Differentiable Rendering And Implicit Functions

Speaker: Angtian Wang

Rendering

Rendering or image synthesis is the process of generating a photorealistic image from 3D model by means of a computer program [1].

- Explicit geometry, e.g., Meshes, Pointcloud, Gaussians Kernels
- Implicit geometry, e.g., NeRF, Signed Distance Functions(SDF)

[1] *Wikipedia:* Rendering: https://en.wikipedia.org/wiki/Rendering_(computer_graphics)

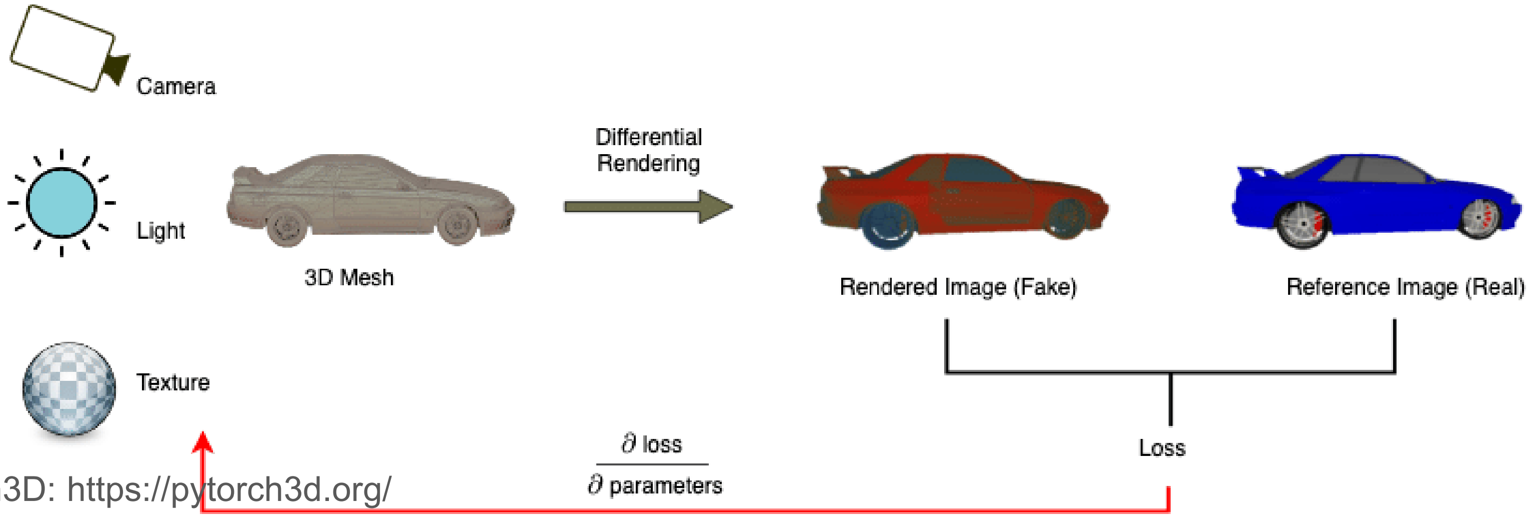
Surface Models

Light is reflected off surfaces to generate an image. These are classic rendering techniques, dating back to the 18th century (Lambert’s Law).

Differentiable Rendering

Differentiable rendering bridges the gap between 2D and 3D by allowing 2D image pixels to be related back to 3D properties of a scene. [1] R(.) is a differentiable function.

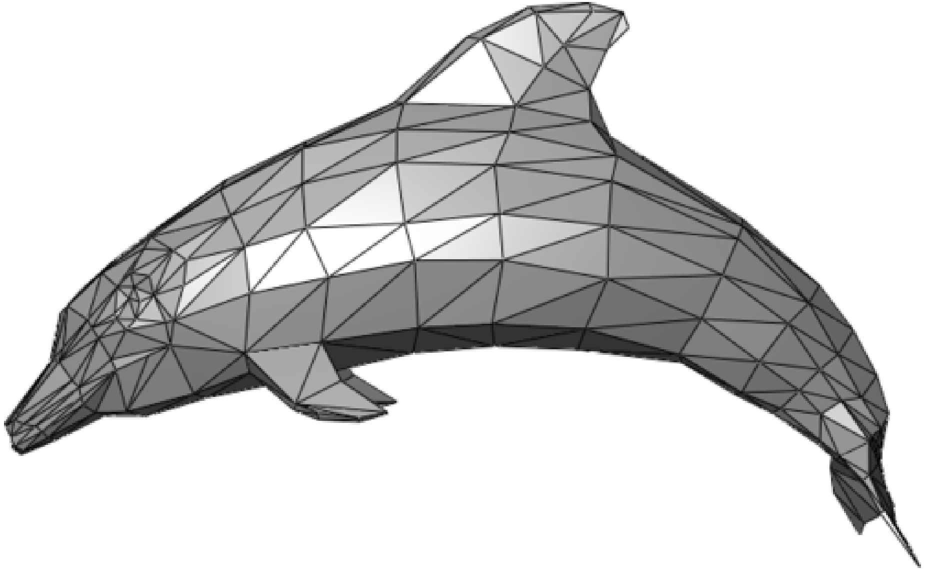
$$I(\text{Pixel}) = R\left(\boxed{\text{Vertex, Faces, Texture, Camera}}\right)$$



[1] PyTorch3D: https://pytorch3d.org/

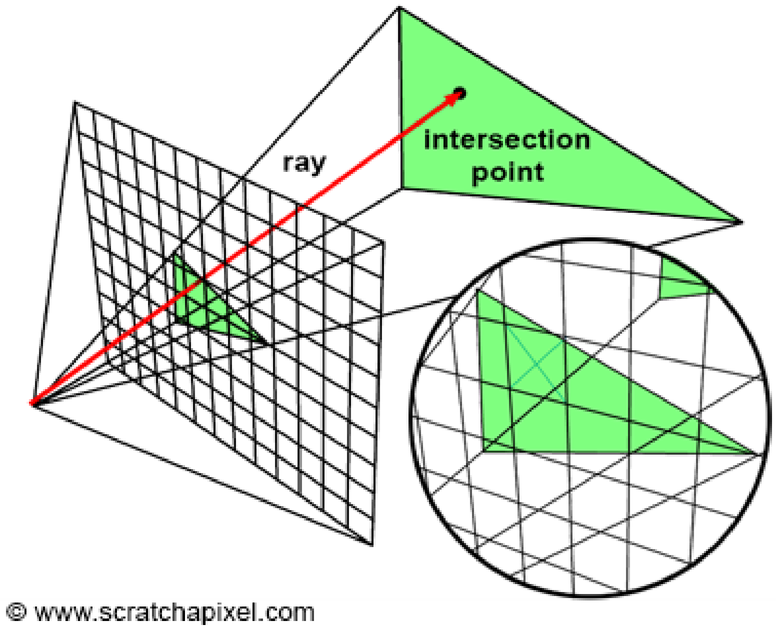
CAD models represent an object by a Polygon Mesh.

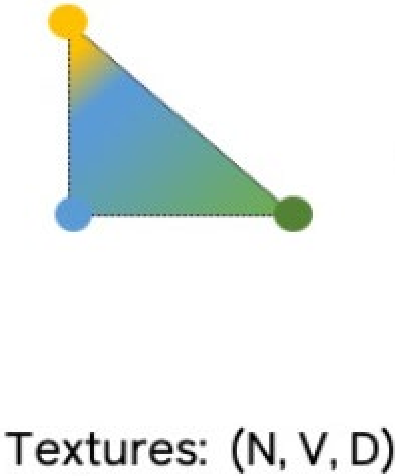
- Polygon mesh is a collection of vertices, edges and faces. The simplest is triangles.



Mesh Rasterization

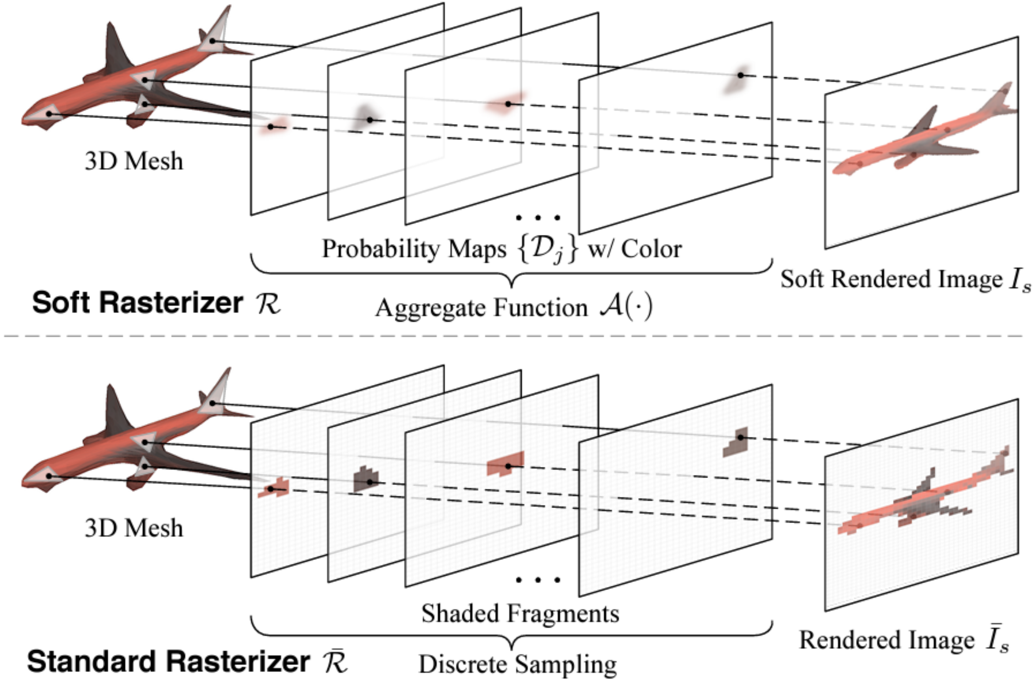
- Rasterization determine the visible primitives on each image pixel.
- Predict properties at the 3D vertices and interpolate between them.





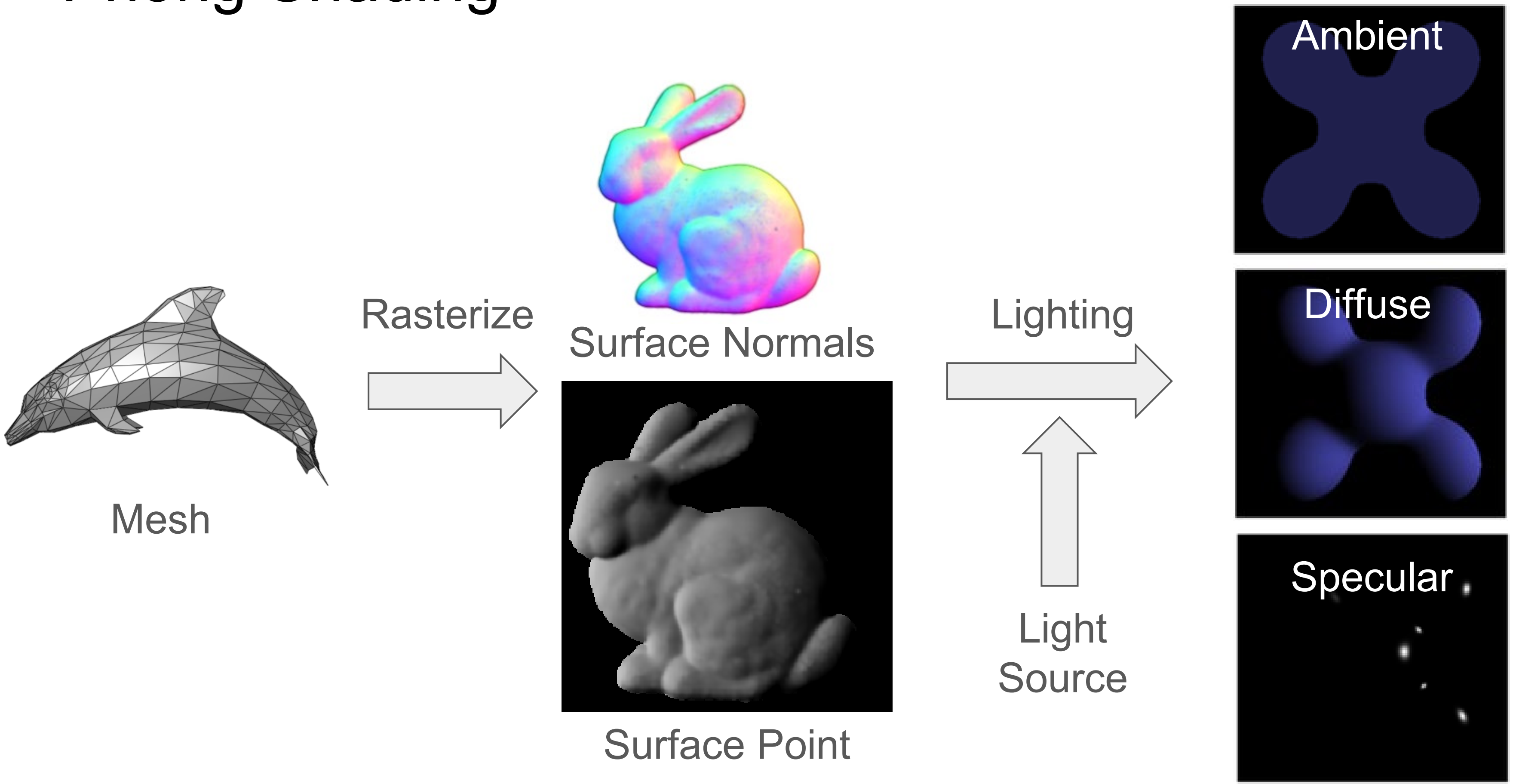
Differentiable Mesh Rasterization

- Soften boundary of triangles.
- Blend triangles based on the distance.



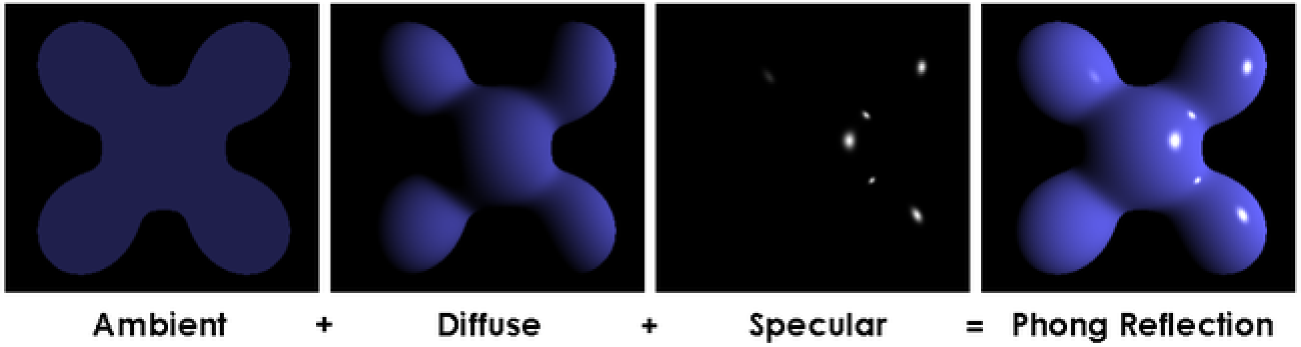
Liu, S., Li, T., Chen, W., & Li, H. (2019). Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 7708-7717).

Phong Shading



Phong, B. T. (1998). Illumination for computer generated pictures. In *Seminal graphics: pioneering efforts that shaped the field* (pp. 95-101).

Phong Shading



$$I_p = k_a i_a + \sum_{m \in \text{lights}} (k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,s}).$$

$\hat{L}_m$ surface point toward light source

$\hat{N}$ surface normal

$\hat{R}_m$ reflected ray

$\hat{V}$ viewing direction

i_a ambient color on pixel p

$i_{m,d}$ diffuse color on pixel p

$i_{m,s}$ specular color on pixel p

m light sources

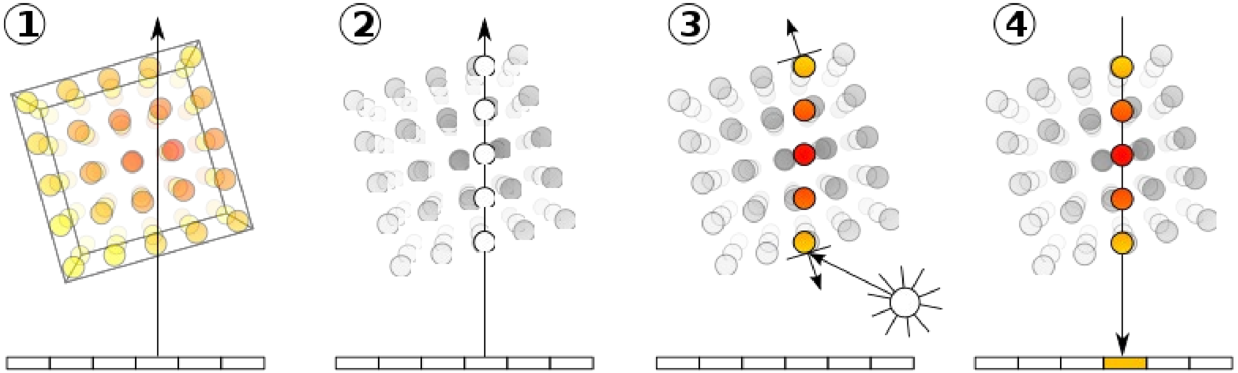
k_s k_d k_a α constants (specular, diffuse, ambient reflection), (shininess)

Phong, B. T. (1998). Illumination for computer generated pictures. In *Seminal graphics: pioneering efforts that shaped the field* (pp. 95-101).

Volume Rendering

This is a learning based approach which represents objects by a volume and not by surfaces. This became popular because researchers showed that this approach can generate very realistic images if it trained from data.

Ray Tracing: Perspective projection



- Considering for each pixel, we cast a viewing ray in the 3D scene. Then instead of project each primitives onto the image plane, we trace primitives that interacted with the viewing rays.

Ray Tracing

- The viewing ray for each pixel is given by

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

The viewing direction ***d*** in a Perspective Camera system can be computed as

$$\mathbf{d} = \mathbf{R}^{-1} \cdot \begin{bmatrix} \frac{u-p_x}{f} \\ \frac{v-p_y}{f} \\ 1 \end{bmatrix}$$

o is given by the camera location
R is the rotation matrix of the camera
p_x, p_y are the camera principal point
u, v is a pixel on the image
f is the focal length

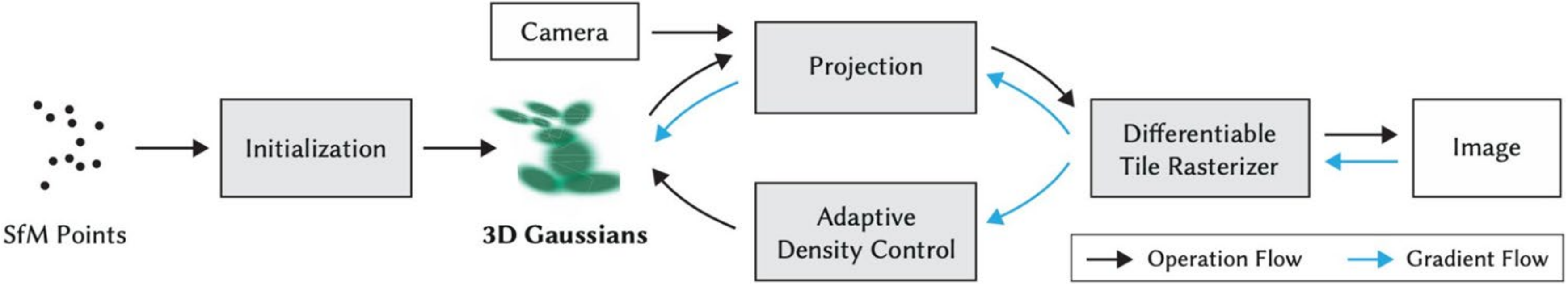
Gaussian Splatting

- We have many images corresponding to views of a scene from different (known) angles. We want to predict the image from a new view.
- This is a learning based approach. The object is modeled by a set of Gaussians, each has a position (mean), a covariance to specify its size, its color coefficients, and its opacity.
- The images are formed by rules in terms of the quantities of these Gaussians.
- The model is trained using a set of images of the object/scene from known viewpoints.
- Then the model learns the quantities of the Gaussians and can generate an image from novel viewpoints.
- It predicts the intensity $C_p(r)$, where r is the viewing rays.

Gaussian Splatting

- Represent objects 3D gaussians.
- Trained with multi-view images with known viewpoints (30~200)

$$\mathcal{G}(\mathbf{x}-\mu) = \frac{1}{2\pi\Sigma(\mathbf{x})^{\frac{1}{2}}}\exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu))$$



Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, *42*(4), 1-14.

Gaussian Splatting

- Assign an intensity function to each Gaussian (i)– next slide.
- Blend the projected gaussians via transmittance.

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

C color on pixel p

c_i color for each gaussian, specified by coefficients of spherical harmonics (next slide).

$$\alpha_i = (1 - \exp(-\sigma_i \delta_i))$$

σ_i is computed by projection the i-th gaussian onto the image pixels, where the projection to pixels p in the image.

δ_i is the intervals on viewing ray.

Gaussian Splatting

- For each Gaussian the Color is expressed by Spherical Harmonics with learnt coefficients.

$$\mathbf{c}_i = \sum_{l=0}^{L} \sum_{m=-l}^{l} k_l^m y_l^m(\theta, \varphi)$$

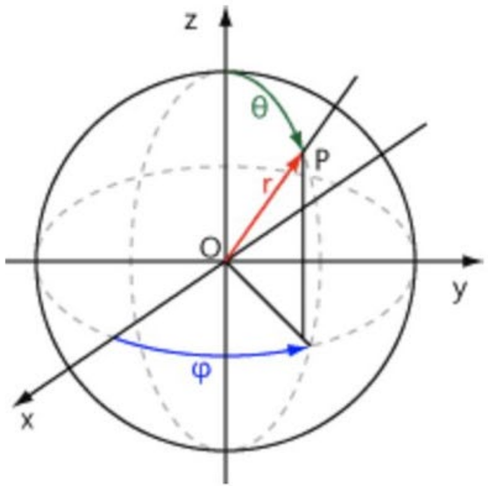
k_l^m are learnable coefficients.

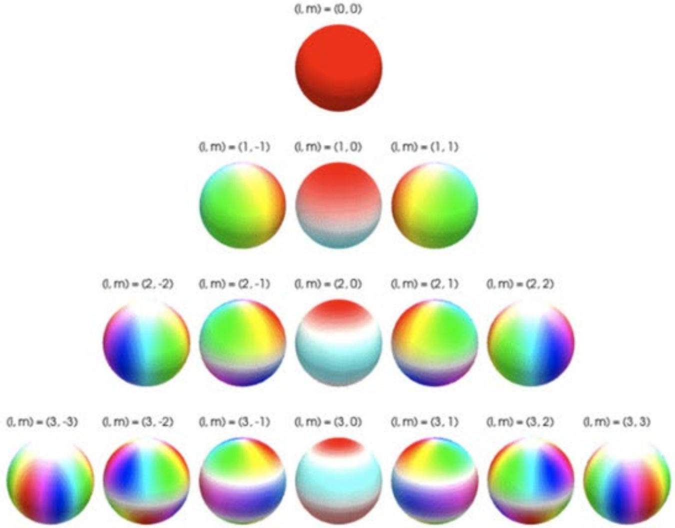
$$y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2} K_l^m \cos(m\varphi) P_l^m(\cos\theta), & m > 0 \\ \sqrt{2} K_l^m \sin(-m\varphi) P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

Green, R. (2003, March). Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference* (Vol. 56, p. 4).

Gaussian Splatting

- Color: Spherical Harmonics.

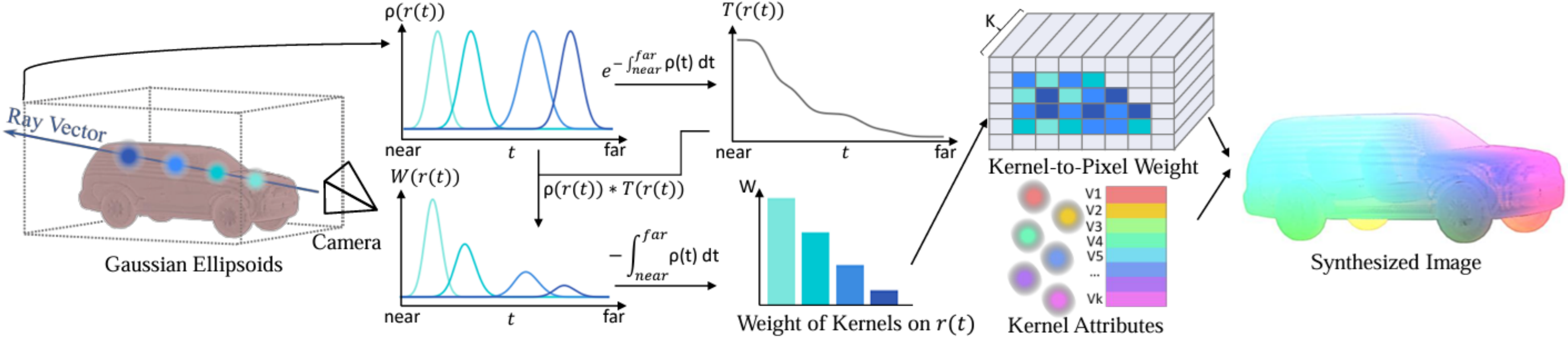




Green, R. (2003, March). Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference* (Vol. 56, p. 4).

Gaussian Splatting

VoGE: Volume Gaussian Ellipsoids



VoGE uses the same idea that represent object using 3D gaussians, but renders in a ray tracing manner instead of rasterization (without assuming Gaussians are one in front of another).

Wang, A., Wang, P., Sun, J., Kortylewski, A., & Yuille, A. (2022, September). VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis. In *The Eleventh International Conference on Learning Representations*.

VoGE: Volume Gaussian Ellipsoids

VoGE represents object using 3D gaussians centered on the vertices of a CAD model:

$$\rho(\mathbf{X}) = \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi \cdot ||\mathbf{\Sigma}_k||_2}} e^{-\frac{1}{2}(\mathbf{X}-\mathbf{M}_k)^T \cdot \mathbf{\Sigma}_k^{-1} \cdot (\mathbf{X}-\mathbf{M}_k)}$$

$\mathbf{X}$ is a location in the 3D scene
$\mathbf{M}_k$ is the center for gaussian k
$\mathbf{\Sigma}_k$ is the covariance for gaussian k

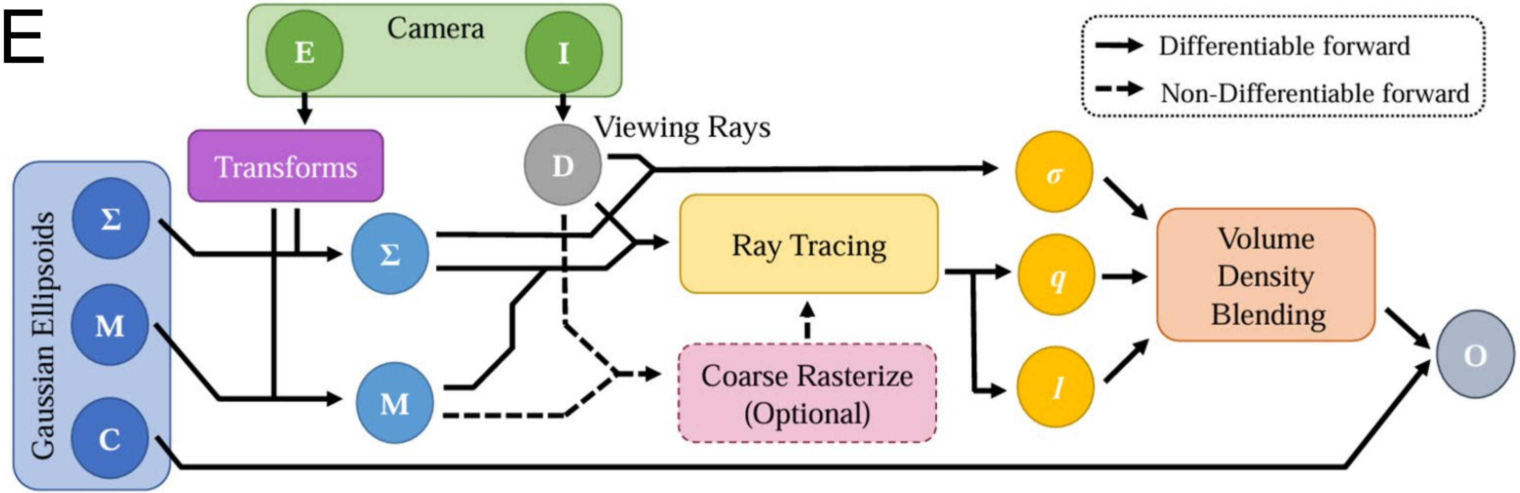
Also, the viewing ray on each pixel is:

$$\mathbf{r}(t) = \mathbf{D} * t$$

$\mathbf{D}$ Is the viewing direction each pixel p of ray $\mathbf{r}(t)$

Wang, A., Wang, P., Sun, J., Kortylewski, A., & Yuille, A. (2022, September). VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis. In *The Eleventh International Conference on Learning Representations*.

VoGE



Instead of project the 3D gaussians onto 2D image plane. We compute the intersection of each 3D gaussian with each viewing ray, which is a 1D gaussian along the ray direction.

$\sigma_m \; l_m \; q_m$ are the parameter for the 1D gaussian on ray $\mathbf{r}(s)$

$$l_m = \frac{\mathbf{M}_m^T \cdot \mathbf{\Sigma}_m^{-1} \cdot \mathbf{D} + \mathbf{D}^T \cdot \mathbf{\Sigma}_m^{-1} \cdot \mathbf{M}_m}{2 \cdot \mathbf{D}^T \cdot \mathbf{\Sigma}_m^{-1} \cdot \mathbf{D}}$$

$$q_m = -\frac{1}{2} \mathbf{V}_m^T \cdot \mathbf{\Sigma}_m^{-1} \cdot \mathbf{V}_m$$

$$\frac{1}{\sigma_m^2} = \mathbf{D}^T \cdot \mathbf{\Sigma}_m^{-1} \cdot \mathbf{D}$$

VoGE: Volume Gaussian Ellipsoids

$$C(\mathbf{r}) = \int_{-\infty}^{\infty} T(t)\rho(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t))dt = \sum_{k=1}^{K} T(l_k)e^{q_k}\mathbf{c}_k$$

where τ is a coefficient that determines the rate of absorption, t_n and t_f denotes the near and far bound alone the ray, $T(t)$ is the transmittance.

$$T(t) = \exp(-\tau \int_{-\infty}^{t} \rho(\mathbf{r}(s))ds) = \exp(-\tau \sum_{m=1}^{K} e^{q_m} \frac{\mathrm{erf}((t-l_m)/\sigma_m) + 1}{2})$$

and

$$\rho_m(\mathbf{r}(s)) = \exp(q_m - \frac{(s-l_m)^2}{2 \cdot \sigma_m^2})$$

Wang, A., Wang, P., Sun, J., Kortylewski, A., & Yuille, A. (2022, September). VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis. In *The Eleventh International Conference on Learning Representations*.

New Topic: Introduce NeRF.

(1 min video)

New Topic: Introduce NeRF.
Ray Tracing Volume Densities.

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

$T(t)$ transmistance on ray $\mathbf{r}(s)$

$C(\mathbf{r})$ Color output on pixel p (for ray $\mathbf{r}(s)$)

$\mathbf{d}$ Viewing direction

$\boldsymbol{\sigma}$ Volume density on a 3D location, represented by a network network learnt from images

$\mathbf{c}$ color on a 3D location, depends on the viewing directions, represented by a network network learnt from images

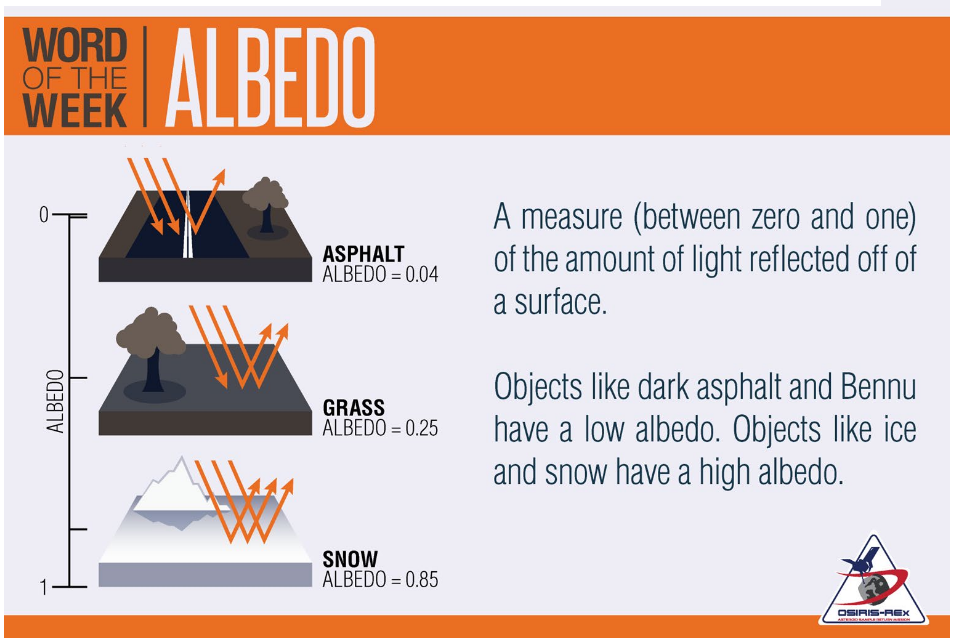
Kajiya J T, Von Herzen B P. Ray tracing volume densities[J]. ACM SIGGRAPH computer graphics, 1984

New Topic: Introduce NeRF.
Ray Tracing Volume Densities.

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

This formulation is obtained via simplify the scatter equation with Blinn's **Low Albedo** approximation.

- There is also a High Albedo solution for the scatter equation.



Kajiya J T, Von Herzen B P. Ray tracing volume densities[J]. ACM SIGGRAPH computer graphics, 1984

NeRF

NeRF represent objects as an implicit function, which compute a color C and volume density σ at each location X in the 3D space. What are the variables? The c and sigma are MLP functions of weights w which are learnt from a set of images of the scene from different (known) viewpoints.

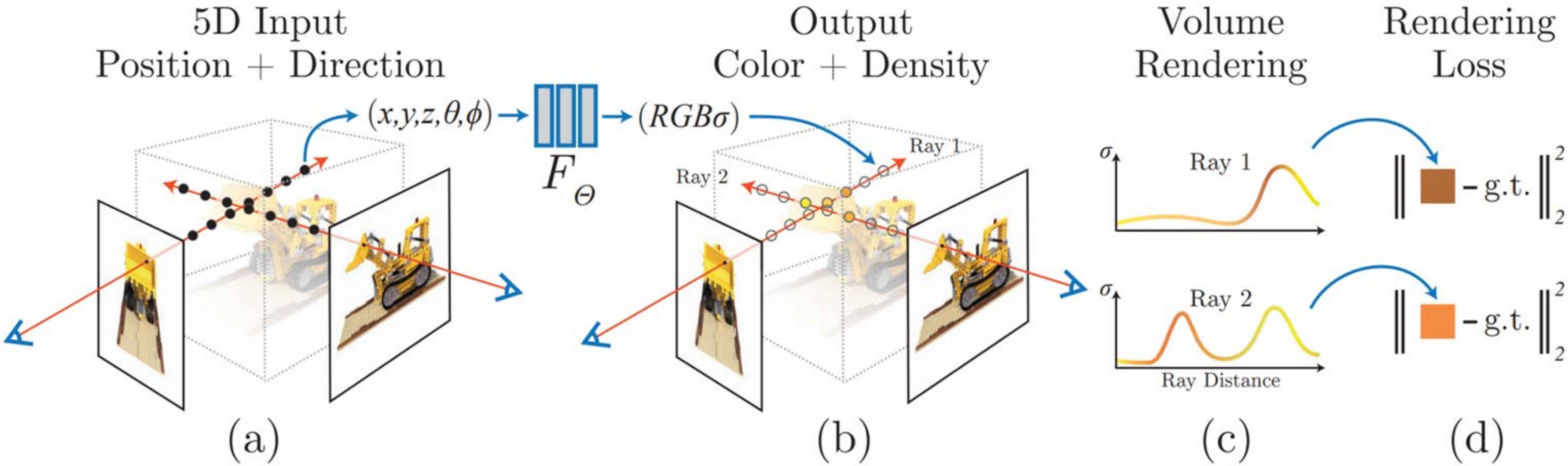
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

$$\Downarrow$$

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

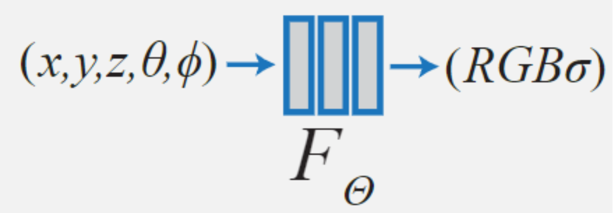
NeRF

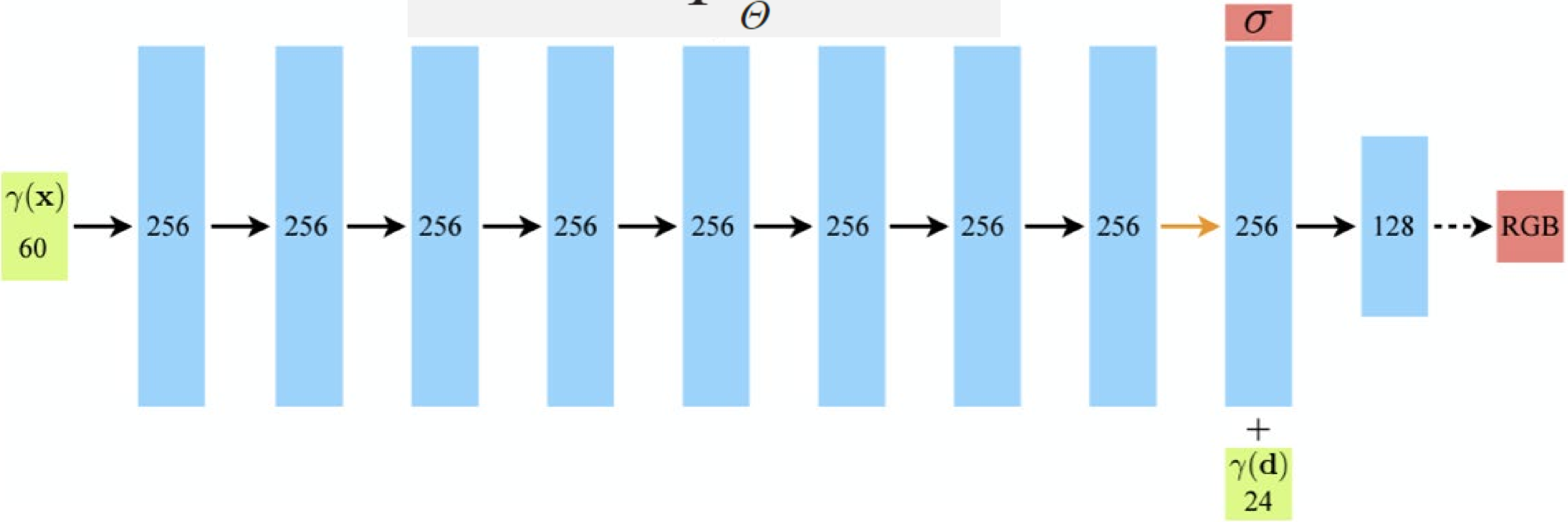
Views of scene from different angles are specified by different rays/cameras.



NeRF

The implicit function is presented using a MLP. The training data is a set of images. The implicit functions (their weights) are learnt by optimizing a function.





NeRF

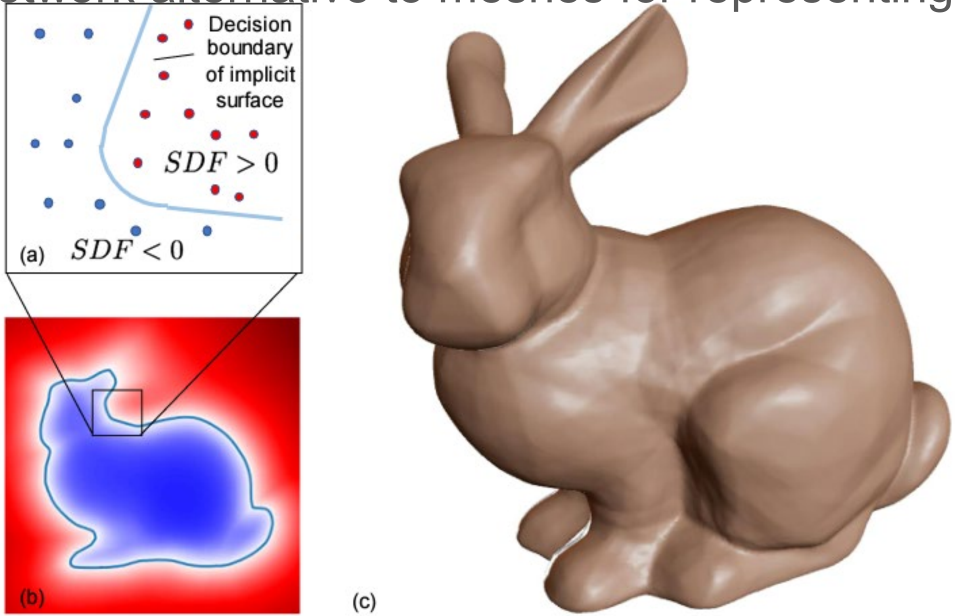
High quality on detailed geometry. (train on ~160 images)

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., & Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5470-5479).

CityNeRF (an application)

SDF: Neural Network representation of a Surface

SDF also represent objects as an implicit function, on each location X, the implicit function indicates the signed distance toward the object surface. This is a neural network alternative to meshes for representing surfaces.

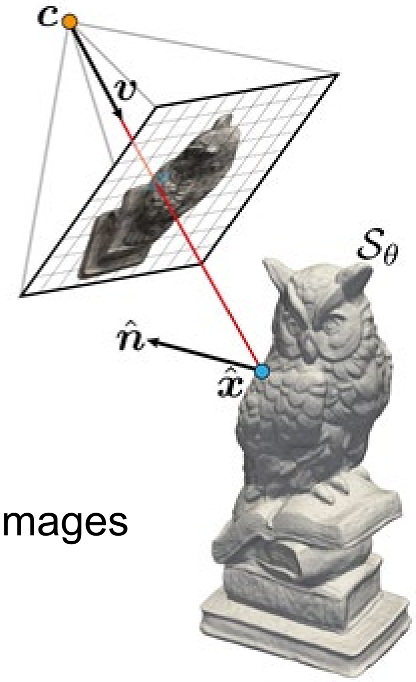


Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Render a SDF

IDR (Implicit Differentiable Renderer) is a surface based rendering technique for SDF.

- Object is represented as SDF.
- Each ray intersect with the hard surface at most once.
- The intersect point that SDF(x) = 0.



$$L_p(\theta, \gamma, \tau) = M(\hat{\boldsymbol{x}}_p, \hat{\boldsymbol{n}}_p, \hat{\boldsymbol{z}}_p, \boldsymbol{v}_p; \gamma),$$

M is an implicit renderer (neural network trained multiview images

$\hat{\boldsymbol{x}}_p$ is the surface points

$\hat{\boldsymbol{n}}_p$ is the surface normal

$\hat{\boldsymbol{z}}_p$ is a geometry latent

$\boldsymbol{v}_p$ is the viewing direction

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., & Lipman, Y. (2020). Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, *33*, 2492-2502.

IDR: Ray March (technical)

To find the surface intersection with a viewing ray, we find the zero level of the SDF along the viewing ray. (i.e. the boundary). We compute gradient to optimize network parameters θ.

Differentiable intersection of viewing direction and geometry

$$\hat{\boldsymbol{x}}(\theta, \tau) = \boldsymbol{c} + t_0 \boldsymbol{v} - \frac{\boldsymbol{v}}{\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0; \theta_0) \cdot \boldsymbol{v}_0} f(\boldsymbol{c} + t_0 \boldsymbol{v}; \theta),$$

$\hat{\boldsymbol{x}}(\theta, \tau) = \boldsymbol{c} + t(\theta, \boldsymbol{c}, \boldsymbol{v})\boldsymbol{v}$ denote the intersection point.

IDR: Loss for optimizing

Mask loss:

$$S(\theta,\tau) = \begin{cases} 1 & R(\tau) \cap \mathcal{S}_\theta \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad \Rightarrow \quad S_\alpha(\theta,\tau) = \text{sigmoid}\left(-\alpha \min_{t \geq 0} f(\boldsymbol{c} + t\boldsymbol{v}; \theta)\right)$$

RGB loss:

$$\text{loss}_{\text{RGB}}(\theta,\gamma,\tau) = \frac{1}{|P|} \sum_{p \in P^{\text{in}}} |I_p - L_p(\theta,\gamma,\tau)|$$

Eikonal loss:

$$\text{loss}_{\text{E}}(\theta) = \mathbb{E}_{\boldsymbol{x}}\big(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x};\theta)\| - 1\big)^2$$

Render a SDF

Vol-SDF is shown SDF functions can also be rendered via volume rendering.

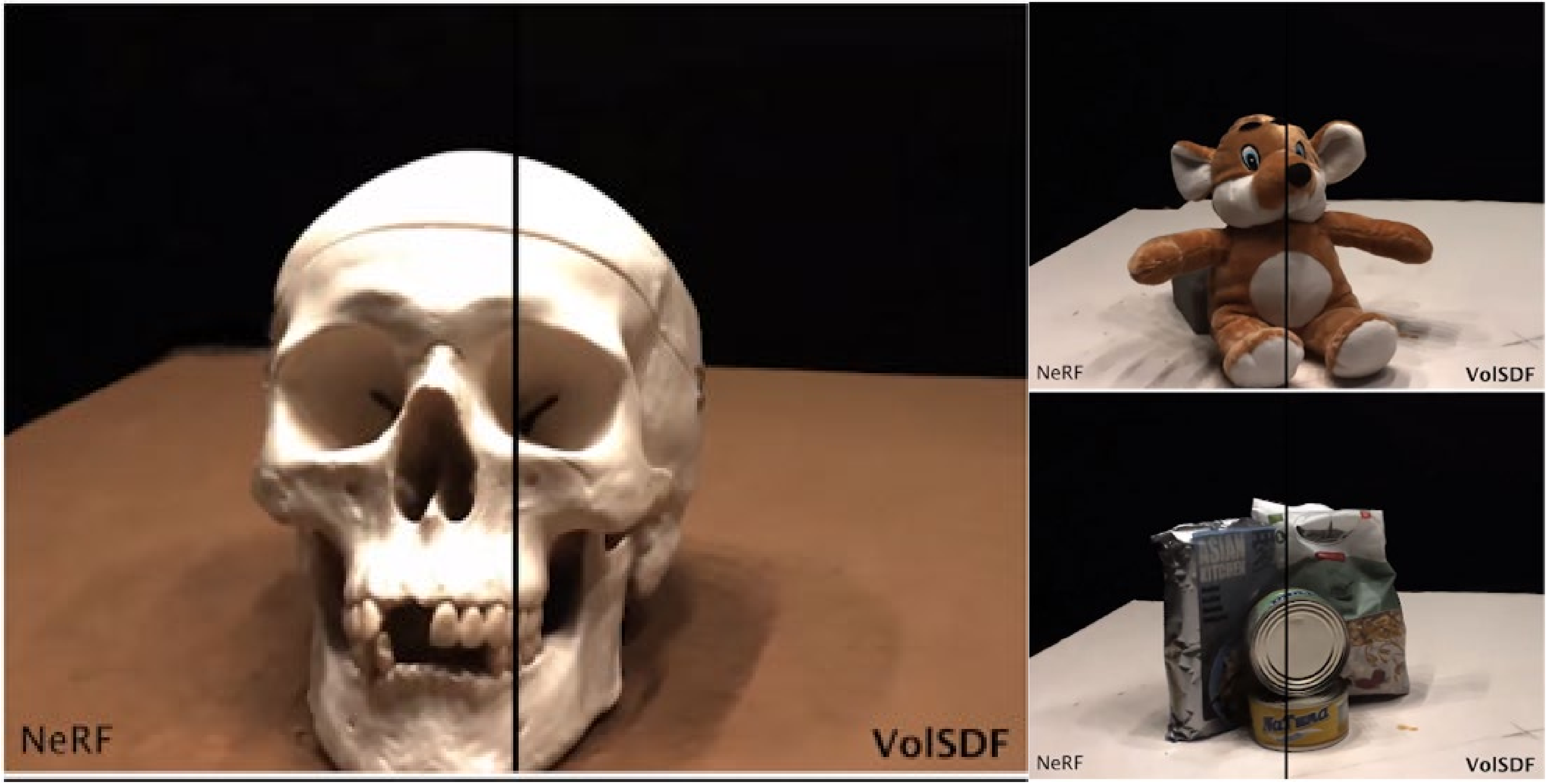
$$\sigma(\boldsymbol{x}) = \alpha \Psi_\beta\left(-d_\Omega(\boldsymbol{x})\right),$$

$$\Psi_\beta(s) = \begin{cases} \frac{1}{2}\exp\left(\frac{s}{\beta}\right) & \text{if } s \leq 0 \\ 1 - \frac{1}{2}\exp\left(-\frac{s}{\beta}\right) & \text{if } s > 0 \end{cases}$$

Yariv, Lior, et al. "Volume rendering of neural implicit surfaces." *Advances in Neural Information Processing Systems* 34 (2021): 4805-4815.

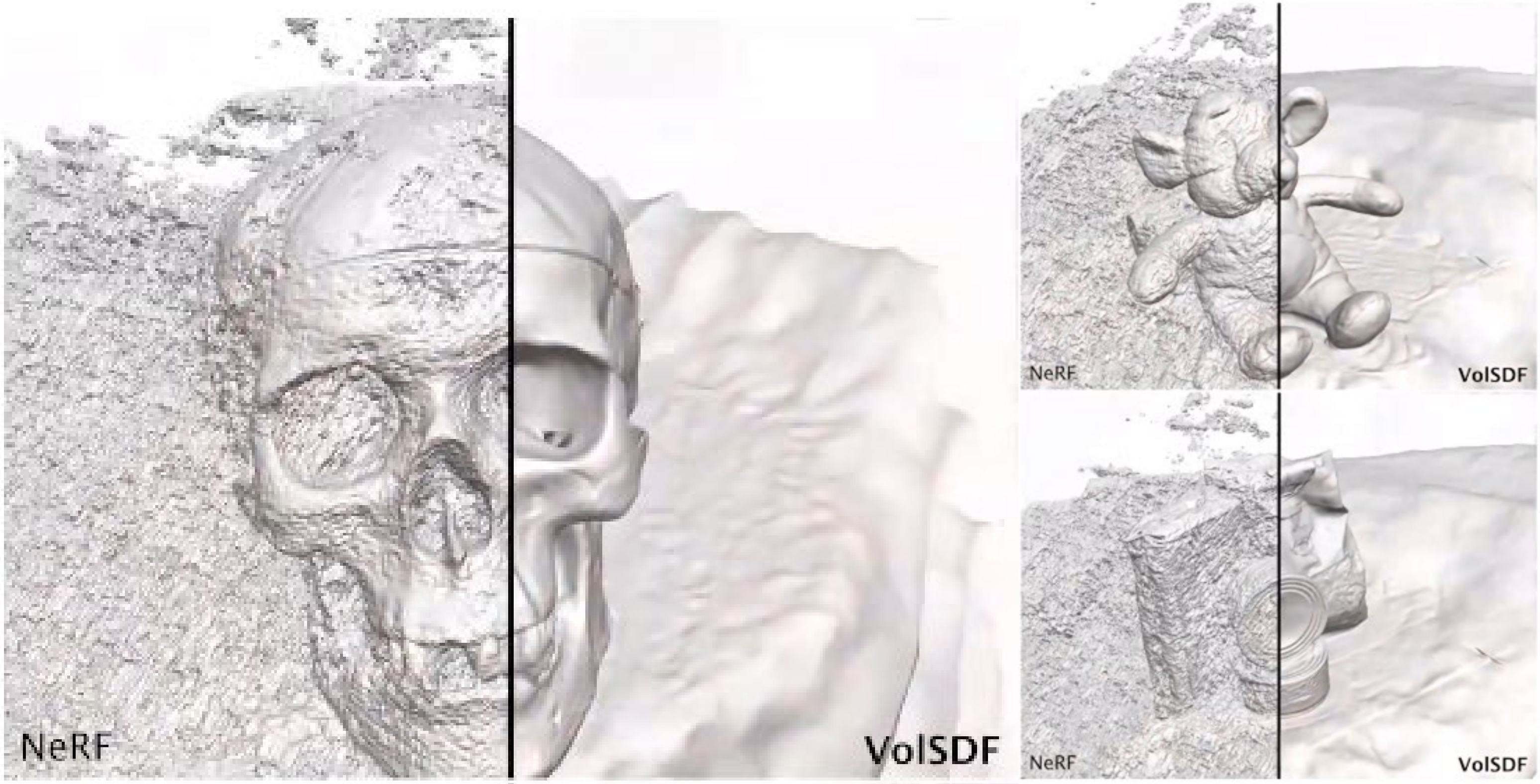
Vol-SDF

Yariv, Lior, et al. "Volume rendering of neural implicit surfaces." *Advances in Neural Information Processing Systems* 34 (2021): 4805-4815.

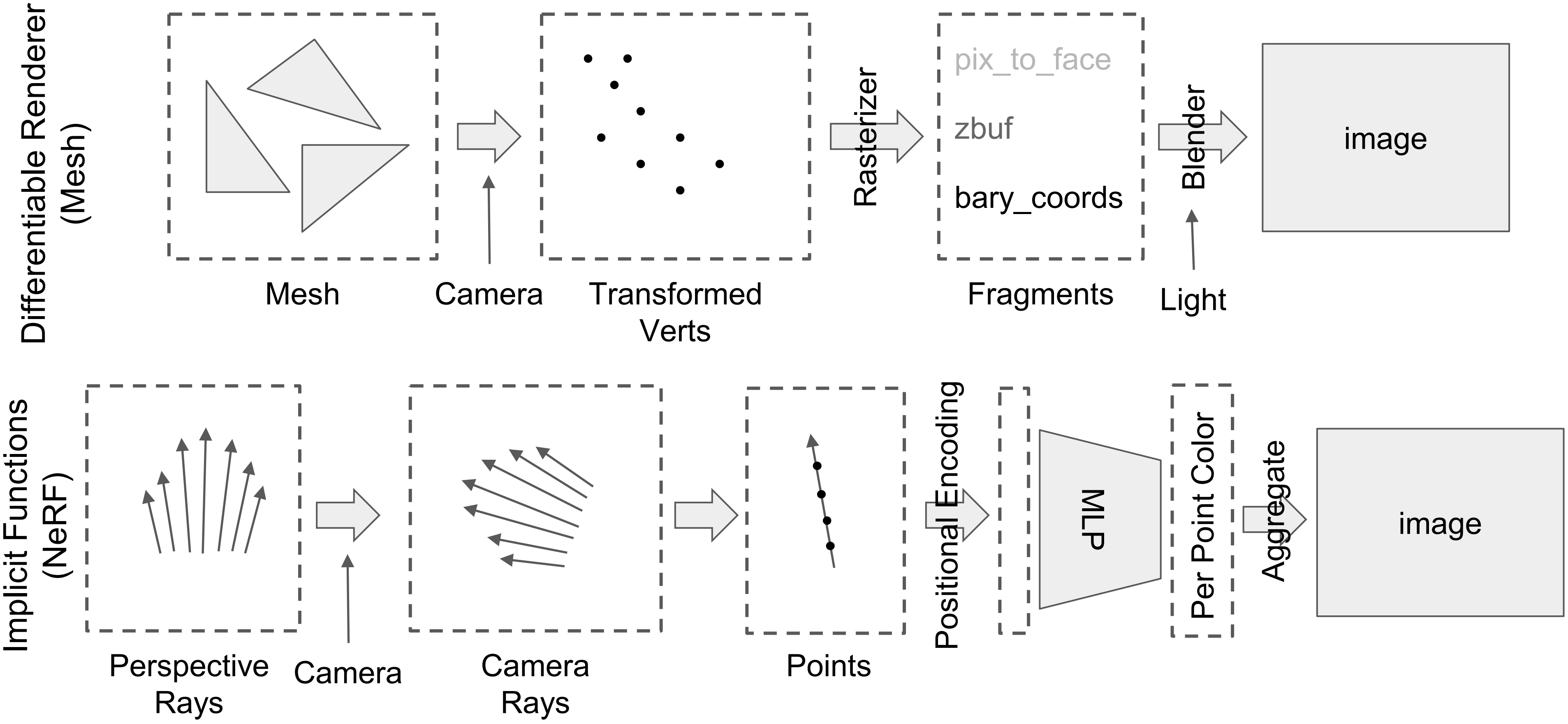


Vol-SDF

Yariv, Lior, et al. "Volume rendering of neural implicit surfaces." *Advances in Neural Information Processing Systems* 34 (2021): 4805-4815.



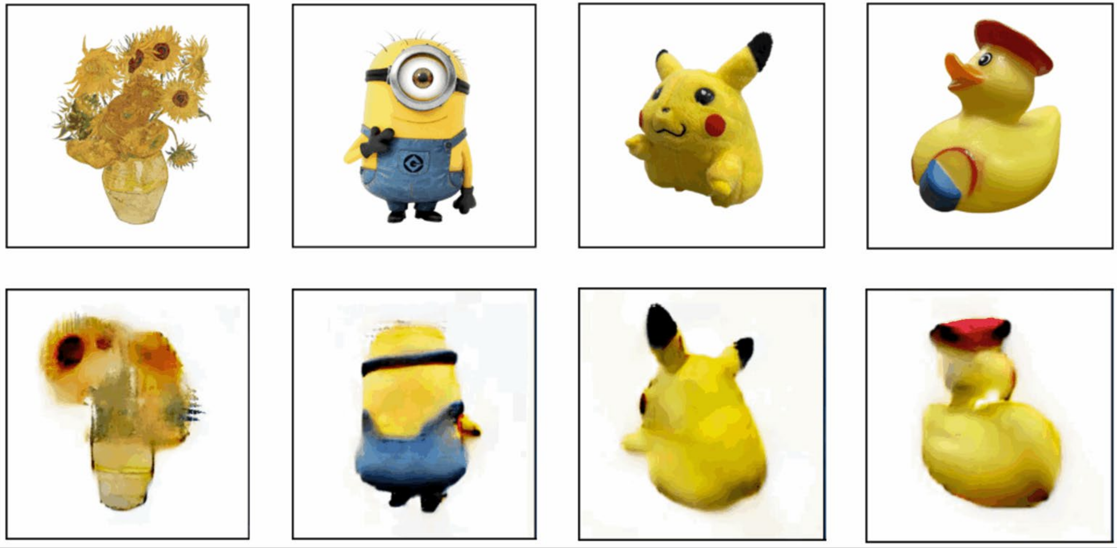
Traditional Differentiable Renderer vs. Implicit Functions



3D from A Single Image

How to reconstruct 3D from just a single image?

One solution is to combine NeRF with Diffusion Models.



Liu, Ruoshi, et al. "Zero-1-to-3: Zero-shot one image to 3d object." *ICCV* 2023.

3D from A Single Image

Zero-1-to-3: a viewpoint-conditioned image diffusion model

Input image

x

f

Diffusion Model
(Zero-1-to-3)

Text prompt (R, T)
Down 30° Left: 90°

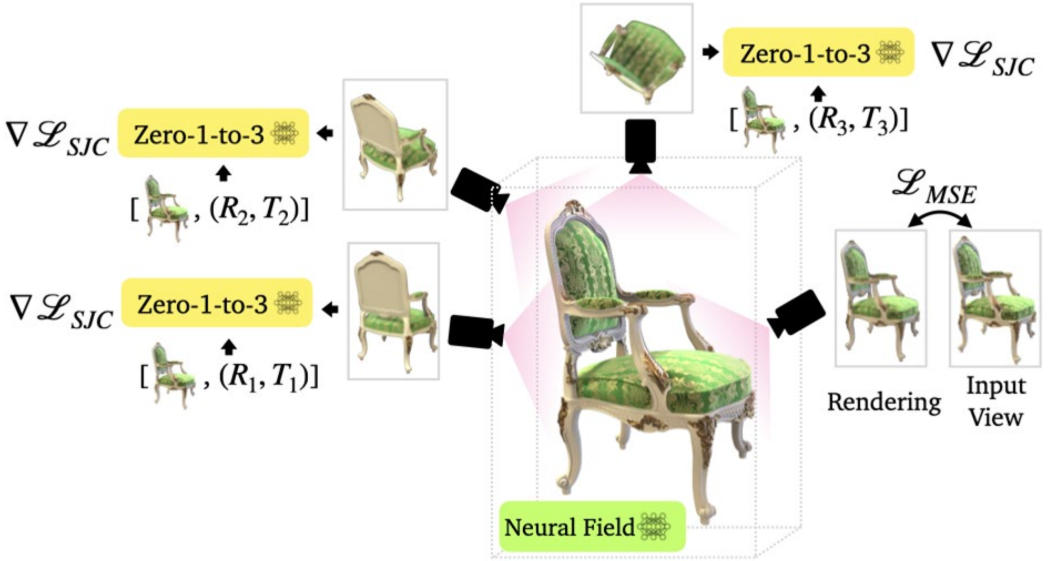
output image

$\hat{x}_{R,T} = f(x, R, T)$

Liu, Ruoshi, et al. "Zero-1-to-3: Zero-shot one image to 3d object." *ICCV* 2023.

3D from A Single Image

During training, we randomly sample viewpoints and use Zero-1-to-3 to supervise the 3D reconstruction.



Liu, Ruoshi, et al. "Zero-1-to-3: Zero-shot one image to 3d object." *ICCV* 2023.

Summary of Talk

Classic Models – Lambertian, Phong, BRDF. Okay for single objects but less good for scenes. Parameters of the models are hand-specified – hard to find albedoes.

Learning based methods – Gaussian splatting, VoGE, NeRF, SDF – require training data often from many known viewpoints. (Do not change lighting).

Learning based methods apply to more complex scenes and give much higher quality images.

Editing Diffusion models by prompts means we can use NeRF or Gaussian splatting to estimate the 3D structure of an object.