic Dataset

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

Ours
(Rendering speed: 1.68 s/frame)

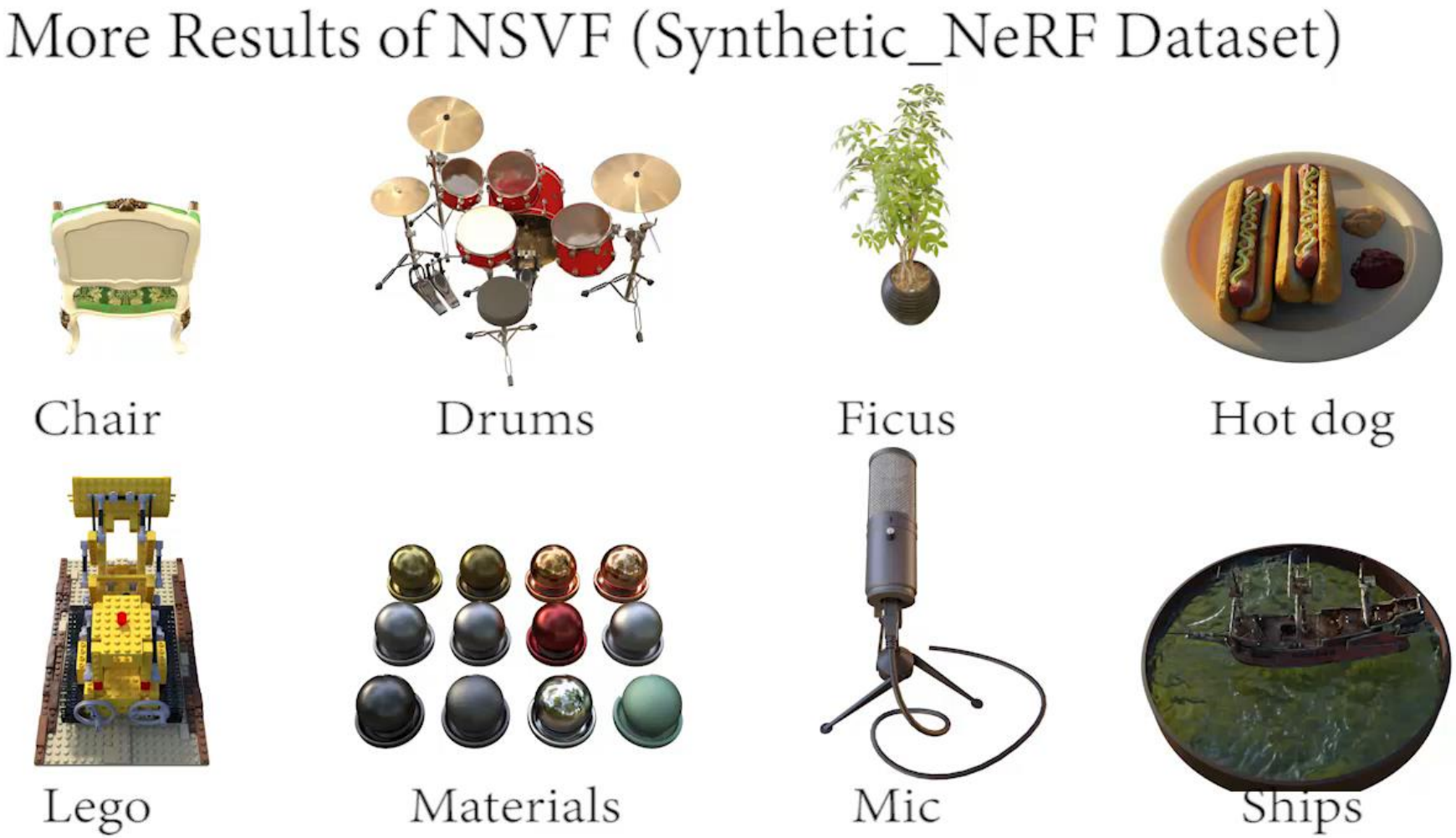
More Results: Synthetic Dataset

Robot

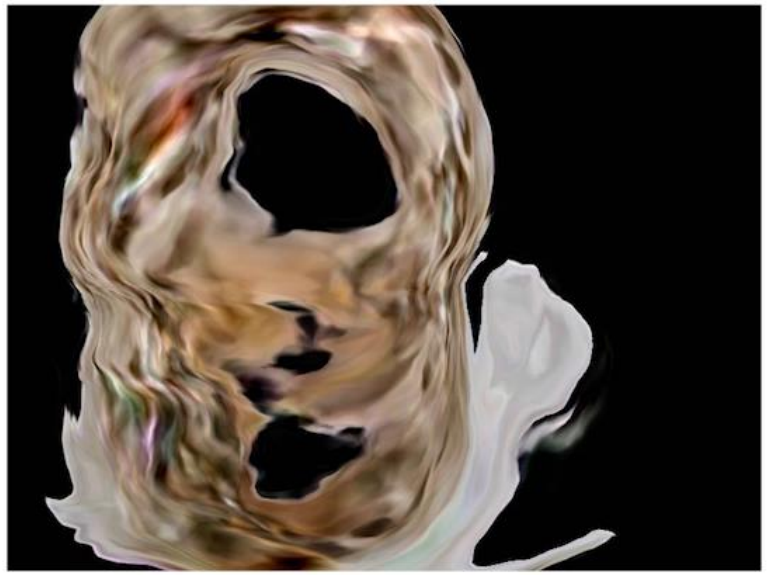
SRN (Sitzmann et al. 2019)
(Rendering speed: 1.10 s/frame)

NSVF
(Rendering speed: 0.6 s/frame)

More Results: Synthetic Dataset



More Results: BlendedMVS Dataset



SRN (Sitzmann et al. 2019)
(Rendering speed: 0.90 s/frame)

Ours
(Rendering speed: 2.15 s/frame)

More Results: BlendedMVS Dataset

Character

SRN (Sitzmann et al. 2019)
(Rendering speed: 0.90 s/frame)

Ours
(Rendering speed: 1.43 s/frame)

More Results: BlendedMVS Dataset

More Results: Real Dataset (Tanks and Temples)

max planck institut informatik

Truck

SRN (Sitzmann et al. 2019)
(Rendering speed: 4 s/frame)

Ours
(Rendering speed: 5.88 s/frame)

More Results: Real Dataset (Tanks and Temples)

More Result: Zoom-in & Zoom-out

More Results: Dynamic Scene

We use a hypernetwork to encode all the 200 frames of the Maria sequence.

Normals of NSVF result

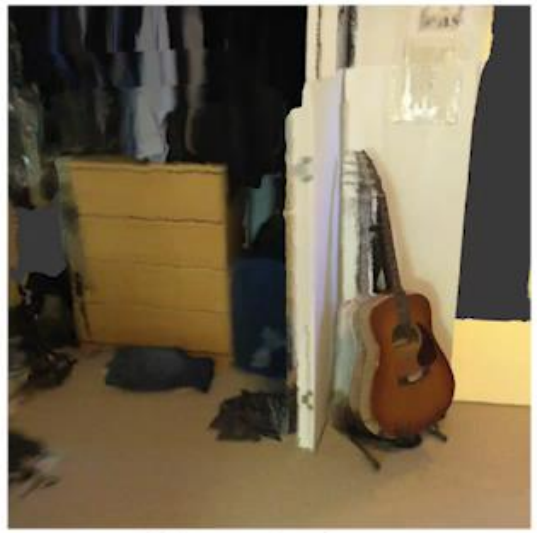
NSVF

(Input sequence from Fraunhofer Heinrich Hertz Institute)

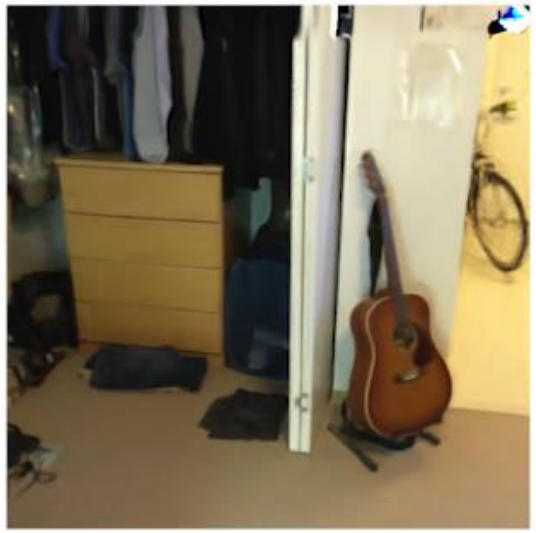
More Results: Large-scale Indoor Scene

We interactively control the camera and synthesize views with camera poses.
We also show the rendered 3D textured mesh as a comparison.
Here the NSVF results are more complete and realistic.

Interactive camera control

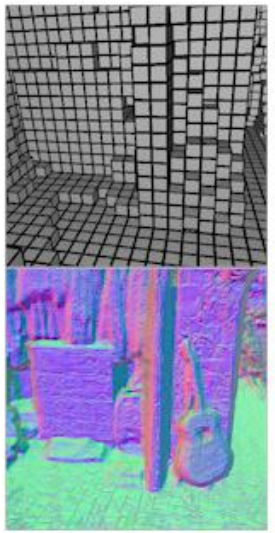


Users' view (Rendered mesh)



NSVF

Sparse voxels



Normals

(Input sequence and 3D mesh from ScanNet [Dai et al. 2017])

More Results: Scene Editing and Composition

We train a single model for 10 synthetic scenes and use the learnt multi-scene model to compose more complex scenes by interactively editing the sparse voxels.

Interactive editing

NSVF

Limitations and Future Work

Handling Complex Background

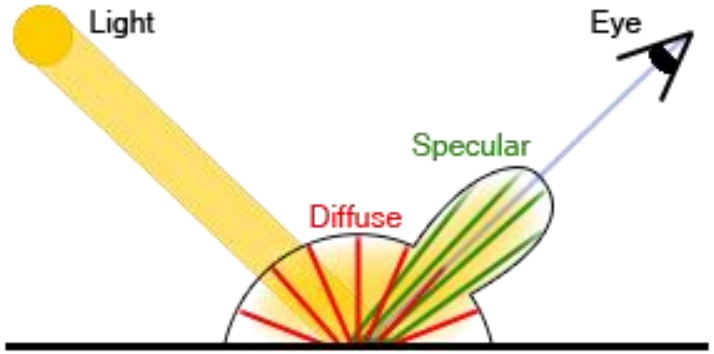
- Our current model cannot handle complex backgrounds; We need to manually mask foreground in the image, which is not feasible for real applications.

- Can we model the complex background and the foreground object jointly so to be able to synthesize foreground as well as background?

Limitations and Future Work

Modeling Lighting Effects

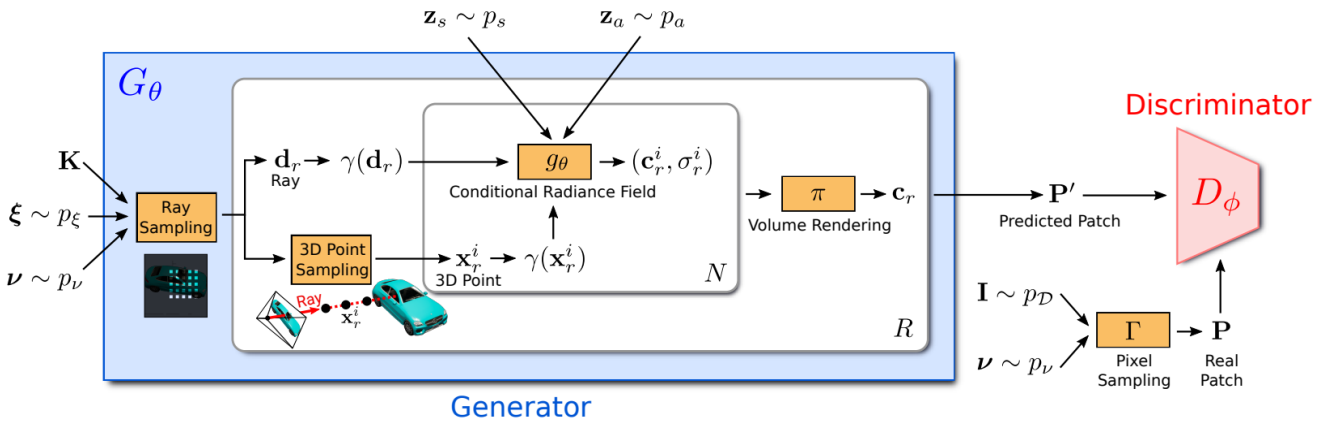
- Our model only models view-dependent color but does not model different lighting components, such as albedo, diffusion and specular, which may lead to the following issues:
  - Hard to recover complex lighting effects;
  - It is impossible to do re-lighting.



- One potential solution is to separately model each component.
  - Can we decompose the lighting effects in an unsupervised way?

Limitations and Future Work

- Simultaneous Camera Motion Estimation and Neural Rendering
  - Our approach requires multi-view images and their corresponding camera parameters as input .
  - Is it possible to simultaneously learn the camera parameters and scene representations? In real applications, it is common to have a large number of images without camera pose information.



Schwarz et al. *"GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis." Arxiv 2020.*

Limitations and Future Work

- Neural Rendering for Humans
  - Our method can simply use a hypernetwork to render dynamic scenes, such as moving humans; however the synthesis quality would degrade when a large number (e.g. 1k) of video frames need to be encoded into a single hypernetwork. We should seek a more efficient way to encode dynamic scenes.
  - Add explicit controls on the NSVF results to achieve human motion reenactment.

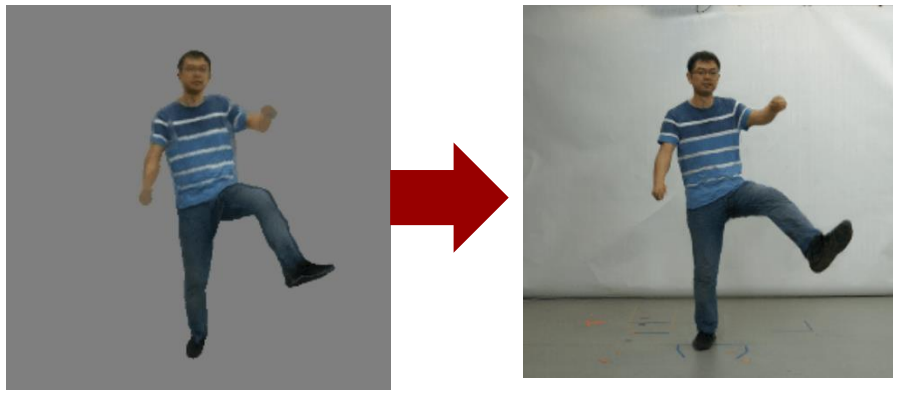


Image from [Liu et al. 2020]

Thank You!

max planck institut informatik

Neural Sparse Voxel Fields

Lingjie Liu*, Jiatao Gu*, Kyaw Zaw Lin, Tat-Seng Chua, Christian Theobalt

Paper link: https://arxiv.org/pdf/2007.11571.pdf

Video link:
https://www.youtube.com/watch?v=RFqPwH7QFEI&list=PLCAViLbA8Ml6KXzGTENfELX8wcPiXWVT8

Graphics Vision & Video