# RENDERING AND GEOMETRY FROM 3D GAUSSIAN SPLATTING

SYMPOSIUM ON GEOMETRY PROCESSING

# COURSE OVERVIEW

- Duration: 90 minutes
- Introduction to 3DGS (Bernhard Kerbl)
- Artifacts (Thomas Köhler)
- Geometry Extraction Overview (Felix Windisch)

- *3D Gaussian Splatting* is **novel-view synthesis**

- Predict custom views from images and poses

- Our Goals
  - **Real-time rendering**
  - High quality
  - Fast and easy to obtain
  - Reasonably compact

# SfM Points

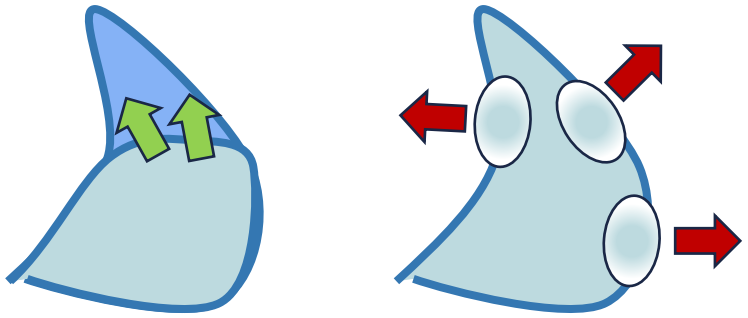
# Splatting

# THE PART WE DIDN‘T MENTION YET: DENSIFICATION

- Increase Gaussians if necessary
- **High positional gradients** mark demanding regions
- Intuition: XYZ gradient acts as a 3D discontinuity detector
- **Densify** regions incrementally

# POSITIONAL GRADIENT

- Caused by Discontinuities
  - Between current rendering and target image
  - Within the actual target image



- Weaknesses
  - Densification occurs primarily in areas that are **already dense**
  - Final model size effectively uncontrollable!
  - No method for introducing user priors

# 3D GAUSSIAN SPLATTING AS MARKOV CHAIN MONTE CARLO (NEURIPS 2024)

- 3DGS starts from SfM points for initialization
- MCMC can use **random** initialization
- **Highest** 3DGS quality, on par with best NeRF-based methods

Table 1: **Quantitative results with same number of Gaussians –** Our method outperforms all baselines even when starting from random initialization, with a large gap in performance when compared with 3DGS [14] – Random. We highlight the **best** and *second-best* for each column.

| | NeRF Synthetic [22] | MipNeRF 360 [2] | Tank & Temples [17] | Deep Blending [13] | OMMO [21] |
|---|---|---|---|---|---|
| | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ |
| NeRF [23] | 31.01 / - / - | 24.85 / 0.66 / 0.43 | - | 21.18 / 0.78 / 0.34 | - |
| Plenoxels [37] | 31.76 / - / - | 23.63 / 0.67 / 0.44 | 21.08 / 0.72 / 0.38 | - | - |
| INGP-Big [23] | 33.18 / - / - | 26.75 / 0.75 / 0.30 | 21.92 / 0.75 / 0.31 | - | - |
| MipNeRF [1] | 33.09 / - / - | 27.60 / 0.81 / 0.25 | - | 21.54 / 0.78 / 0.37 | - |
| MipNeRF360 [2] | - | 29.23 / 0.84 / 0.21 | 22.22 / 0.76 / 0.26 | - | - |
| 3DGS [14]→ [4] | 33.32 / - / - | 28.69 / 0.87 / 0.22 | 23.14 / 0.84 / 0.21 | - | - |
| 3DGS [14] (Random) | *33.42* / **0.97** / **0.04** | 27.89 / 0.84 / 0.26 | 21.93 / 0.79 / 0.27 | 29.55 / **0.90** / 0.33 | 28.24 / 0.88 / 0.24 |
| Ours (Random) | **33.80** / **0.97** / **0.04** | *29.72* / *0.89* / **0.19** | *24.21* / **0.86** / **0.19** | **29.71** / **0.90** / **0.32** | *29.31* / *0.90* / **0.20** |
| 3DGS [14] (SfM) | - | 29.30 / 0.88 / 0.21 | 23.67 / 0.84 / 0.22 | 29.64 / **0.90** / **0.32** | 28.83 / 0.89 / 0.22 |
| Ours (SfM) | - | **29.89** / **0.90** / **0.19** | **24.29** / **0.86** / **0.19** | *29.67* / 0.89 / **0.32** | **29.52** / **0.91** / **0.20** |

**Shakiba Kheradmand[1], Daniel Rebain[1], Gopal Sharma[1], Weiwei Sun[1], Yang-Che Tseng[1], Hossam Isack[2], Abhishek Kar[2], Andrea Tagliasacchi[3, 4, 5], Kwang Moo Yi[1]**

[1]University of British Columbia, [2]Google Research, [3]Google DeepMind, [4]Simon Fraser University, [5]University of Toronto

# USEFUL GAUSSIAN SPLATS

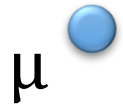
- Dynamic Scenes: 4D Gaussian Splatting
- Inverse Rendering and PBR Gaussians (Relighting)
- Generative AI for 3D Content
- Applications! 3DGS is used for
  - Surface Reconstruction
  - Fast Structure-from-Motion and Multi-View Stereo
  - Collision and Physics Simulation (Robotics)
  - **Human Avatars** (Full session at SIGGRAPH Asia ’24)

# 3D GAUSSIAN SPLATTING EVERYWHERE

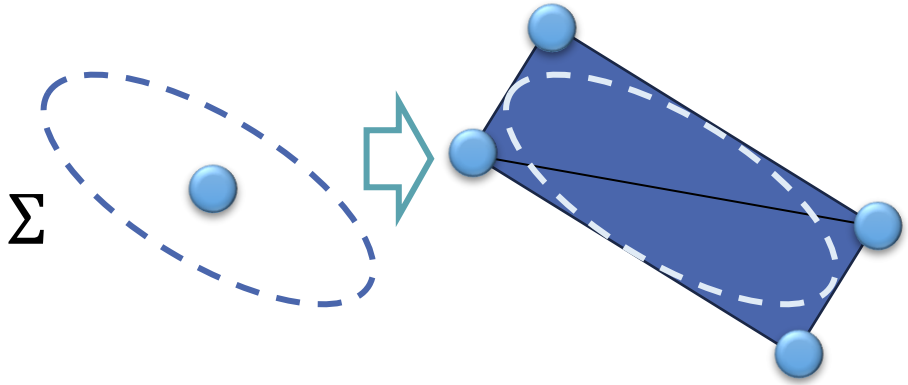
# 3D GAUSSIAN SPLATTING & ALPHA BLENDING

$$I(x) = \sum_i \alpha_i(x) c_i \prod_j^i 1 - a_j(x), \qquad \alpha = oG(x), \qquad G(x) = e^{-0.5(x-\mu)^T \Sigma'^{-1}(x-\mu)}$$

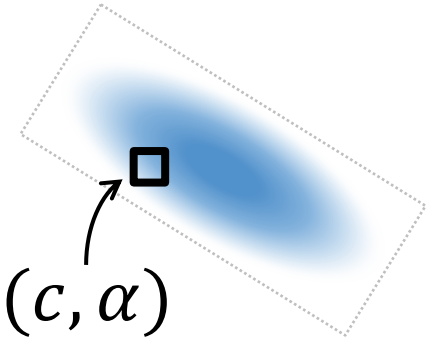
1. Vertex Shader




2. Geometry Shader




3. Fragment Shader

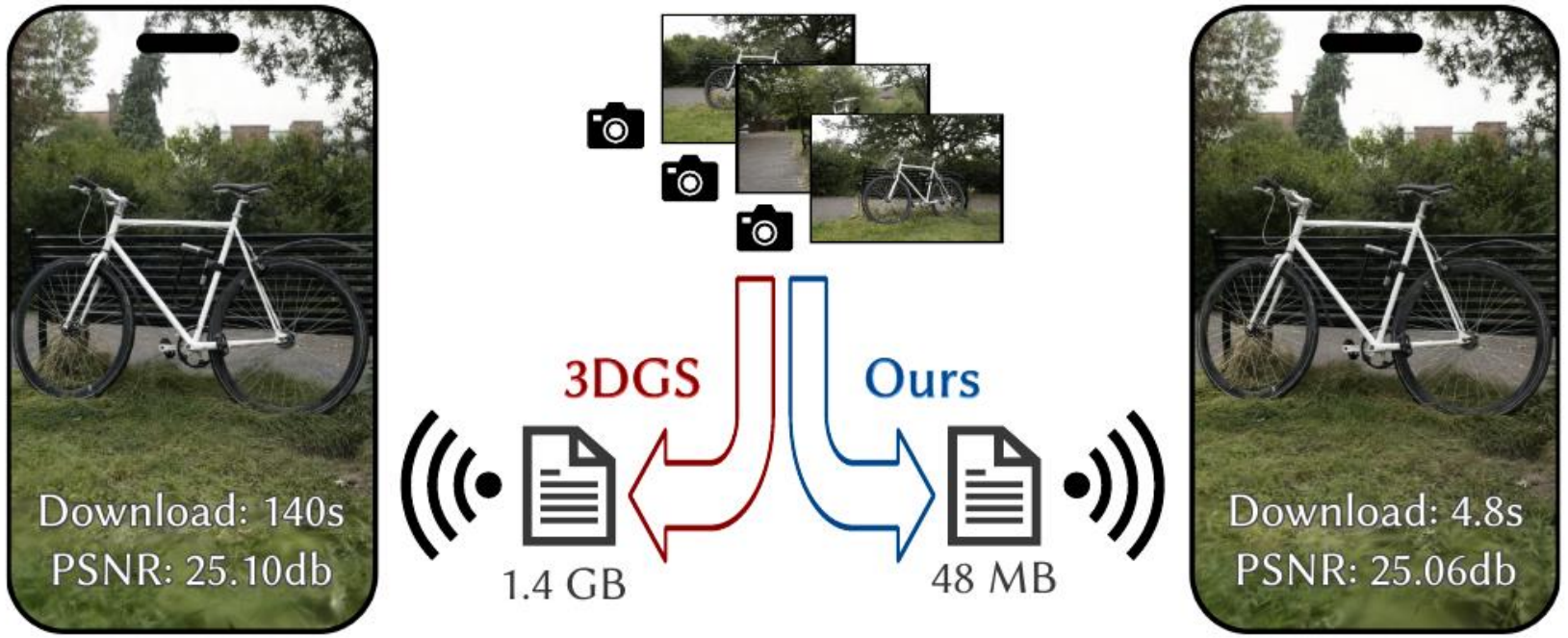



4. Blending

$$\alpha_1 \square + (1 - \alpha_1)\alpha_2 \square$$

# REDUCING THE SIZE OF 3DGS SCENES

- Is 3DGS a solution for fast, portable 3D viewing?
- Training speed ✅ Rendering speed ✅ Download speed ❌
- Generated `.ply` range from a few dozen MiB to more than one GiB
- Smaller than Plenoxels volumes, but much bigger than NeRF scenes

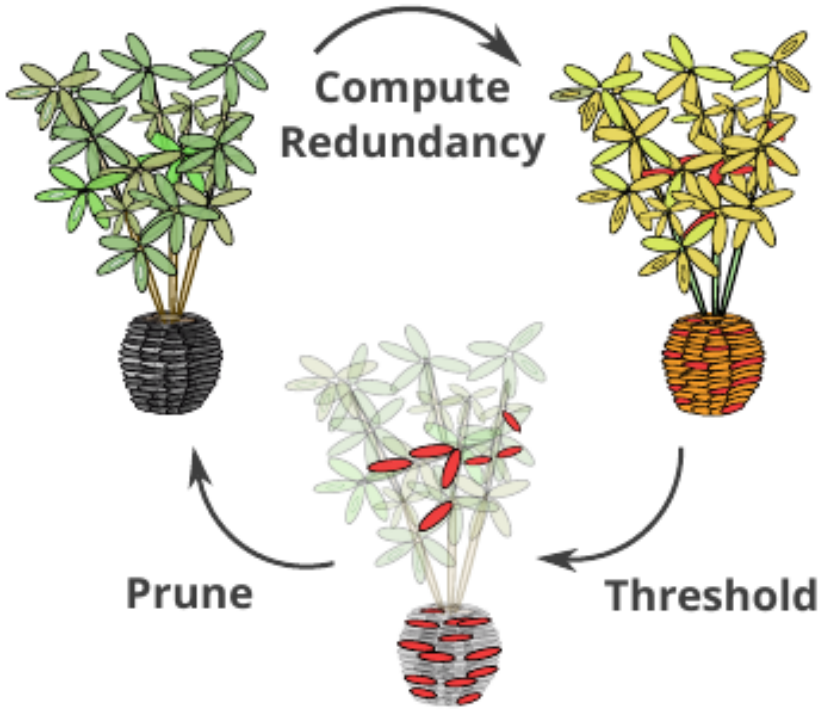



# SHRINK IT DOWN! COMPRESSED 3DGS

# THE MEMORY COST OF 3DGS MODELS

- 59 x 4 bytes to represent a single Gaussian
- Millions of them!

# 3-STEP GAUSSIAN COMPRESSION




**1. Pruning (start-to-end)**

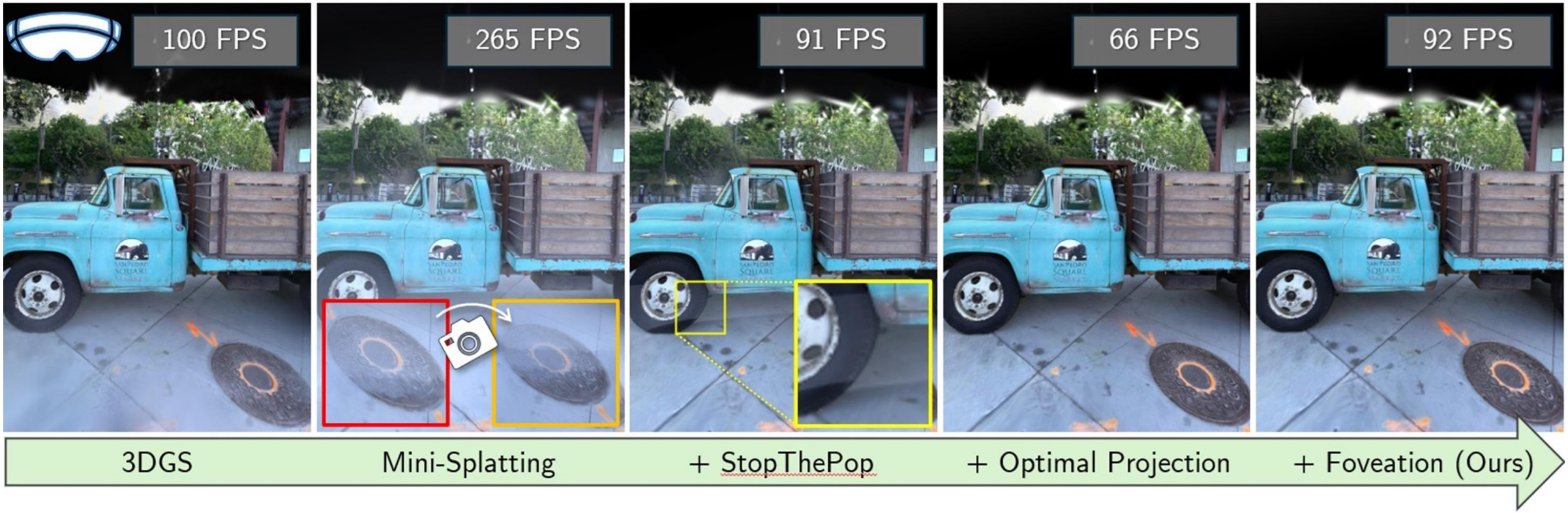
+0.03db / 2.37×

# Reducing the Memory Footprint of 3D Gaussian Splatting

submission n°1

# BRINGING IT ALL TOGETHER: 3DGS FOR VIRTUAL REALITY

- Small models + artifact-free rendering are vital for a good user experience

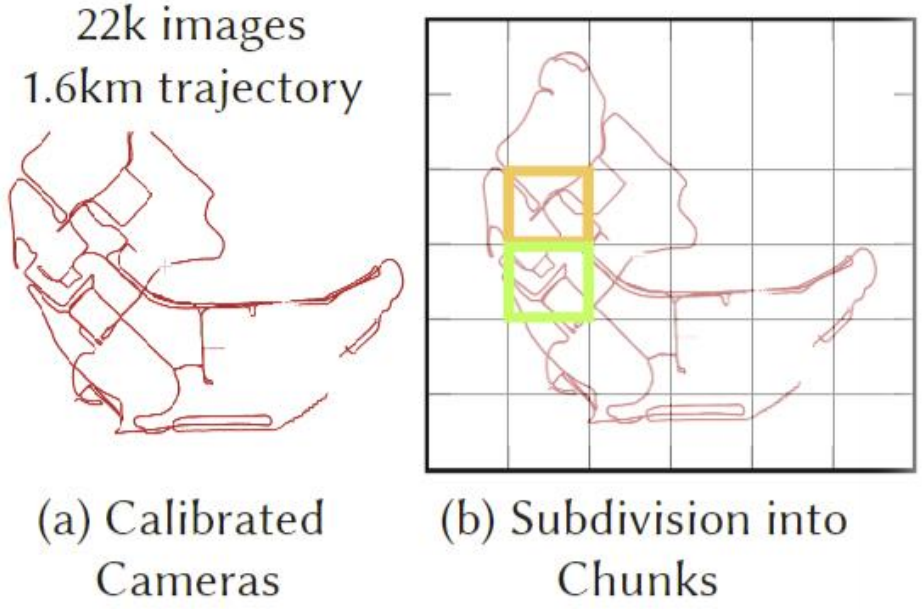


# Truck


# SCALING IT UP: LEVEL-OF-DETAIL 3DGS

# A HIERARCHICAL 3D GAUSSIAN REPRESENTATION (SIGGRAPH ‘24)

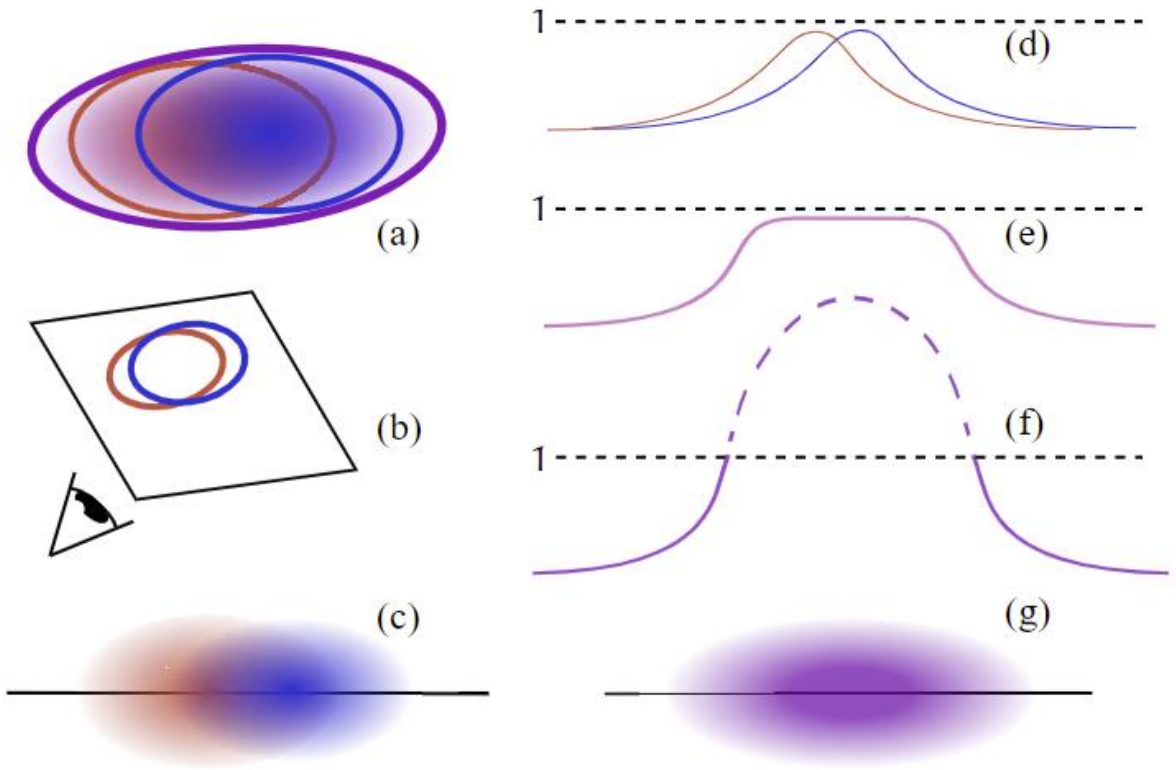
Bernhard Kerbl*, Andreas Meuleman*, Georgios Kopanas, Alexandre Lanvin, Michael Wimmer, George Drettakis



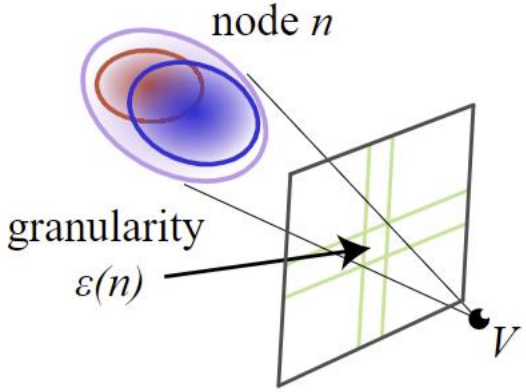

(a) Calibrated Cameras

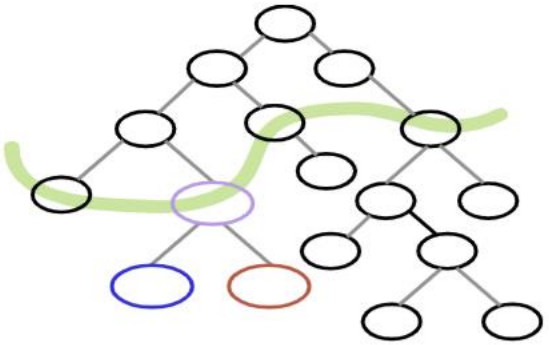
(b) Subdivision into Chunks

# LOD GENERATION, SELECTION & SWITCHING

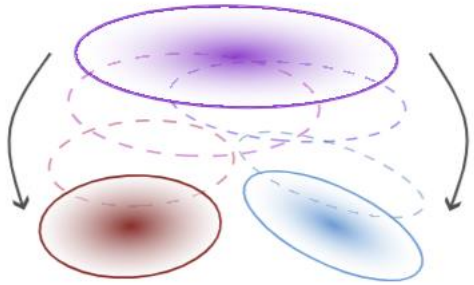



**Generate**



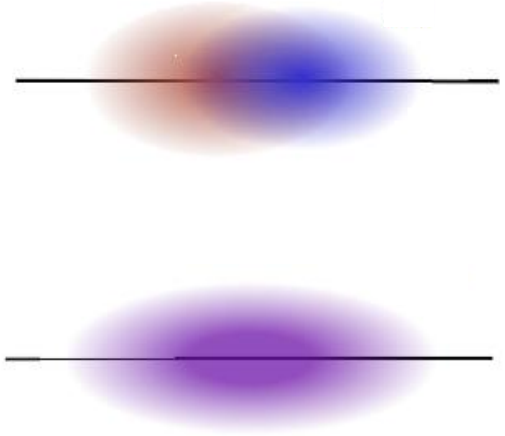
**Select**



**Switch**

# GENERATING HIERARCHICAL LEVELS

- How do cluster multiple Gaussians into one?
- Mean and covariance are well-covered in literature, method to minimize KL divergence:



$$\mu^{(l+1)} = \sum_{i}^{N} w_i \mu_i^{(l)},$$

$$\Sigma^{(l+1)} = \sum_{i}^{N} w_i (\Sigma_i^{(l)} + (\mu_i^{(l)} - \mu^{(l+1)})(\mu_i^{(l)} - \mu^{(l+1)})^T)$$

- What about *the weights?* We compute them using the product of opacity and ***surface area***

# LEVEL-OF-DETAIL SELECTION BASED ON DISTANCE AND QUALITY

Layer L

Layer L + 2

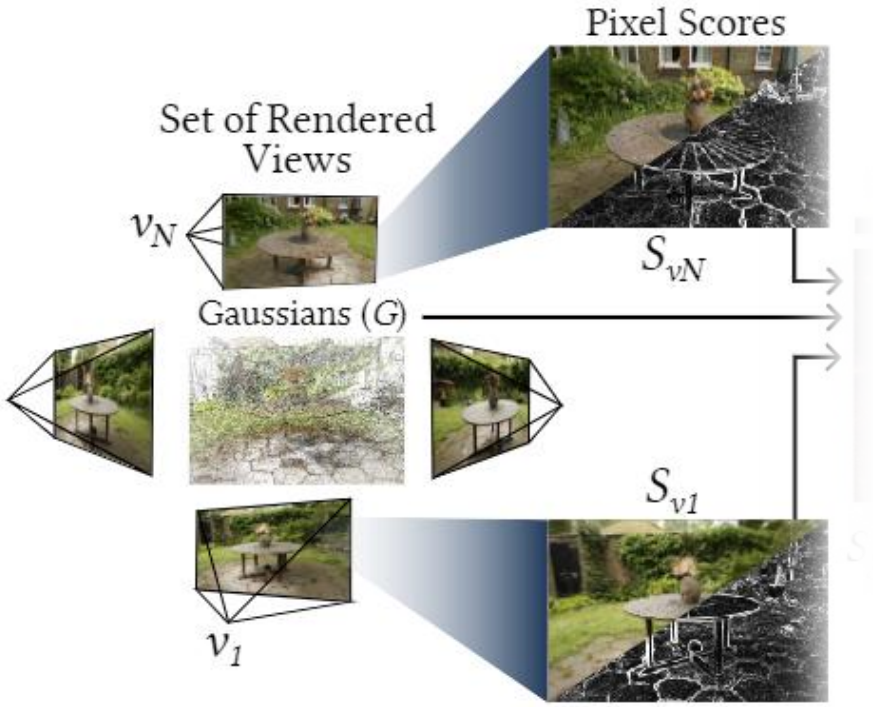
# STREAMING RENDERER PROTOTYPE

- At each point, the GPU only renders (and stores) the selected Gaussians
- As the camera moves, Gaussians are streamed to, and evicted from, GPU VRAM

# TAMING 3DGS: HIGH-FIDELITY MODELS IN 10 MINUTES

Saswat Mallick*, Rahul Goel*, Bernhard Kerbl, Francisco Carrasco, Markus Steinberger, Fernando de la Torre


# TAMING 3DGS



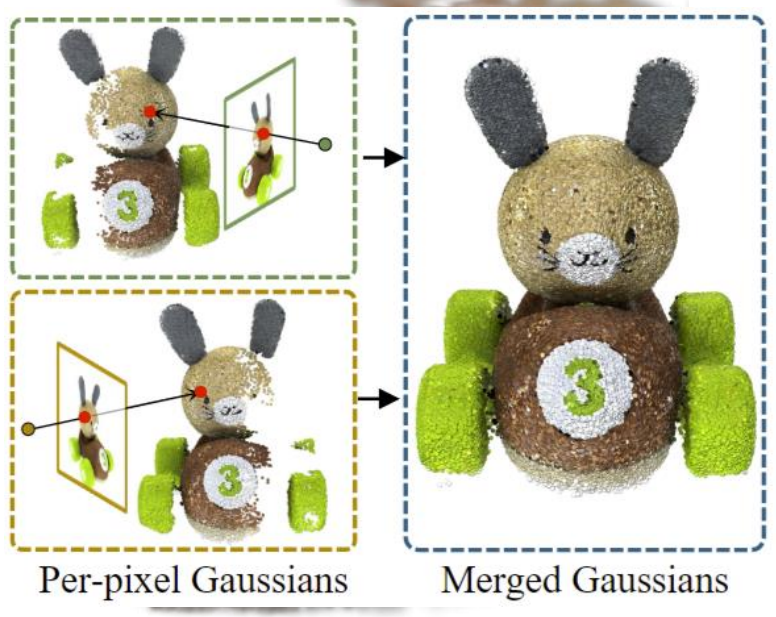
# Taming 3DGS
# High-Quality Radiance Fields with Limited Resources



TOWARDS
LIGHTNING-FAST
3D GAUSSIAN MODELS
Generating scene-scale reconstructions in seconds

# ATTENTION IS ALL YOU NEED FOR INSTANT 3D GAUSSIAN SPLATS

- Idea: use transformers (+ attention) to predict per-pixel Gaussians from sparse inputs
- Inductive 3D bias via epipolar attention: Pixelsplat (Charatan et al., 2024)
- Simplify, only exhaustive cross attention: GS-LRM (Zhang et al., 2024)
- Mix Transformers + Mamba: Long-LRM (Ziwen et al., 2024)
  - Quadratic complexity in number of tokens painfully slow
  - Go from sparse inputs to denser ones with Mamba



# FROM UP TO 32 IMAGES, IN 1.3 SECONDS

# THANK YOU!

https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

https://github.com/graphdeco-inria/gaussian-splatting

Funding: ERC Advanced Grant FUNGRAPH
http://fungraph.inria.fr

# THANK YOU

Bernhard Kerbl
kerbl@cg.tuwien.ac.at

Thomas Köhler
t.koehler@tugraz.at

Felix Windisch
felix.windisch@tugraz.at

# Artifacts in Gaussian Splatting

Thomas Köhler

Symposium on Geometry Processing - 2025


Source: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

# Artifacts?


Source: https://cekavis.site/VRSplat/

# Goals

1. Understand origin of artefacts
2. Explore different approaches
3. Overview of latest research

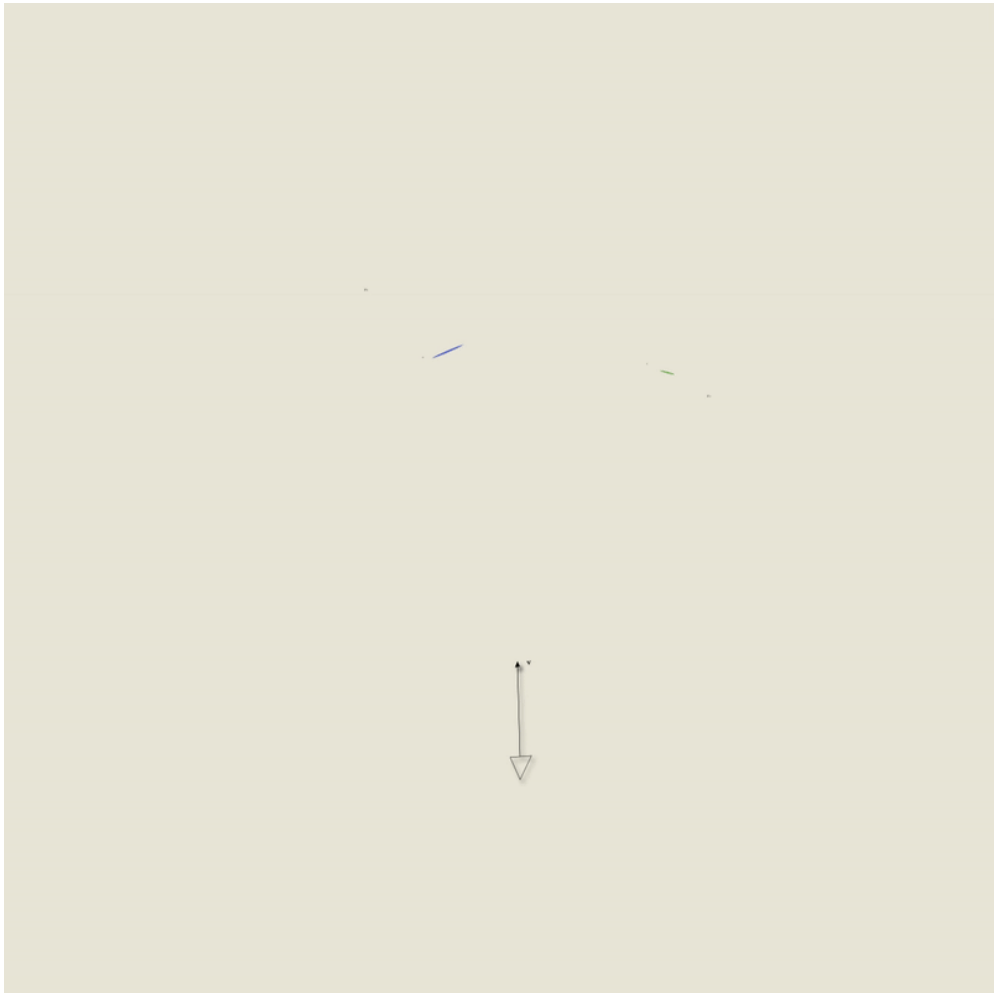
# Popping

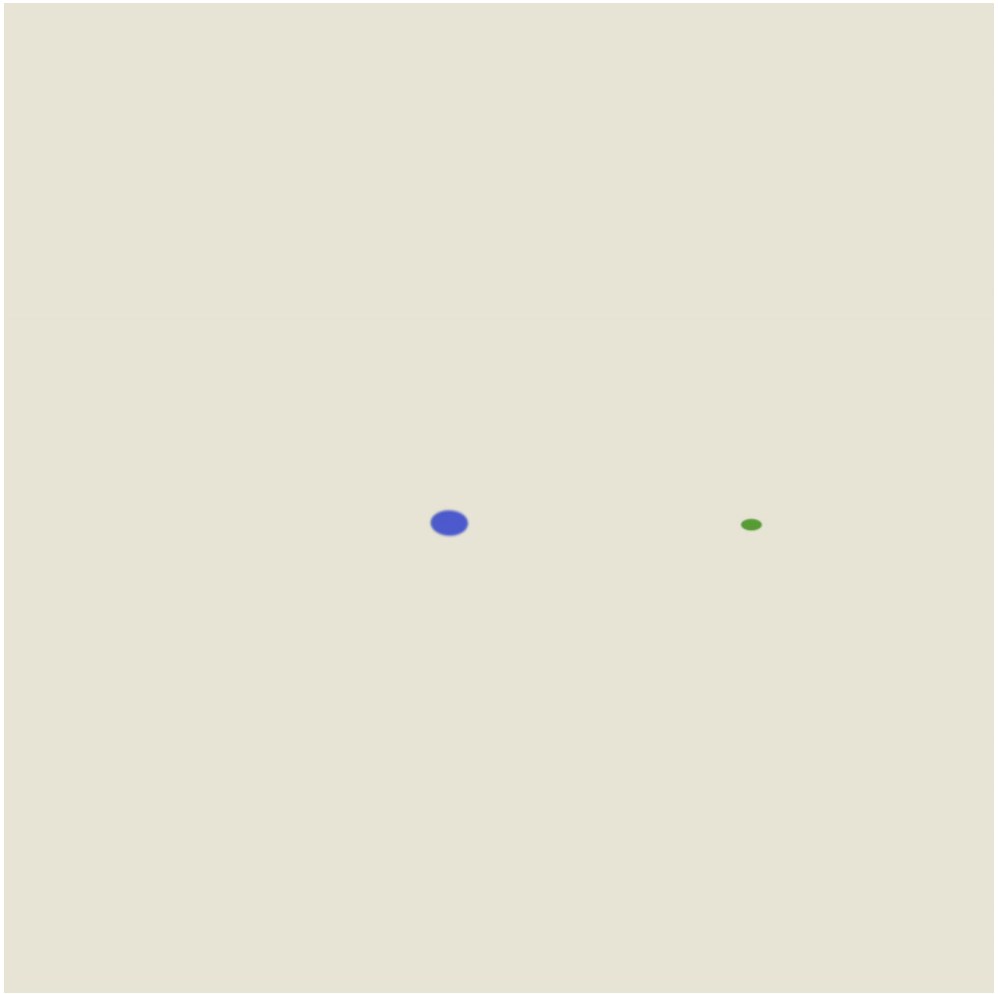
- Gaussians are transparent [Kerbl et al. 2023]
- Alpha blending:
  - Order dependent

$$C(\mathbf{r}) = \sum_{i=1}^{N_\mathbf{r}} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

- Easiest: Global sort view space mean

# Popping

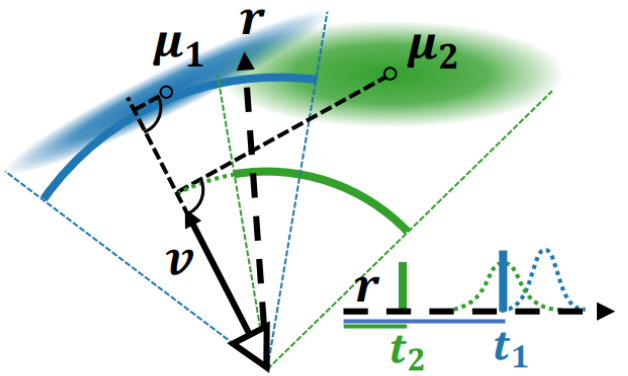




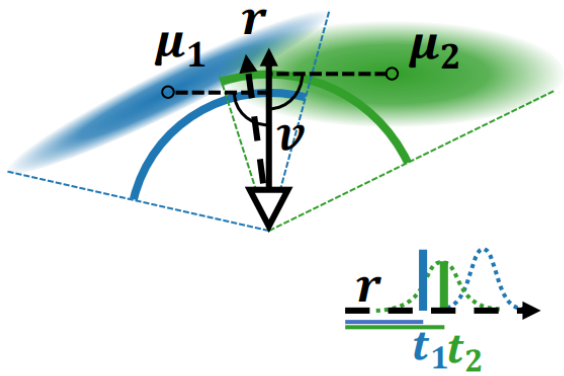
Source: https://r4dl.github.io/StopThePop/

# Depth

$$t_i = \mu_i^T \mathbf{v}$$

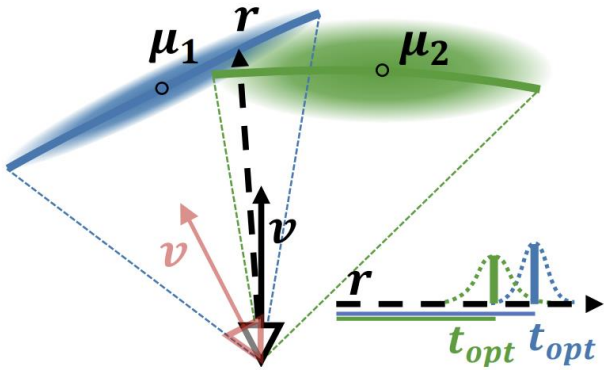


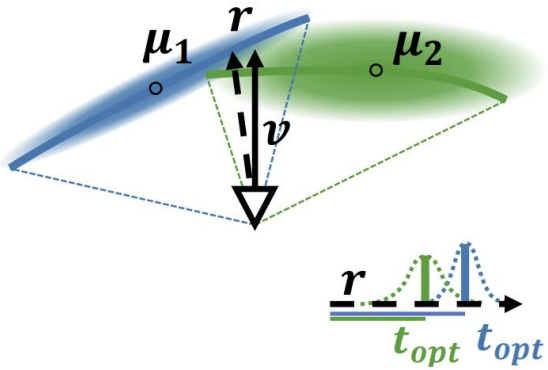

(c)



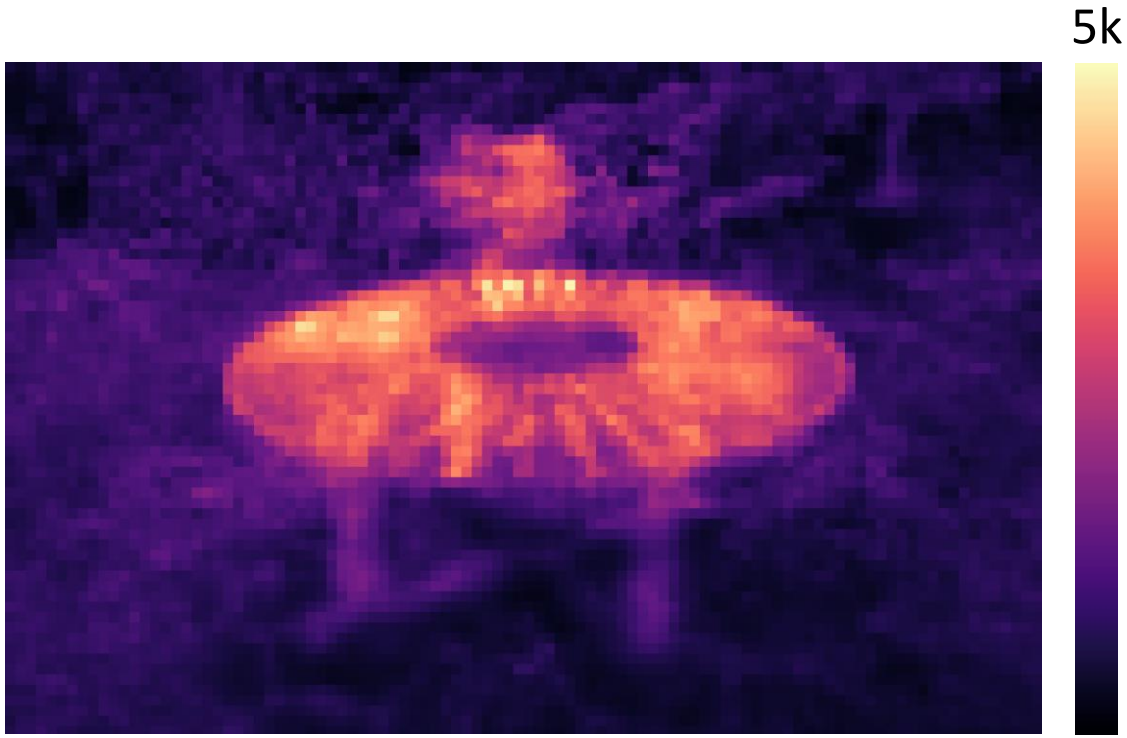

(d)

$$t_{opt} = \frac{\mathbf{d}^T \Sigma^{-1} (\mu - \mathbf{o})}{\mathbf{d}^T \Sigma^{-1} \mathbf{d}}$$



# Sorting




N Gaussians Per Tile

| | | 3DGS | Full Sort |
|---|---|---|---|
| Train | $\delta_{\max}$ | 28.445 | 0.000 |
| | $\delta_{\text{avg}}$ | 3.688 | 0.000 |
| | runtime | 1.00 | 142.03 |
| Bonsai | $\delta_{\max}$ | 33.543 | 0.000 |
| | $\delta_{\text{avg}}$ | 3.786 | 0.000 |
| | runtime | 1.00 | 179.70 |

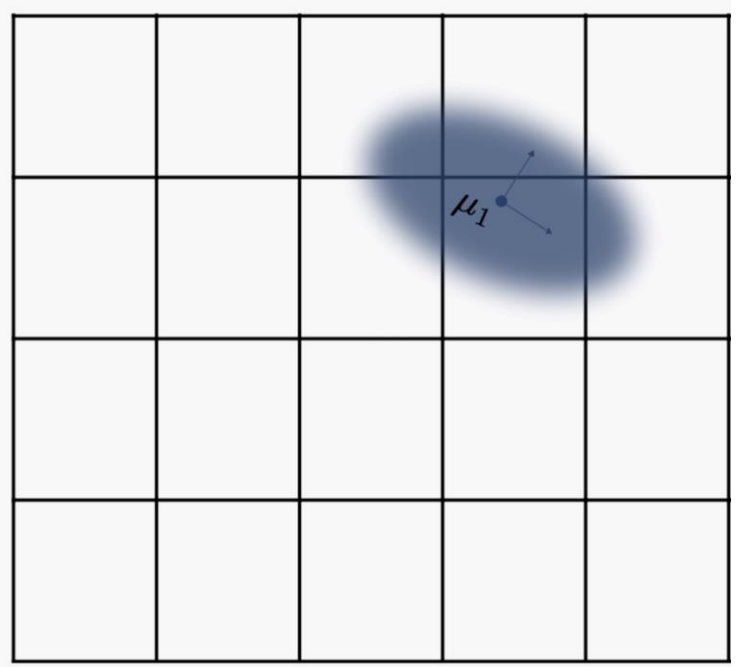
[Radl et al. 2024]

# Order Independent Transparency

- Approximate sort with k-Buffers [Callahan et al. 2005]
- Fixed-size buffers
  - Avoids memory overhead of A-buffers
  - Blend closest fragments on overflow
- Relies on nearly sorted incoming fragments

# Tile Based Depth

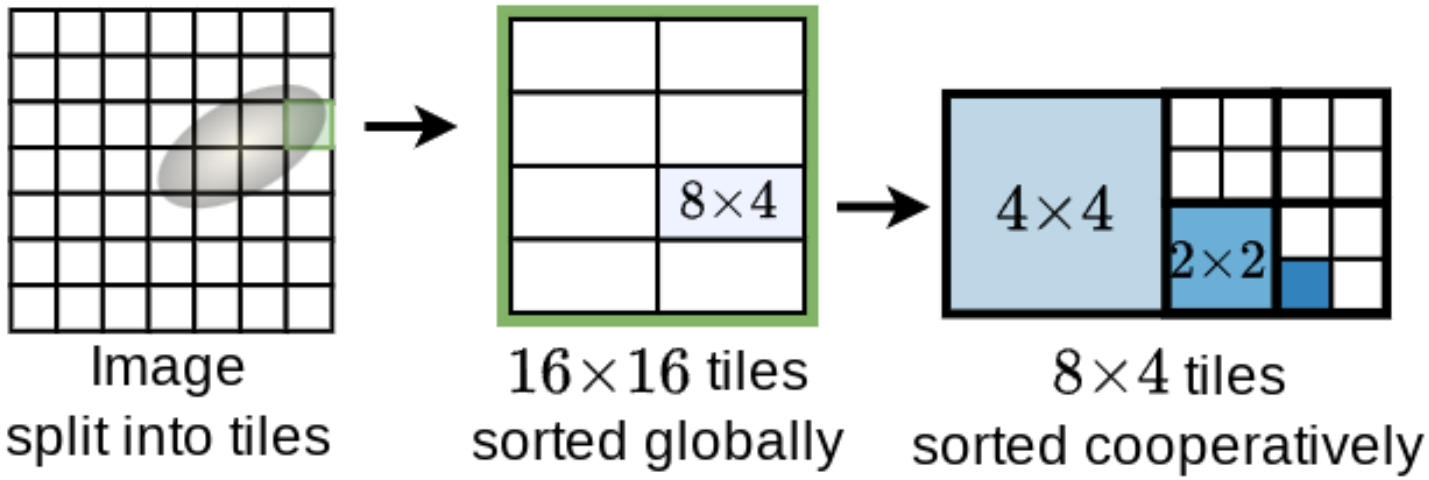
- Depth smooth across neighbouring rays
  - Similar sorting order
  - Sort Gaussians per tile
- Higher accuracy
- Avoids per-pixel sort



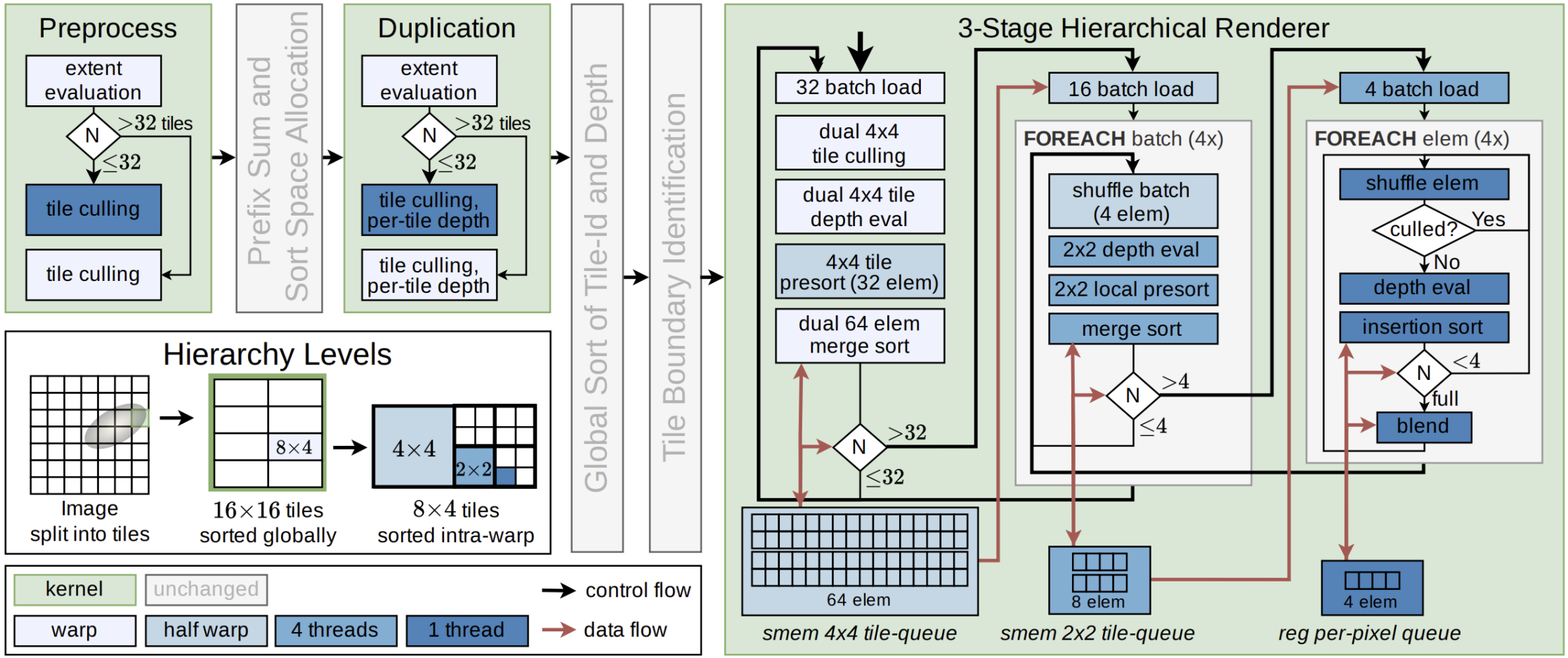
# Tile Based Depth

# Hierarchical Sorting

- Refine sort oder
  - Incrementally smaller tiles
  - Sort queues on every hierarchy



# Hierarchical Sorting



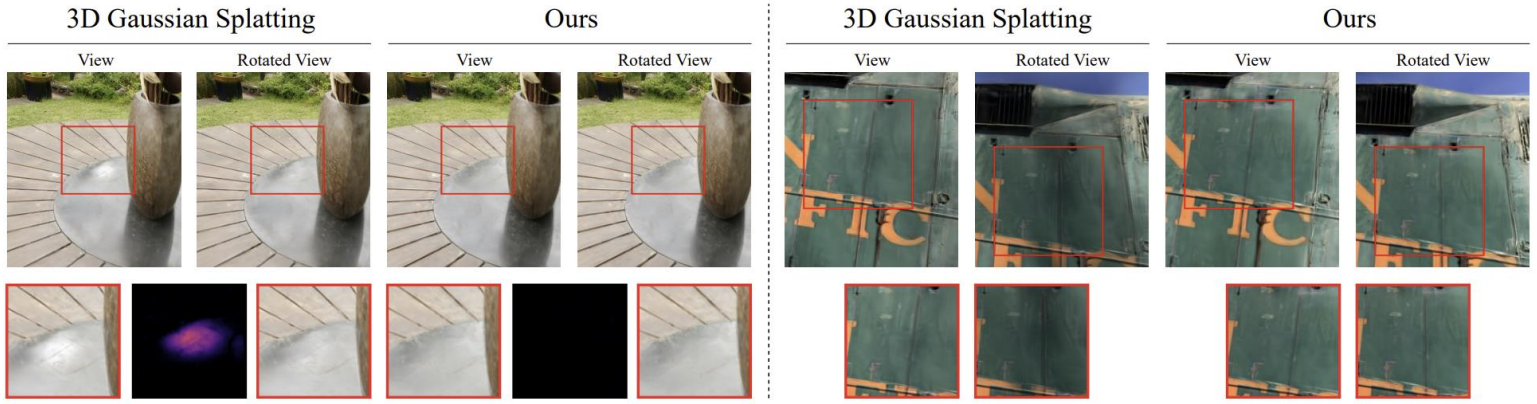
# Hierachical Sorting

| | | 3DGS | Full | Resorting Window | | | | Ours |
|---|---|---|---|---|---|---|---|---|
| | | | | 4 | 8 | 16 | 24 | |
| Train | $\delta_{max}$ | 28.445 | 0.000 | 5.867 | 3.882 | 3.544 | 4.580 | 0.575 |
| | $\delta_{avg}$ | 3.688 | 0.000 | 0.124 | 0.045 | 0.014 | 0.007 | 0.003 |
| | runtime | 1.00 | 142.03 | 1.21 | 1.66 | 2.70 | 4.22 | 0.92 |
| Bonsai | $\delta_{max}$ | 33.543 | 0.000 | 12.786 | 8.954 | 6.391 | 5.595 | 3.098 |
| | $\delta_{avg}$ | 3.786 | 0.000 | 0.265 | 0.110 | 0.039 | 0.019 | 0.006 |
| | runtime | 1.00 | 179.70 | 1.76 | 2.58 | 4.33 | 6.88 | 1.47 |

# StopThePop [Radl et al. 2024]

## StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering

LUKAS RADL* and MICHAEL STEINER*, Graz University of Technology, Austria
MATHIAS PARGER, Huawei Technologies, Austria
ALEXANDER WEINRAUCH, Graz University of Technology, Austria
BERNHARD KERBL, TU Wien, Austria
MARKUS STEINBERGER, Graz University of Technology, Austria and Huawei Technologies, Austria



# Popping

# Hybrid Transparency

- Alternative: Approximate sort with first K Gaussians [Hahlbohm et al., 2025]

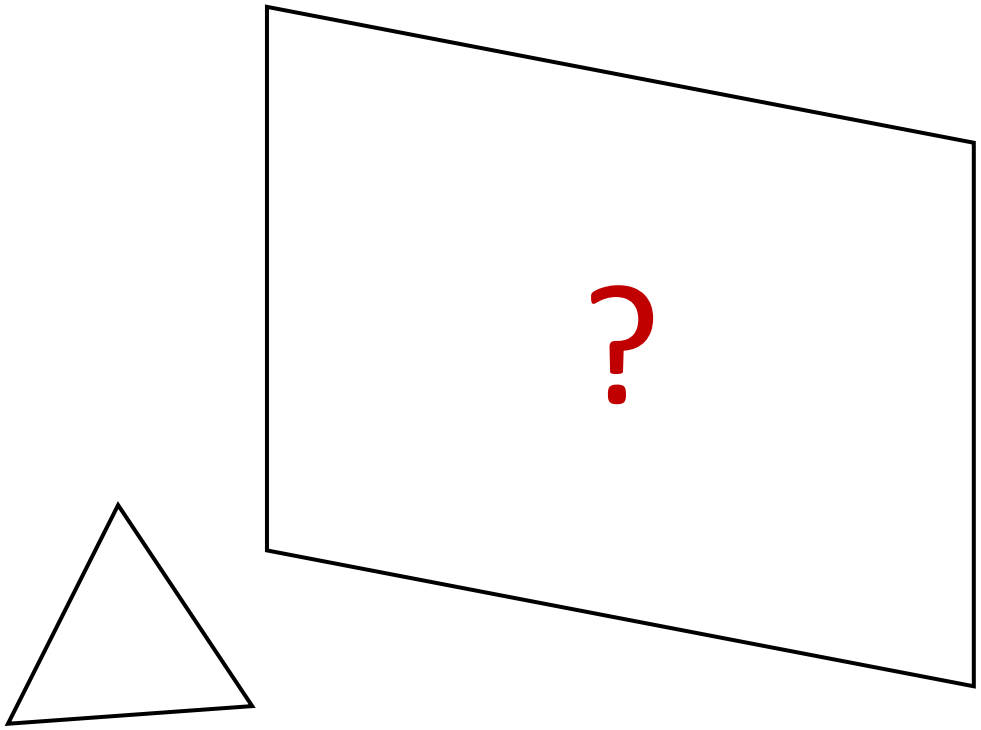
$$C = \sum_{i=1}^{K} \alpha_i T_i \boldsymbol{c}_i + T_{K+1}\left((1 - T_{\text{tail}})\boldsymbol{c}_{\text{tail}} + T_{\text{tail}}\boldsymbol{c}_{\text{bg}}\right)$$

- Quality – Speed trade-off
  - K = 16

# Distortions


Source: https://fhahlbohm.github.io/htgs/

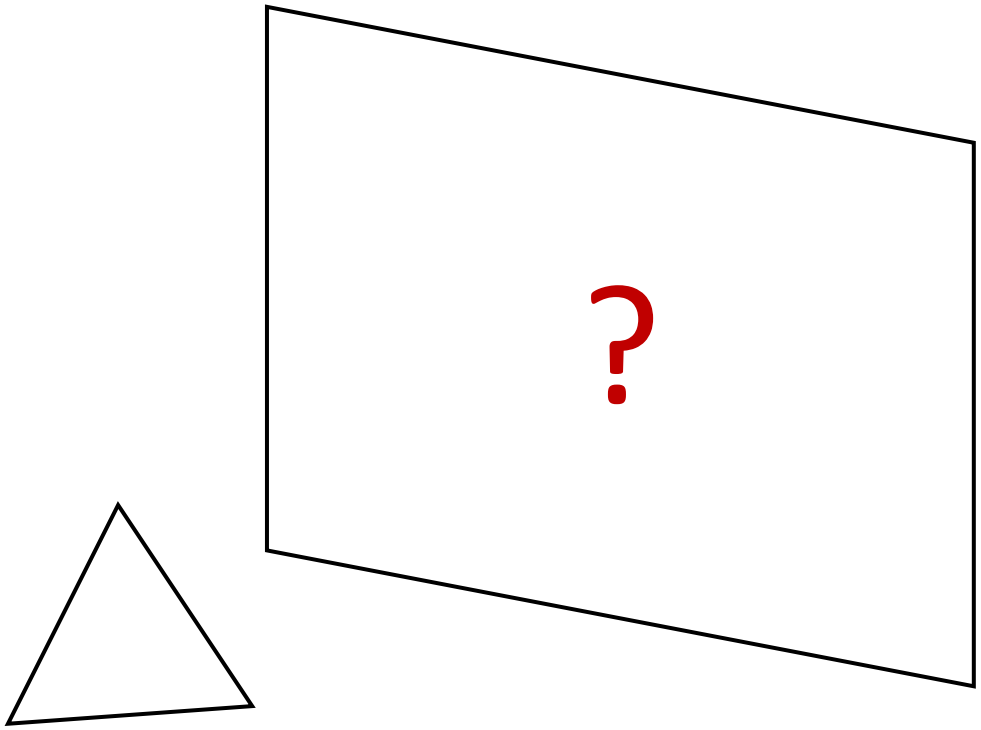
# Splatting




$$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)}$$

$$(x', y') = \left( \frac{f_x x}{z}, \frac{f_y y}{z} \right)$$

# Splatting

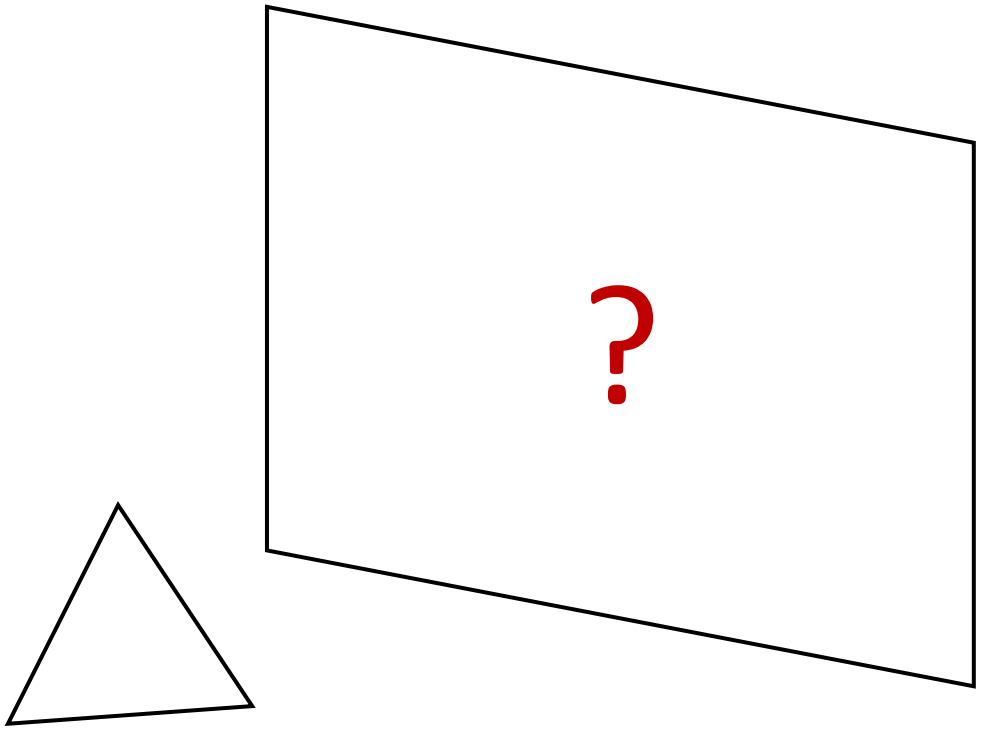


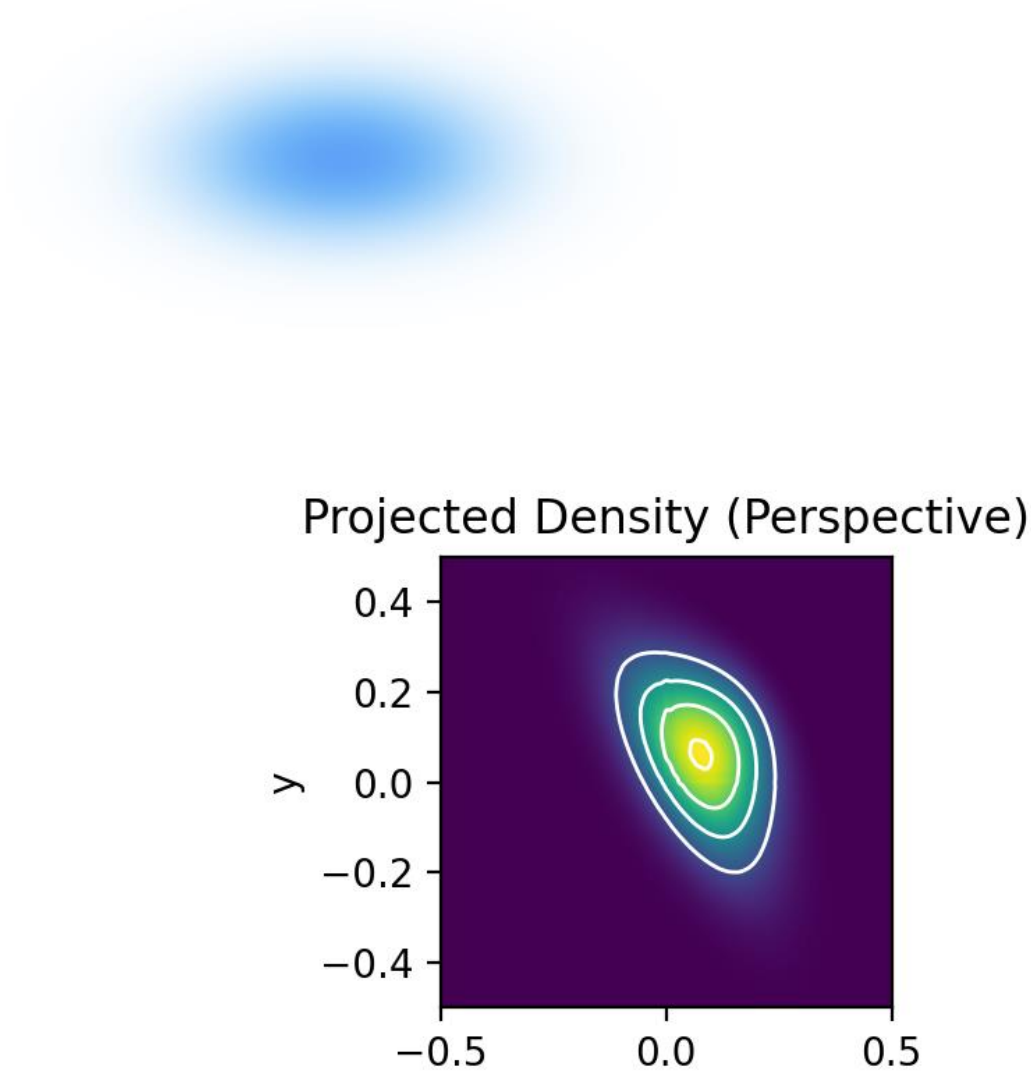

$$p(x,y) \propto \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) dz$$

$$I(x',y') = \int G\left(\frac{zx'}{f_x}, \frac{zy'}{f_y}, z\right) \left|\frac{\partial(x,y)}{\partial(x',y')}\right| dz$$

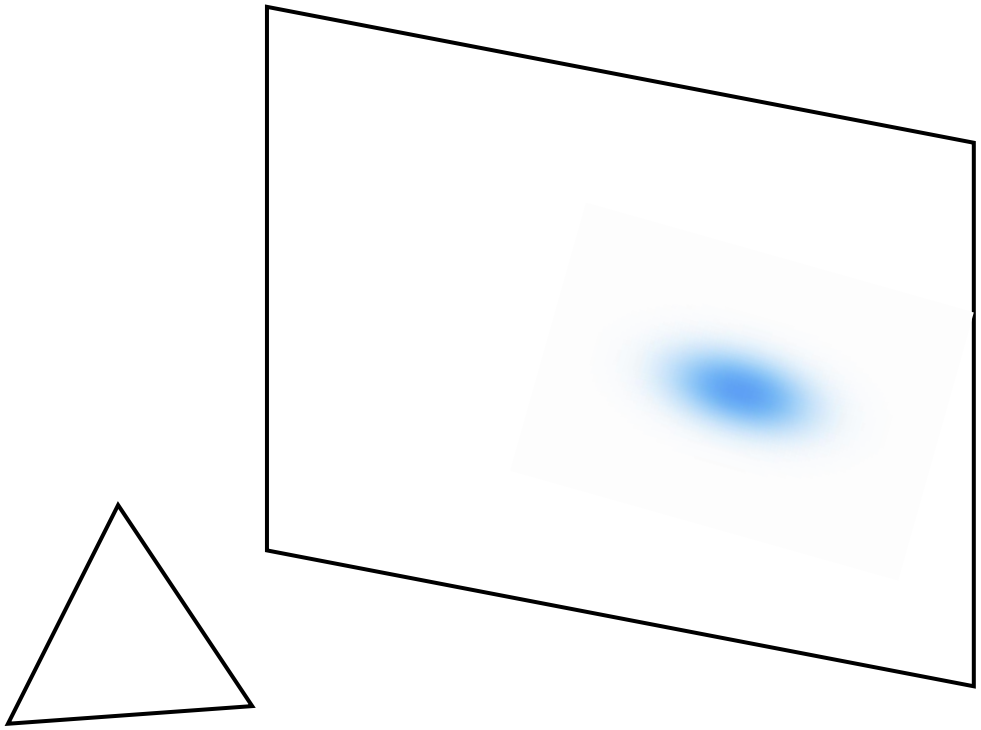
$$I(x',y') = \int G\left(\frac{zx'}{f_x}, \frac{zy'}{f_y}, z\right) \cdot \frac{z^2}{f_x f_y} dz$$

# Splatting



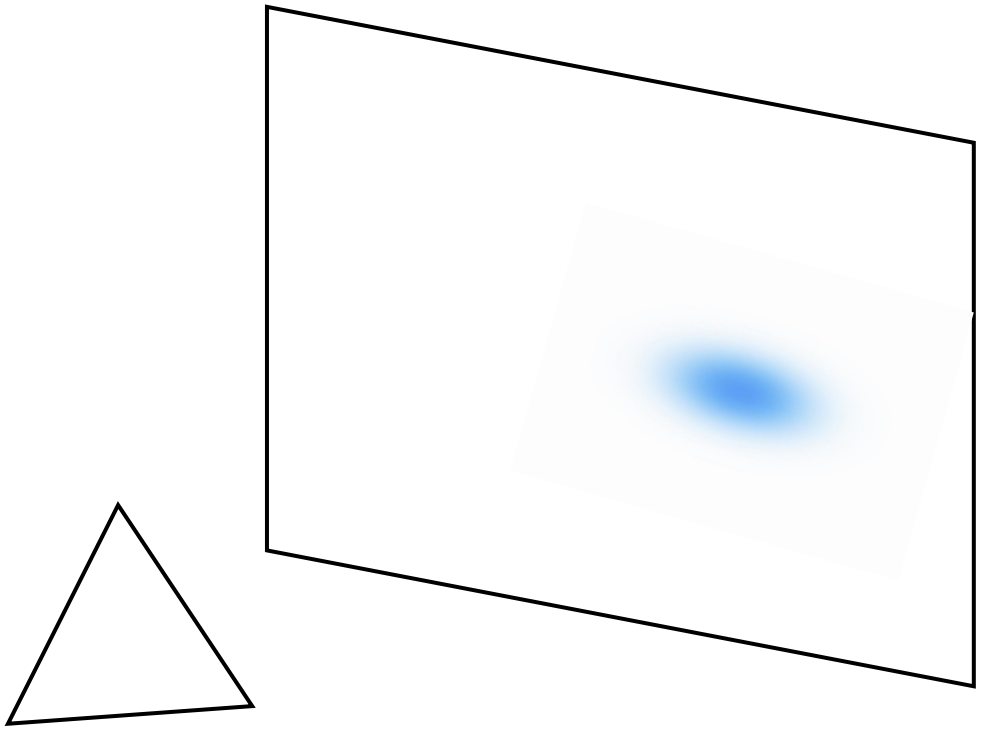
# Splatting

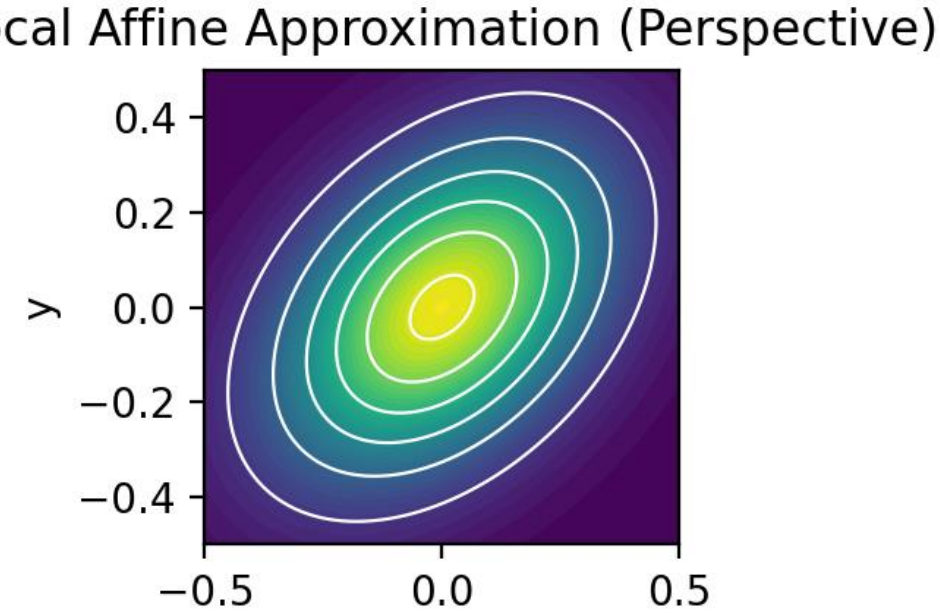


$$\Sigma' = JW\,\Sigma\,W^T J^T$$

$$J(\mu) = \begin{bmatrix} \frac{f_x}{\mu_Z} & 0 & -f_x \frac{\mu_X}{\mu_Z^2} \\ 0 & \frac{f_y}{\mu_Z} & -f_y \frac{\mu_Y}{\mu_Z^2} \end{bmatrix}$$

# Splatting

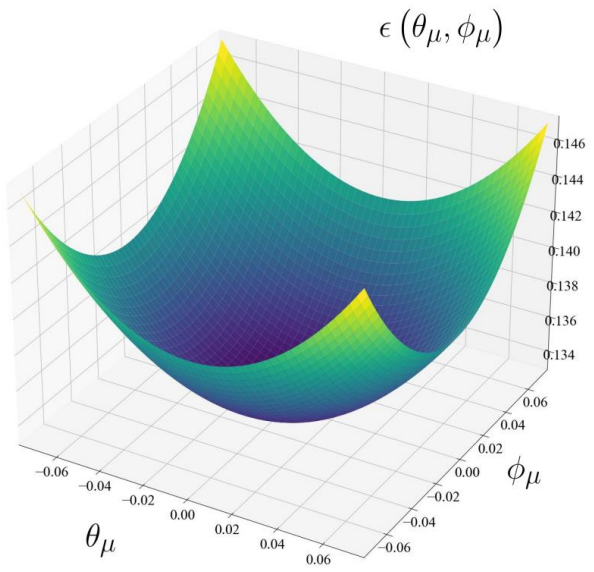



Local Affine Approximation (Perspective)
0.4
0.2
0.0
−0.2
−0.4
y
−0.5
0.0
0.5

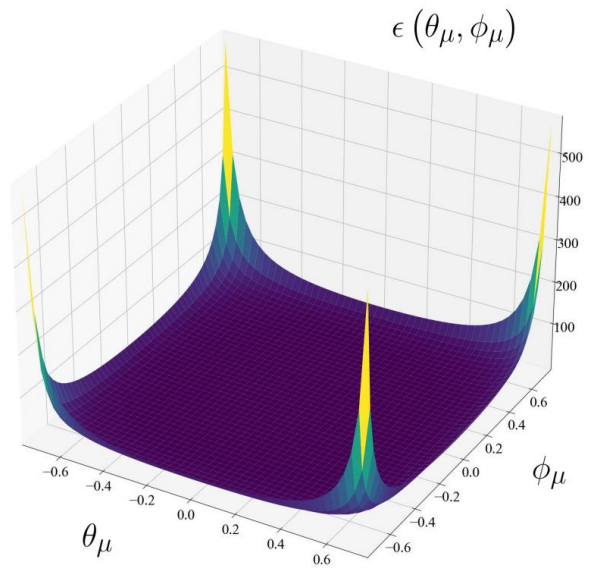
# Error Analysis

- Error increases at border
  [Huang et al. 2024]

- Effect increased by higher FOV

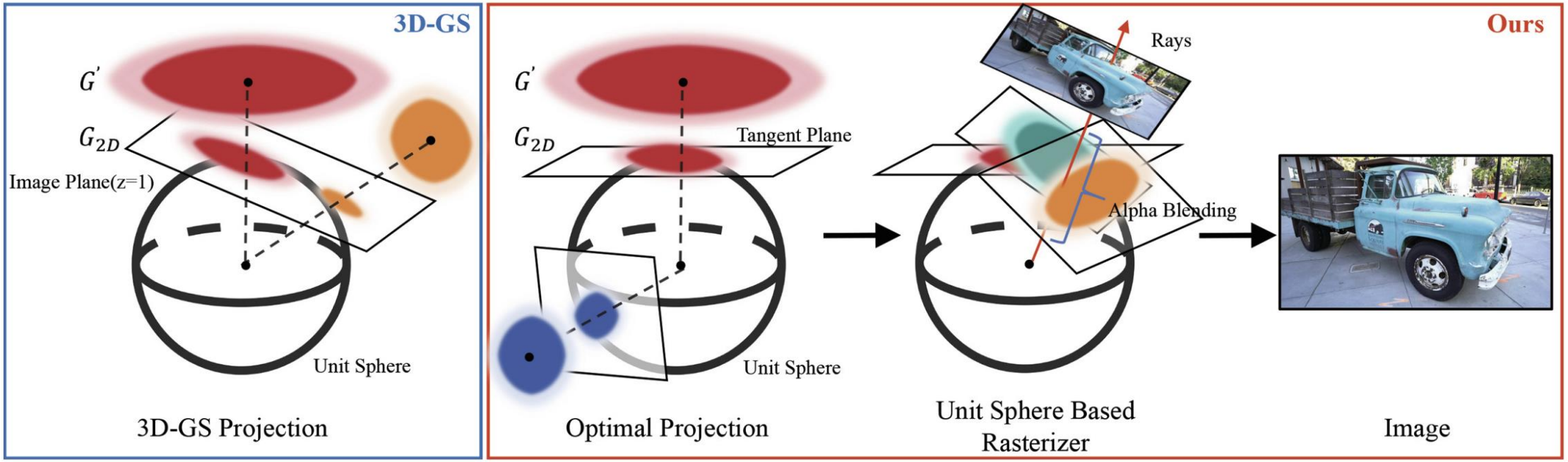



**(a)** $\lambda = 0.095$




**(b)** $\lambda = 0.95$

# Optimal Projection [Huang et al. 2024]



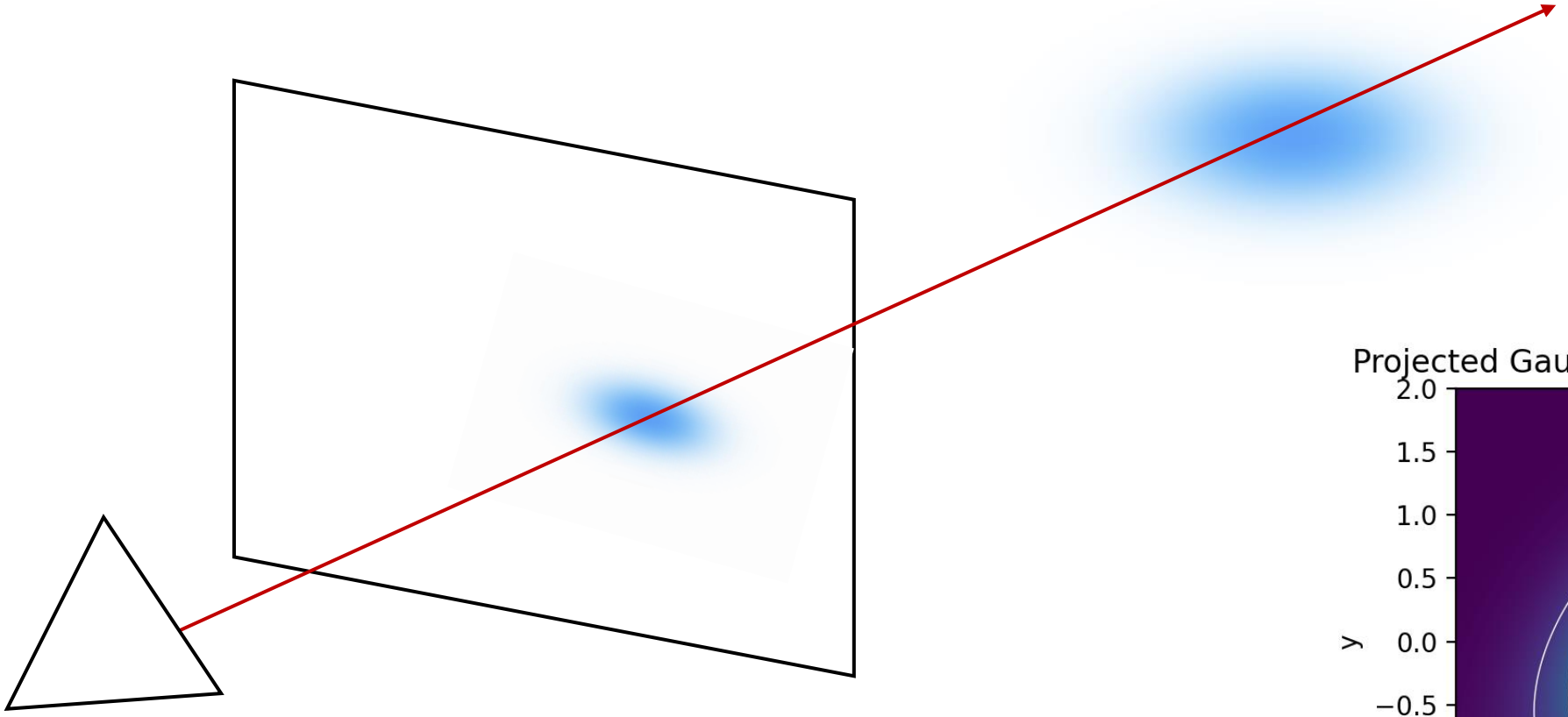
# VRSplat [Tu et al. 2025]

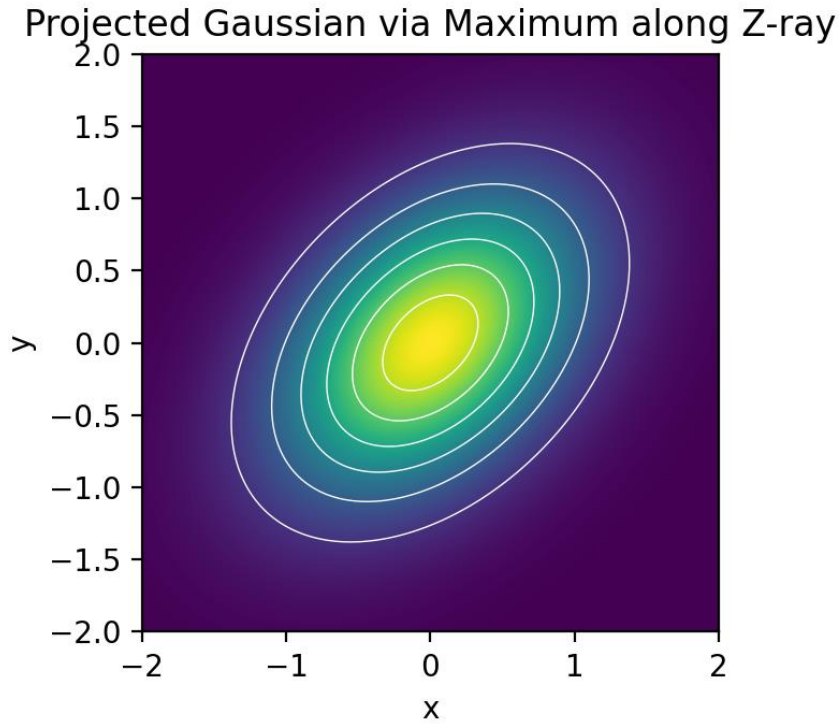
Check for updates

## VRSplat: Fast and Robust Gaussian Splatting for Virtual Reality

XUECHANG TU, Peking University, China
LUKAS RADL, Graz University of Technology, Austria
MICHAEL STEINER, Graz University of Technology, Austria
MARKUS STEINBERGER, Graz University of Technology, Austria and Huawei Technologies, Austria
BERNHARD KERBL, Carnegie Mellon University, USA
FERNANDO DE LA TORRE, Carnegie Mellon University, USA

# 3D Evaluation





# 3D Evaluation

3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes

Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for

v2 [cs.GR] 10 Aug 2024

] 31 Jan 2025

**Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency**

Florian Hahlbohm[1] Fabian Friederichs[1] Tim Weyrich[2,3] Linus Franke[2] Moritz Kappel[1]
Susana Castillo[1] Marc Stamminger[2] Martin Eisemann[1] Marcus Magnor[1,4]

[1]Computer Graphics Lab, TU Braunschweig, Germany {lastname}@cg.cs.tu-bs.de
[2] Visual Computing Erlangen, FAU Erlangen-Nürnberg, Germany {firstname.lastname}@fau.de
[3]University College London, United Kingdom [4]University of New Mexico, USA

https://fhahlbohm.github.io/htgs/

# 3D Evaluation

- Volumetric consistent
  - Self occlusion
  - Analytically not possible (overlap)

**Does 3D Gaussian Splatting Need Accurate Volumetric Rendering?**

A. Celarek[1], G. Kopanas[2,3,4], G. Drettakis[3,4], M. Wimmer[1] and B. Kerbl[1]

[1]TU Wien, Austria, [2]Google, United Kingdom, [3]Inria, France, [4]Université Côte d'Azur, France





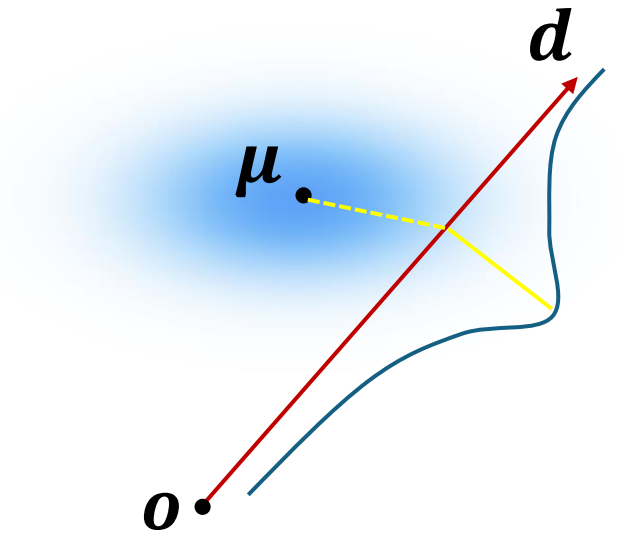
**Figure 1:** LEGO *scene from the NeRF-synthetic dataset.* ***Left****: 4k Gaussians, rendered with 3D Gaussian Splatting and with extinction-based volume ray marching.* ***Right****: 100k Gaussians with the same two techniques. While the more principled ray-marching technique yields superior quality for fewer Gaussians, this benefit vanishes in qualitative and quantitative assessment when increasing their number.*

# 3D Evaluation

- Point sampling
  - Easy to implement
  - Avoids projection error

- Where to sample?

$$\tau_{max} = \frac{(\boldsymbol{\mu} - \boldsymbol{o})^{\mathrm{T}} \Sigma^{-1} \boldsymbol{d}}{\boldsymbol{d}^{T} \Sigma^{-1} \boldsymbol{d}}$$



# 3D Evaluation

# Aliasing

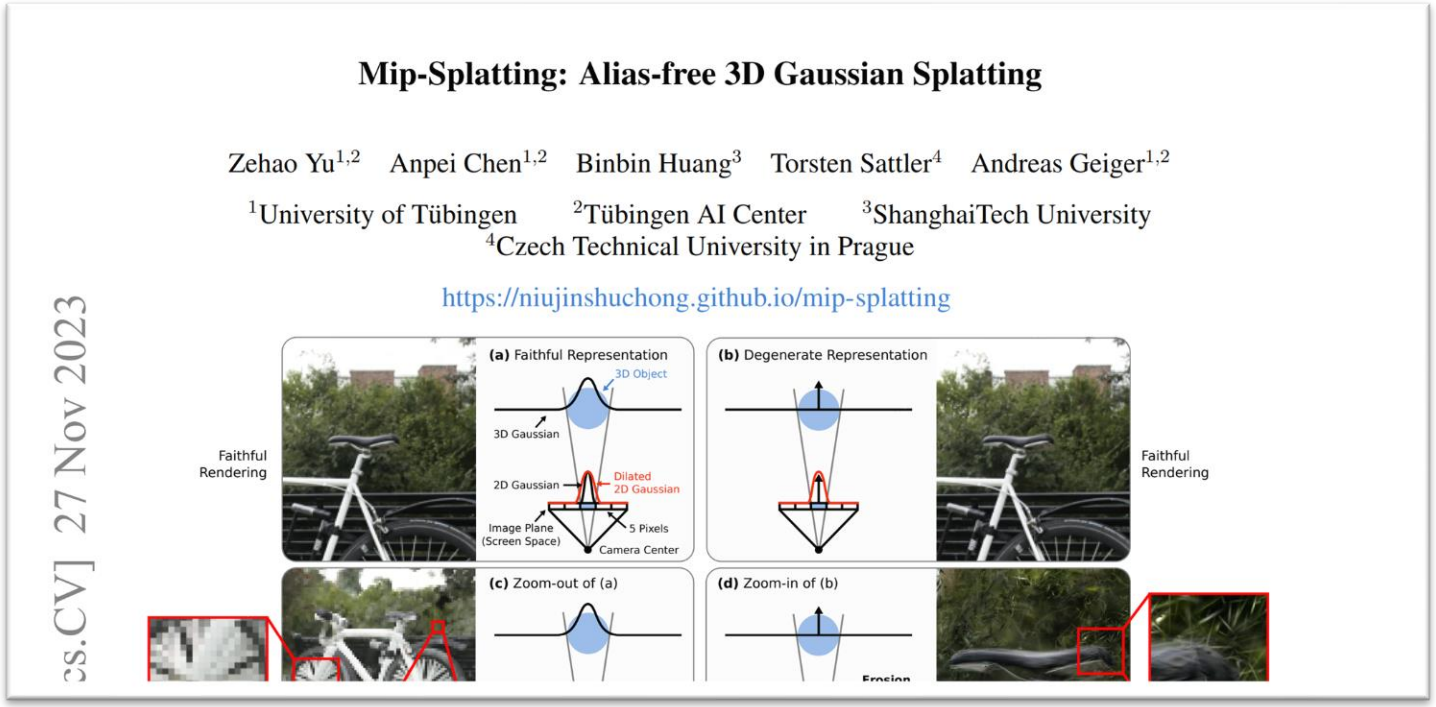
# Aliasing

- 3DGS has projected 2D Gaussian
  - Screen-space filter
  - Ensures minimum screen extent → no aliasing

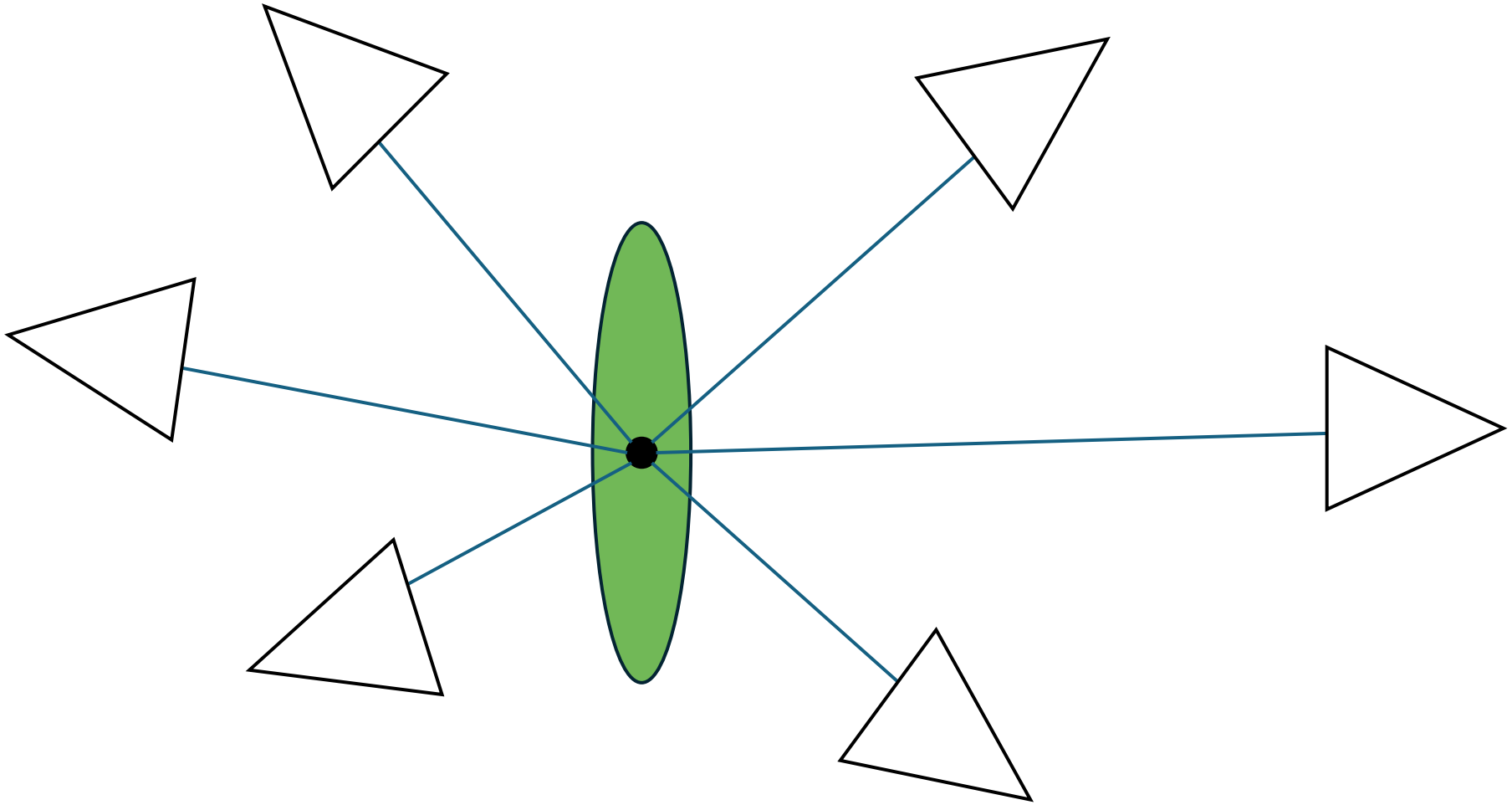
- 3D Evaluation
  - Not as easy

# Mip-Splatting [Yu et al. 2024]

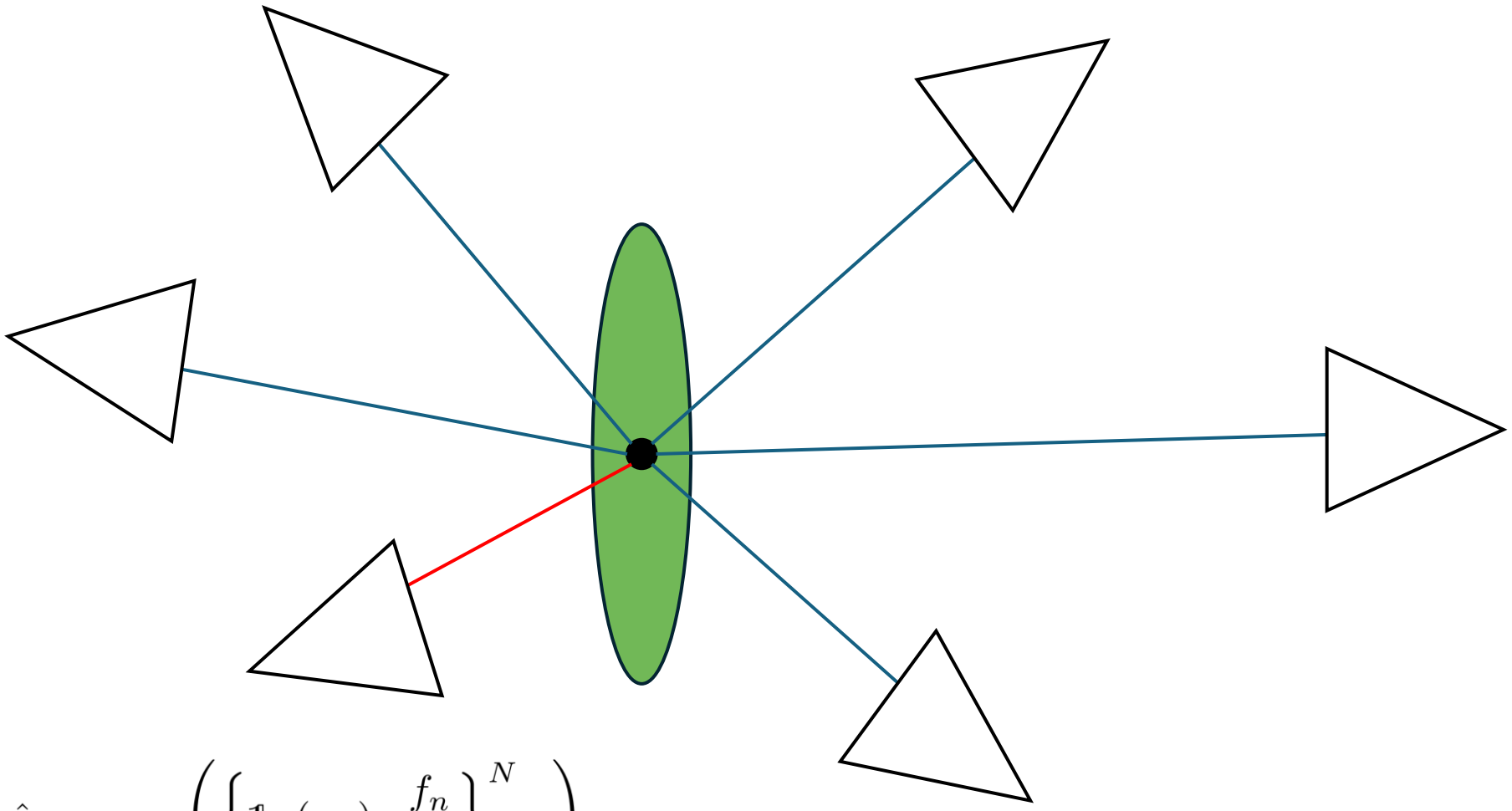
- 2D screen-space filter
  - Not viable for 3D evaluation
- 3D world space filter
  - Used in multiple works



# 3D Mip-Filter



# 3D Mip-Filter



$$\hat{\nu}_k = \max\left(\left\{\mathbb{1}_n(\mathbf{p}_k)\cdot\frac{f_n}{d_n}\right\}_{n=1}^{N}\right)$$

# Mip-Splatting [Yu et al. 2024]

$$\mathcal{G}_k(\mathbf{x})_{\text{reg}} = (\mathcal{G}_k \otimes \mathcal{G}_{\text{low}})(\mathbf{x})$$

$$\mathcal{G}_k(\mathbf{x})_{\text{reg}} = \sqrt{\frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_k + \frac{s}{\hat{\nu}_k} \cdot \mathbf{I}|}}\, e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_k)^T (\mathbf{\Sigma}_k + \frac{s}{\hat{\nu}_k} \cdot \mathbf{I})^{-1} (\mathbf{x}-\mathbf{p}_k)}$$

- At least one camera during training
- Fixed after training!

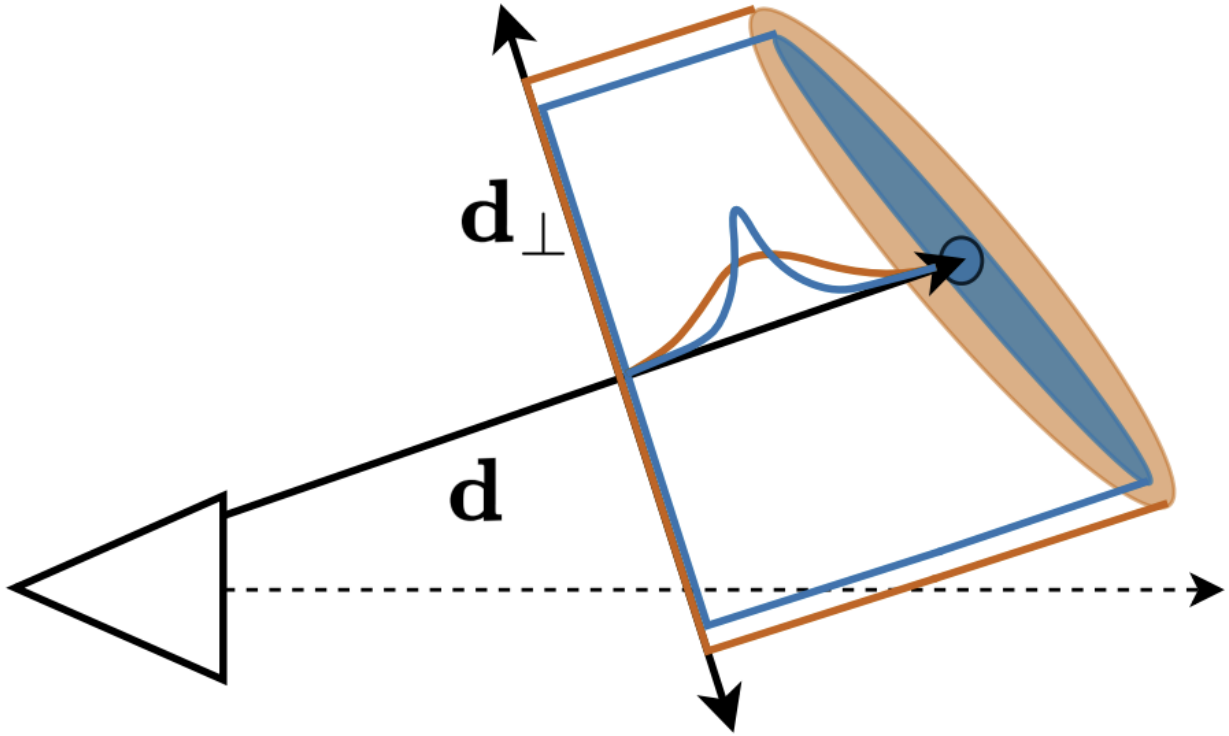
# Fixed 3D Filter

# Adaptive 3D Filter

- Recompute 3D filter for each rendering view
- Normalizes total volume $\sqrt{\frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_k + \frac{s}{\hat{\nu}_k} \cdot \mathbf{I}|}}$
- Point sampling does consider volume
  - Single point per Gaussian
  - Over-regularization

# Adaptive 3D Filter

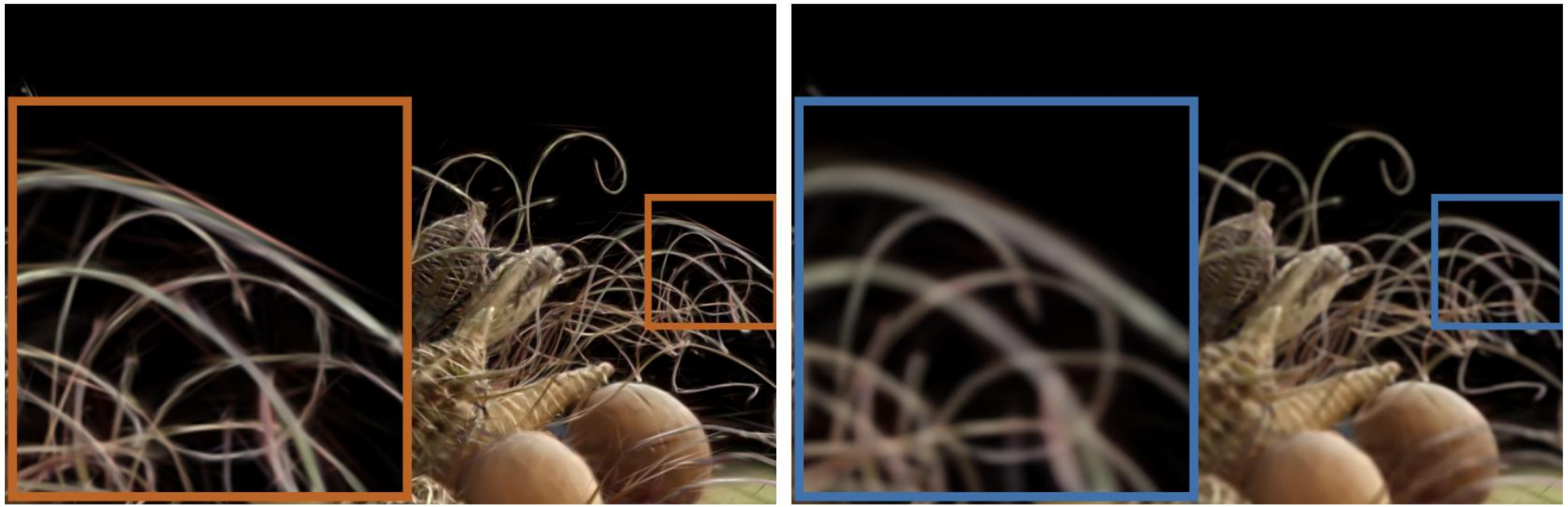
- Only consider change perpendicular to d
- Avoids over-regularization




$$\sqrt{\frac{|\boldsymbol{\Sigma}_\perp|}{|\hat{\boldsymbol{\Sigma}}_\perp|}} = \sqrt{\frac{|\boldsymbol{\Sigma}|\ \mathbf{d}^\top \boldsymbol{\Sigma}^{-1}\mathbf{d}}{|\hat{\boldsymbol{\Sigma}}|\ \mathbf{d}^\top \hat{\boldsymbol{\Sigma}}^{-1}\mathbf{d}}} = \sqrt{\frac{\mathbf{d}_1'^2\ \mathbf{s}_2^2\ \mathbf{s}_3^2 + \mathbf{d}_2'^2\ \mathbf{s}_1^2\ \mathbf{s}_3^2 + \mathbf{d}_3'^2\ \mathbf{s}_1^2\ \mathbf{s}_2^2}{\mathbf{d}_1'^2\ \hat{\mathbf{s}}_2\ \hat{\mathbf{s}}_3 + \mathbf{d}_2'^2\ \hat{\mathbf{s}}_1\ \hat{\mathbf{s}}_3 + \mathbf{d}_3'^2\ \hat{\mathbf{s}}_1\ \hat{\mathbf{s}}_2}}$$

# Over-Sampling

- Gaussians too small in close out-of-distribution views
  - Bound with max training frequency (Mip-Splatting)



# Adaptive Filter

# AAA-Gaussians [Köhler et al. 2025]



**AAA-Gaussians: Anti-Aliased and Artifact-Free 3D Gaussian Rendering**

Michael Steiner*[1] Thomas Köhler*[1] Lukas Radl[1]
Felix Windisch[1] Dieter Schmalstieg[1,2] Markus Steinberger[1]
[1]Graz University of Technology [2]University of Stuttgart

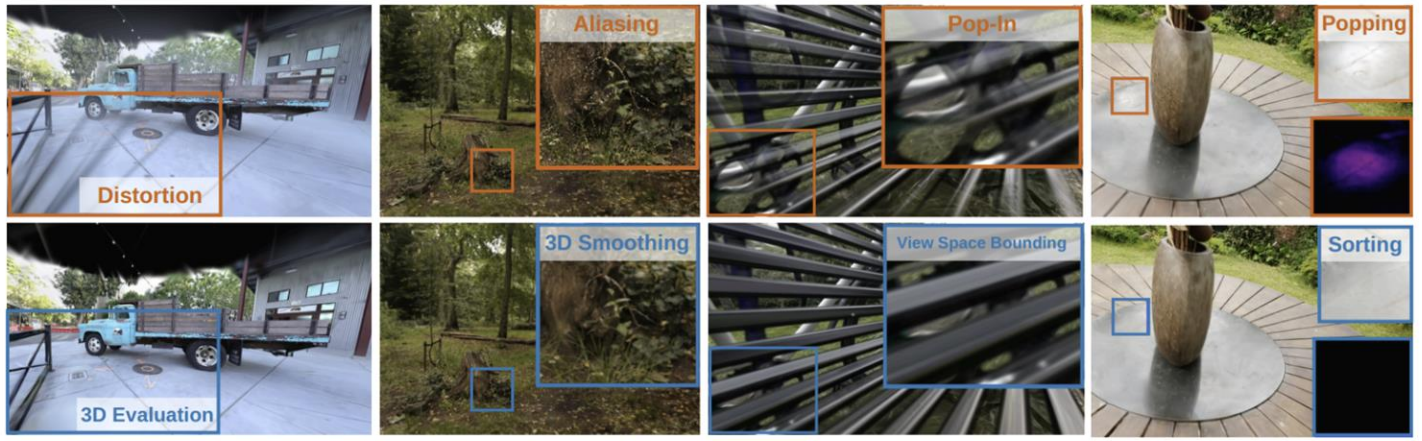
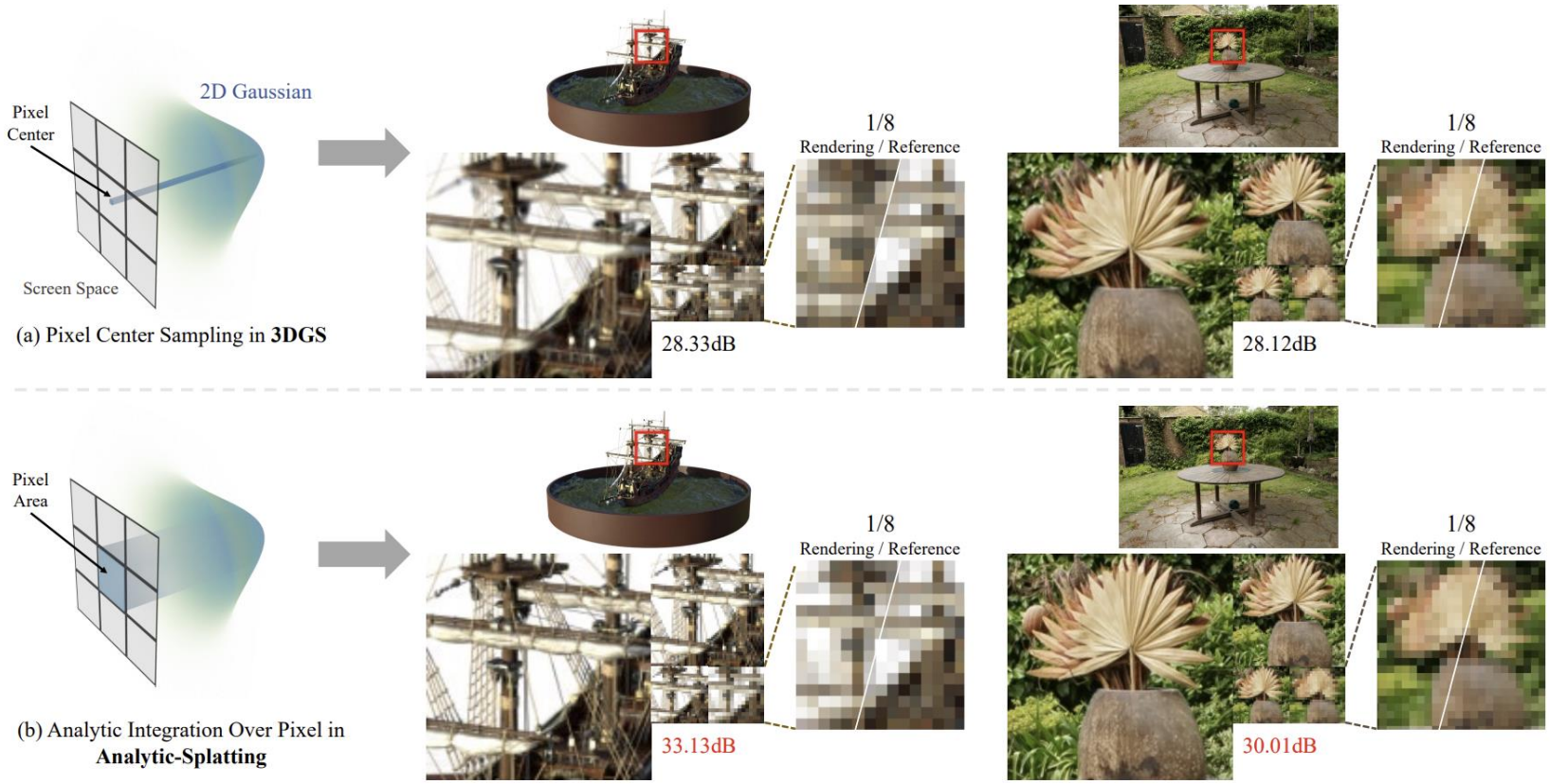



Figure 1. (Top row) 3DGS rasterization approaches encounter artifacts in out-of-distribution camera settings: (1) Distortions from 2D splat approximations in large field-of-view renderings. (2) 3D evaluation specific aliasing artifacts when zooming out. (3) Incorrect culling results in screen space when the camera is close to objects. (4) Popping due to depth simplifications and global sorting. (Bottom row) Our method addresses these issues with: (1) 3D Gaussian evaluation, (2) a correct aliasing filter, adapted specifically to Gaussian evaluation in 3D, (3) accurate and robust bounding, and (4) efficient 3D culling integrated into hierarchical sorting.

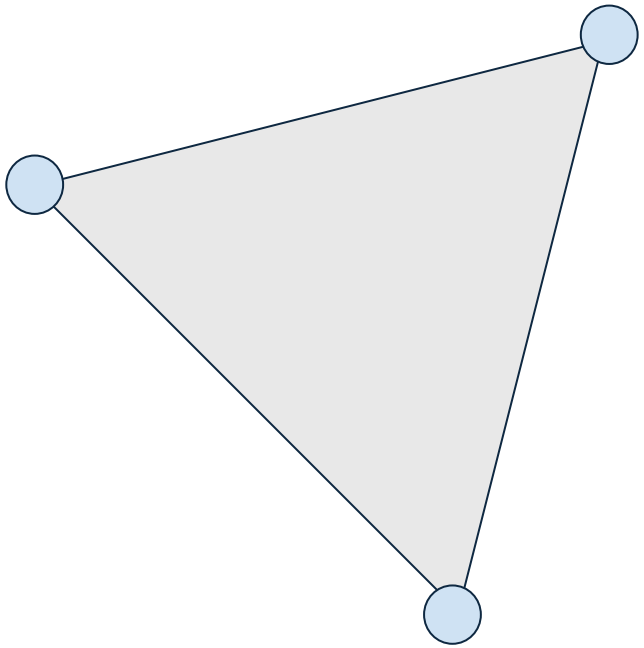
# Analytic-Splatting [Liang et al. 2024]



# Other Artifacts

- Floaters
- Holes
- Depth imprecision (Sky)
- Over-blur


Mesh Extraction from 3DGS



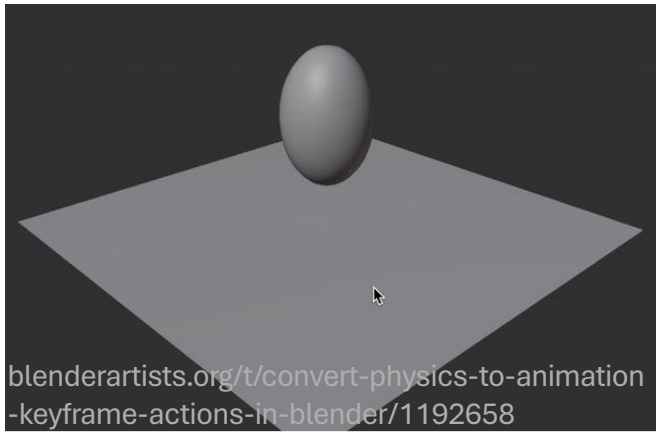
*Every triangle is a love triangle when you love triangles.*

Pythagoras, probably

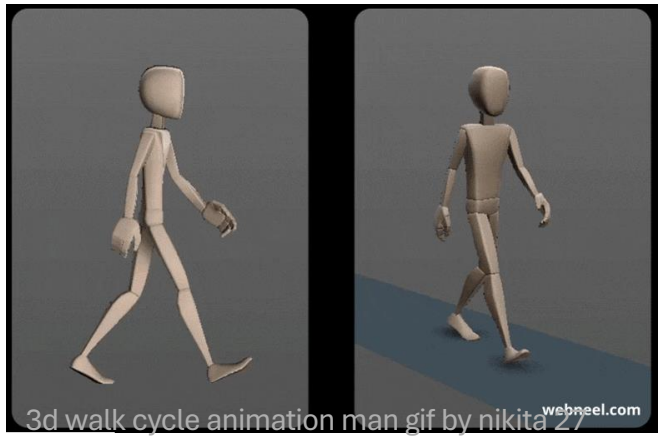


[James Acaster]

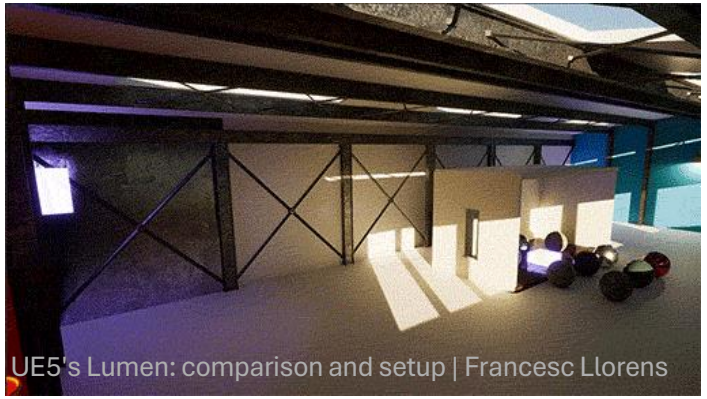
# Why convert 3DGS to Mesh?



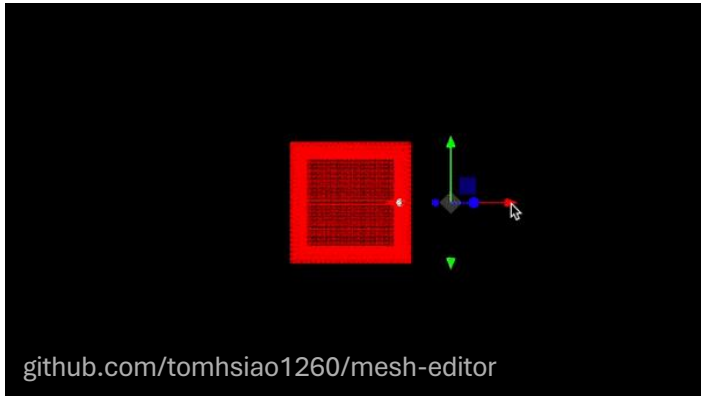

Simulation




Animation

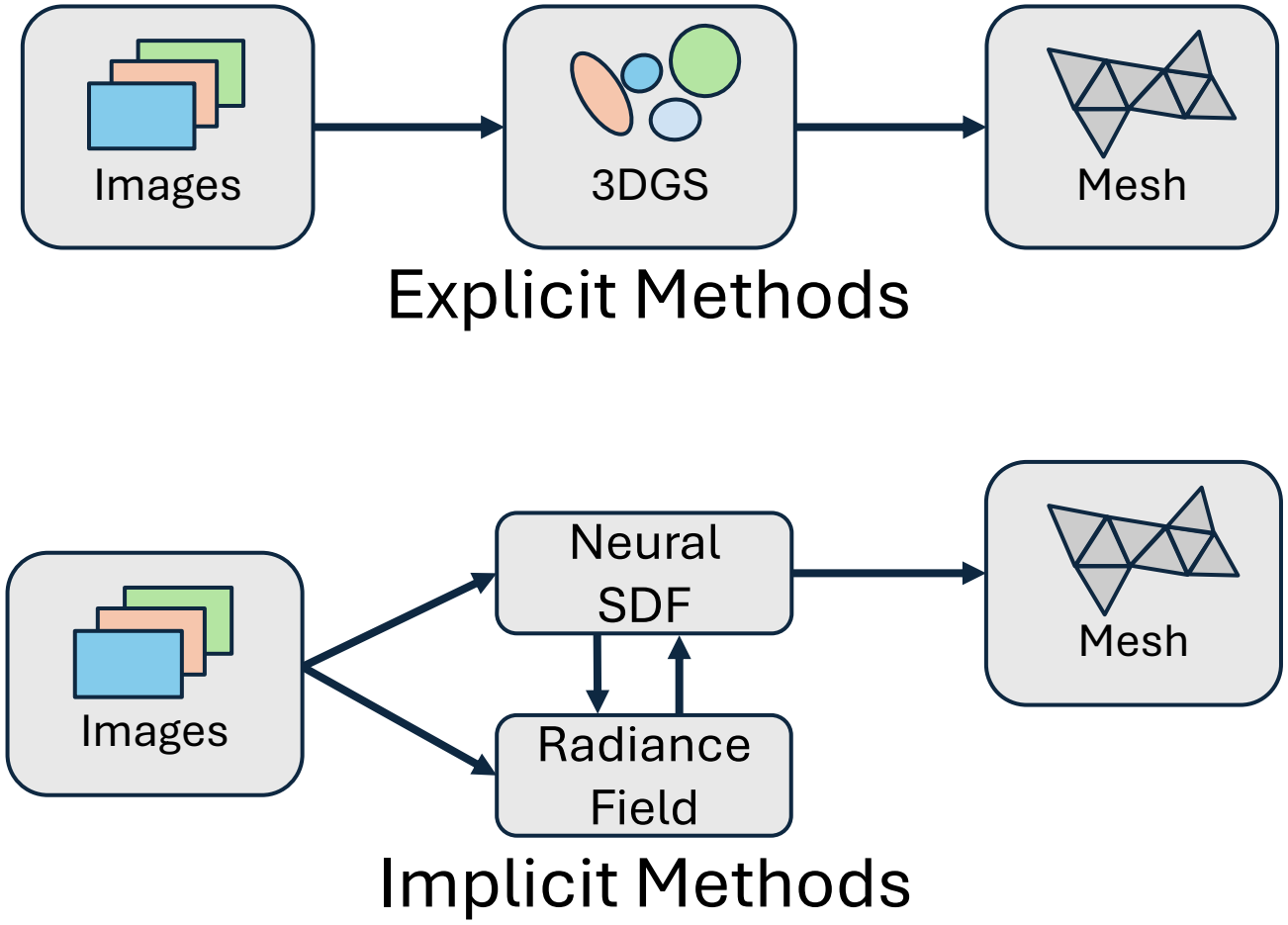



Re-Lighting

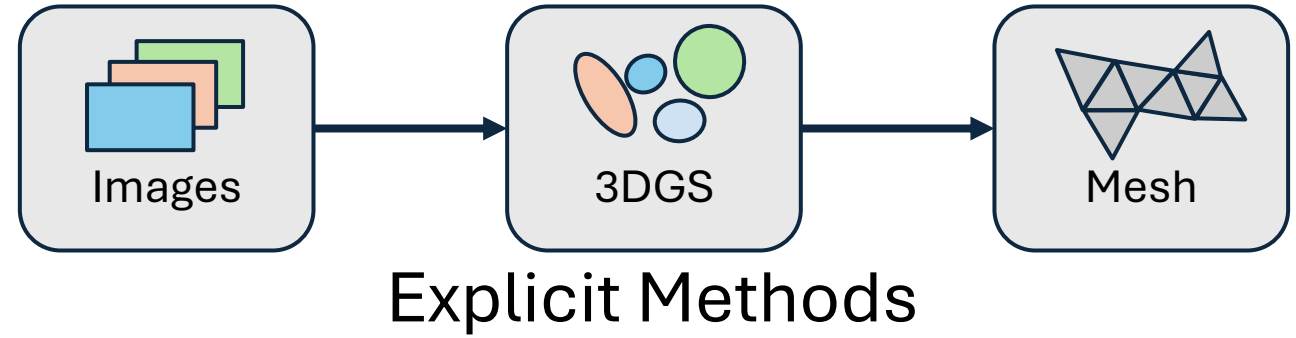



Editing

# Disambiguation

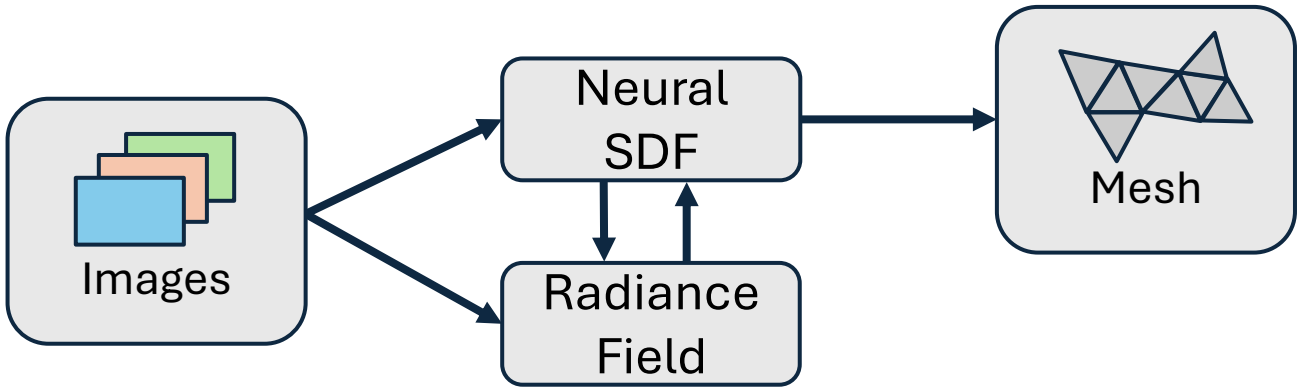


# Disambiguation




Explicit Methods

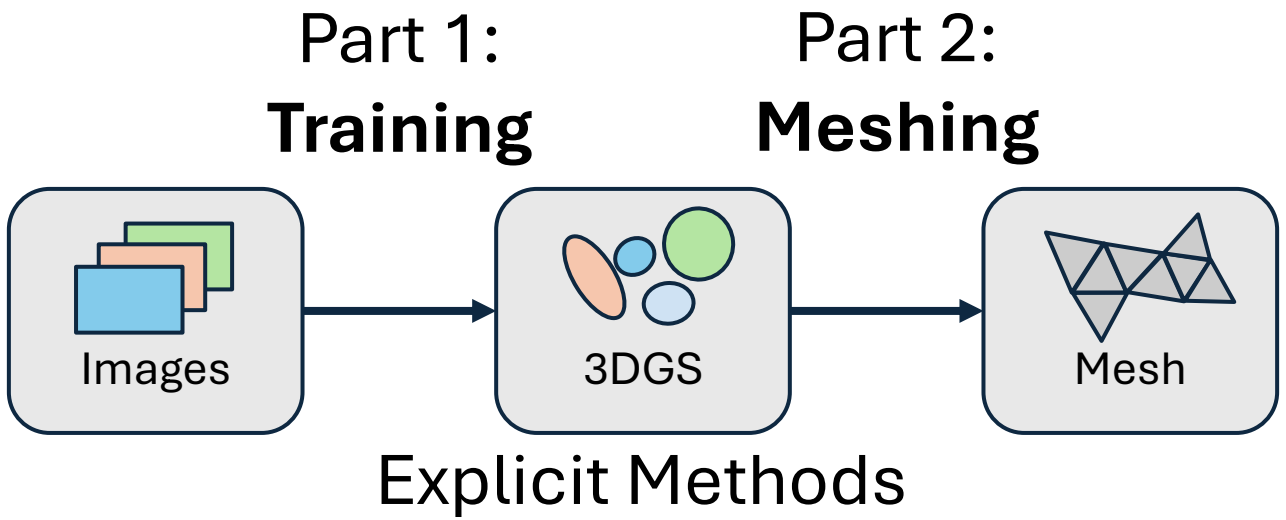
~Minutes



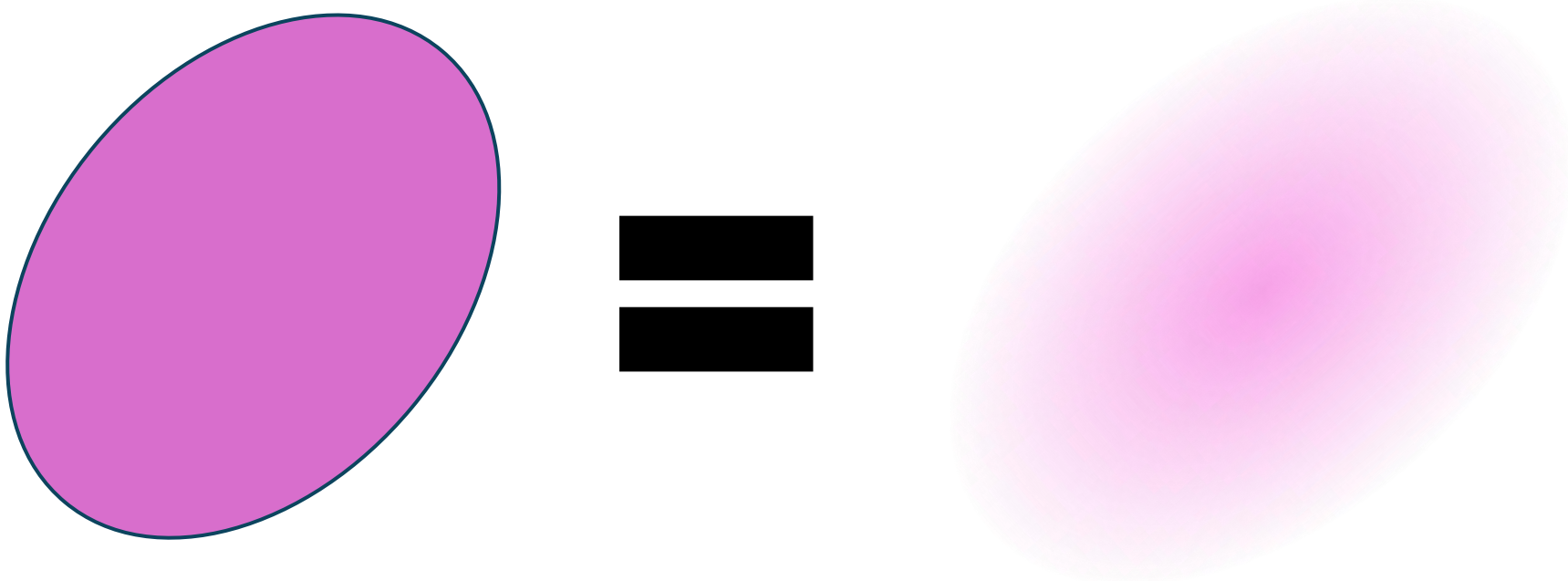

Implicit Methods

~Hours

# Disambiguation



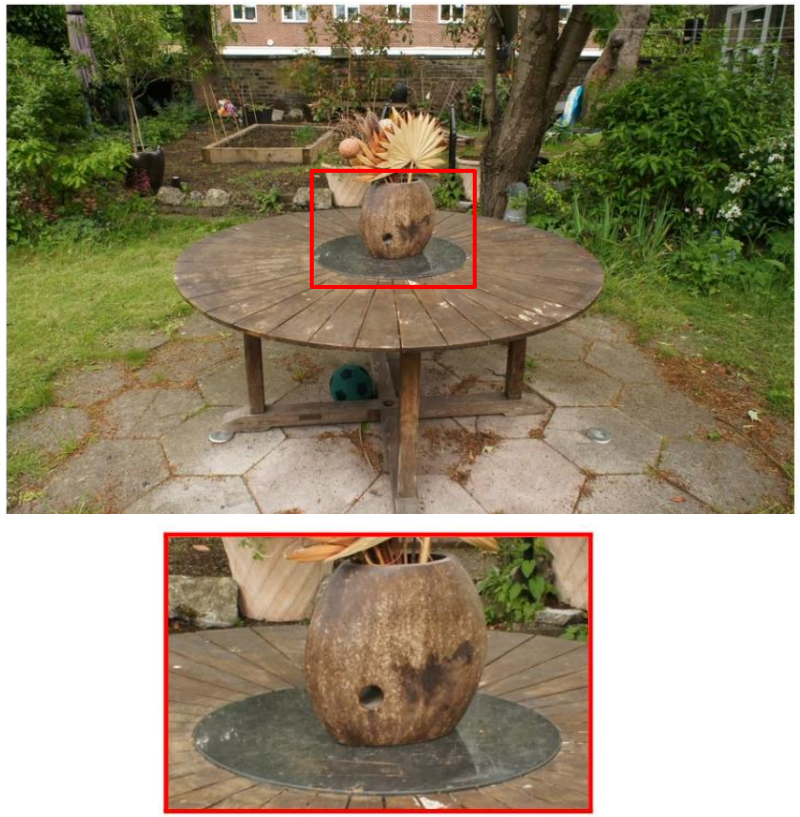
# Gaussians and Ellipsoids



# Part 1: Training

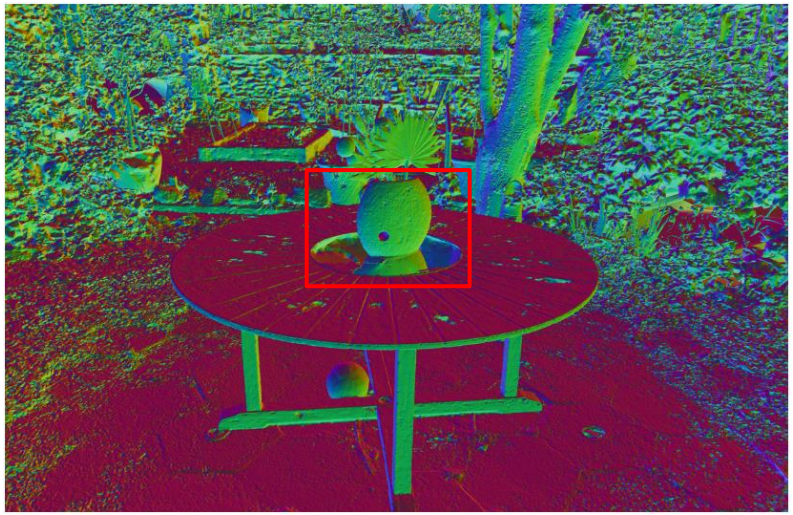
# Part 1 Training

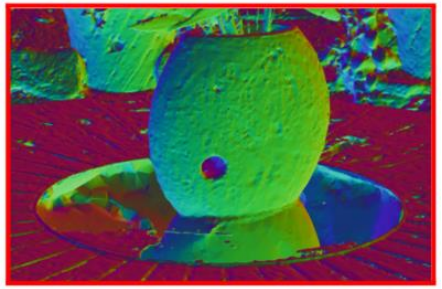
# Part 1: Training





3D Gaussians



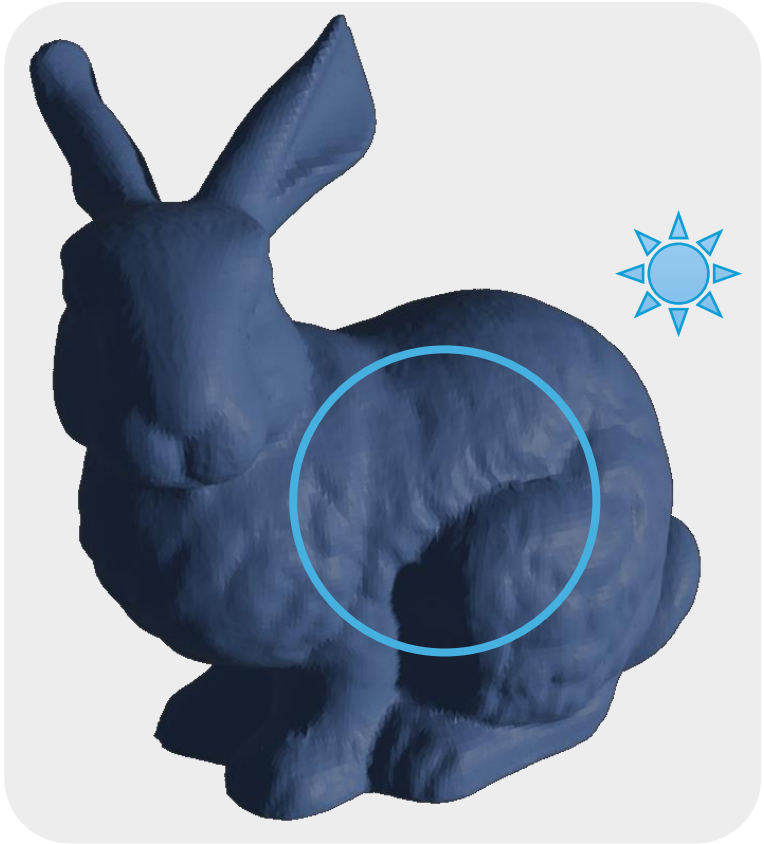


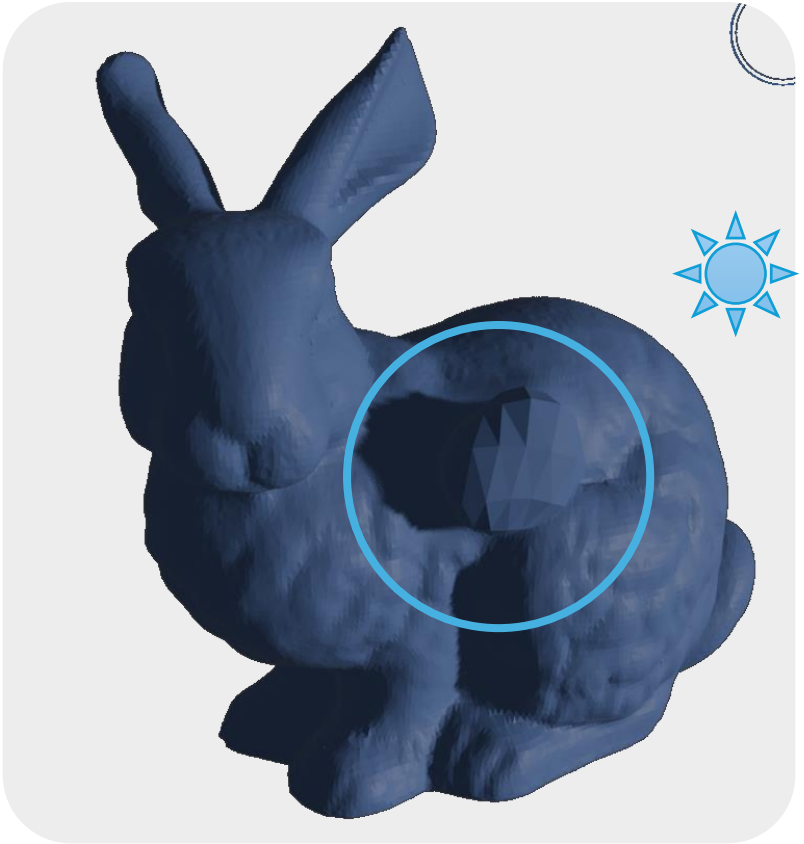
Mesh



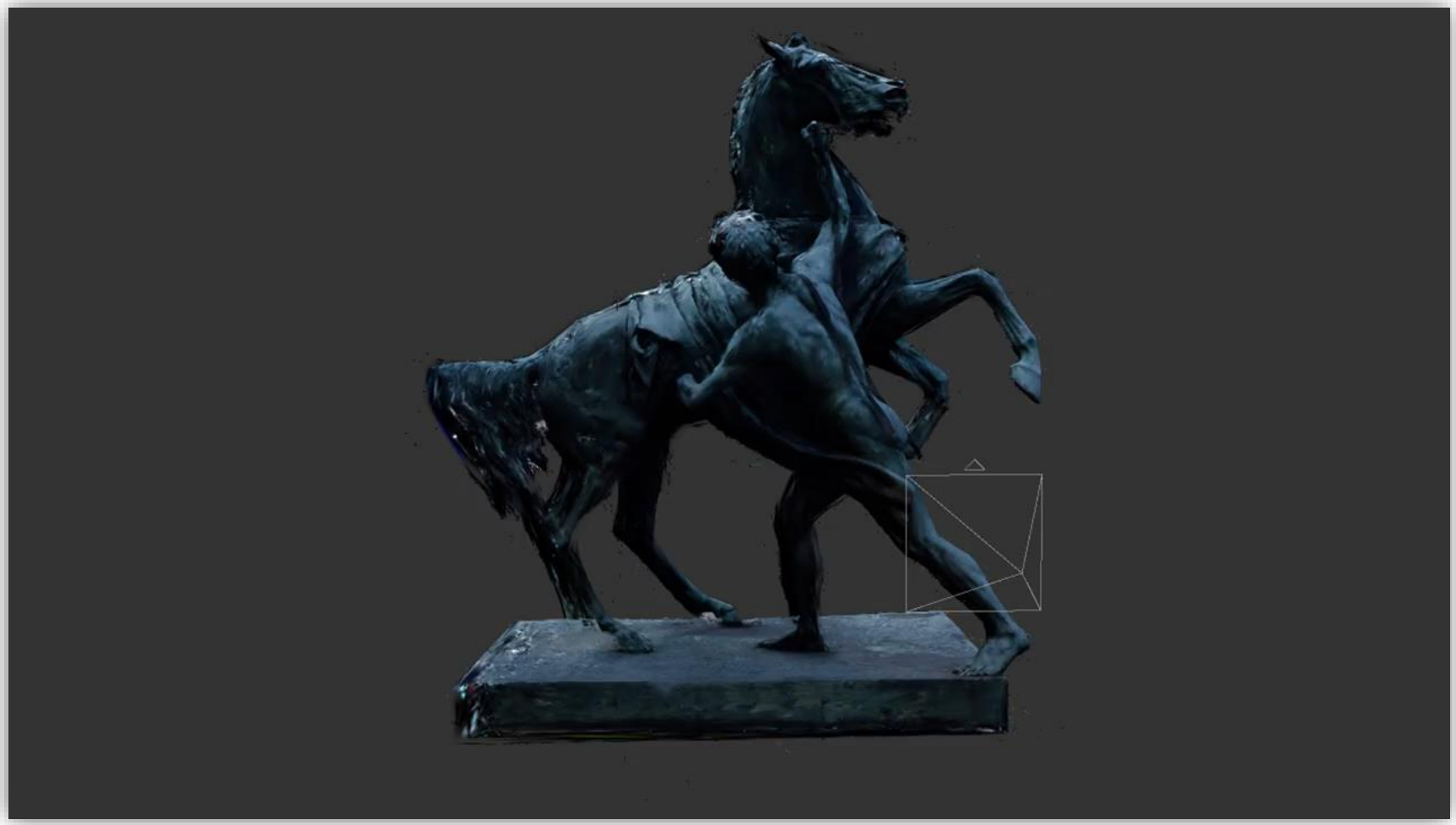
[Radl et al. 2025] SOF: Sorted Opacity Fields for Fast Unbounded Surface Reconstructio

# Part 1: Training





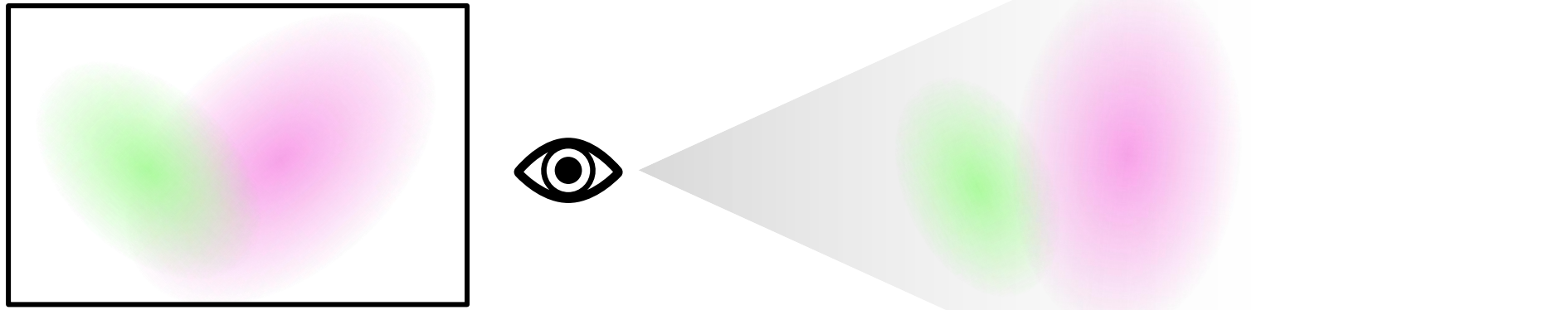
# 3D Gaussians are blobs of light

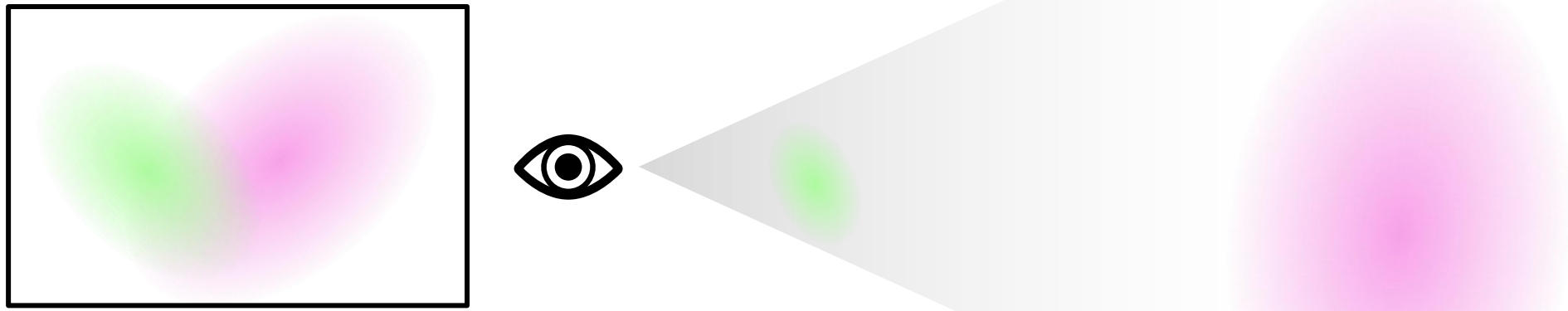


[Oli Huttonen on Youtube: Minimalist Gaussian Splatting: How Few Camera Views Do You Really

# 3D Gaussians are blobs of light

# 3DGS is underconstrained





# Constraining 3DGS

$$\mathcal{L} = \underbrace{\lambda_1 \mathcal{L}_{D-SSIM} + (1 - \lambda_1)\mathcal{L}_1}_{Visual\ \ Accuracy}$$

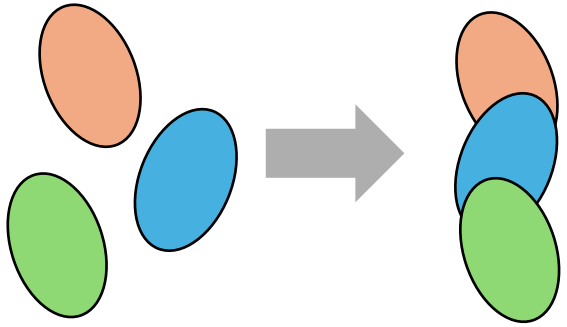
We have ground truth images, but no ground truth geometry

→ Rely on Priors (More Surface, Less Volume)

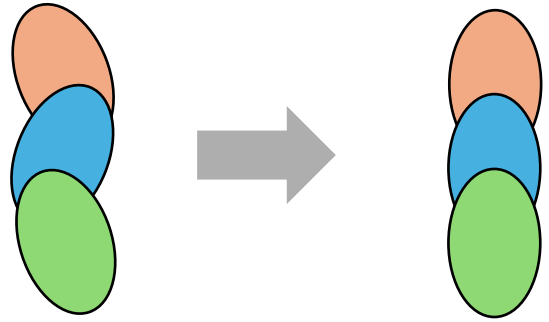
Tradeoff between Geometric Accuracy and Performance/Fidelity

# Geometric Losses

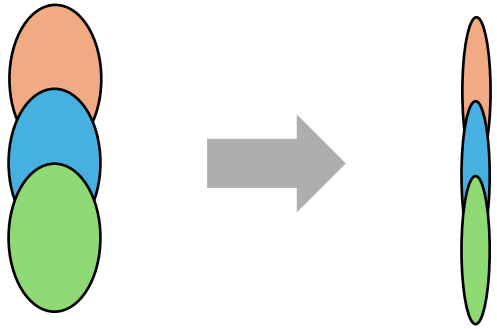
$$\mathcal{L}_{geometry} = \mathcal{L}_{dist} + \mathcal{L}_{normal} + \mathcal{L}_{flat} + \mathcal{L}_{multiview}$$



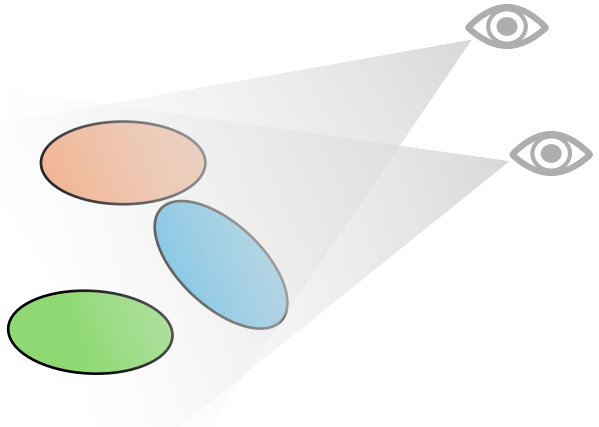
**1. Depth Distortion Loss**



2. Normal Consistency Loss

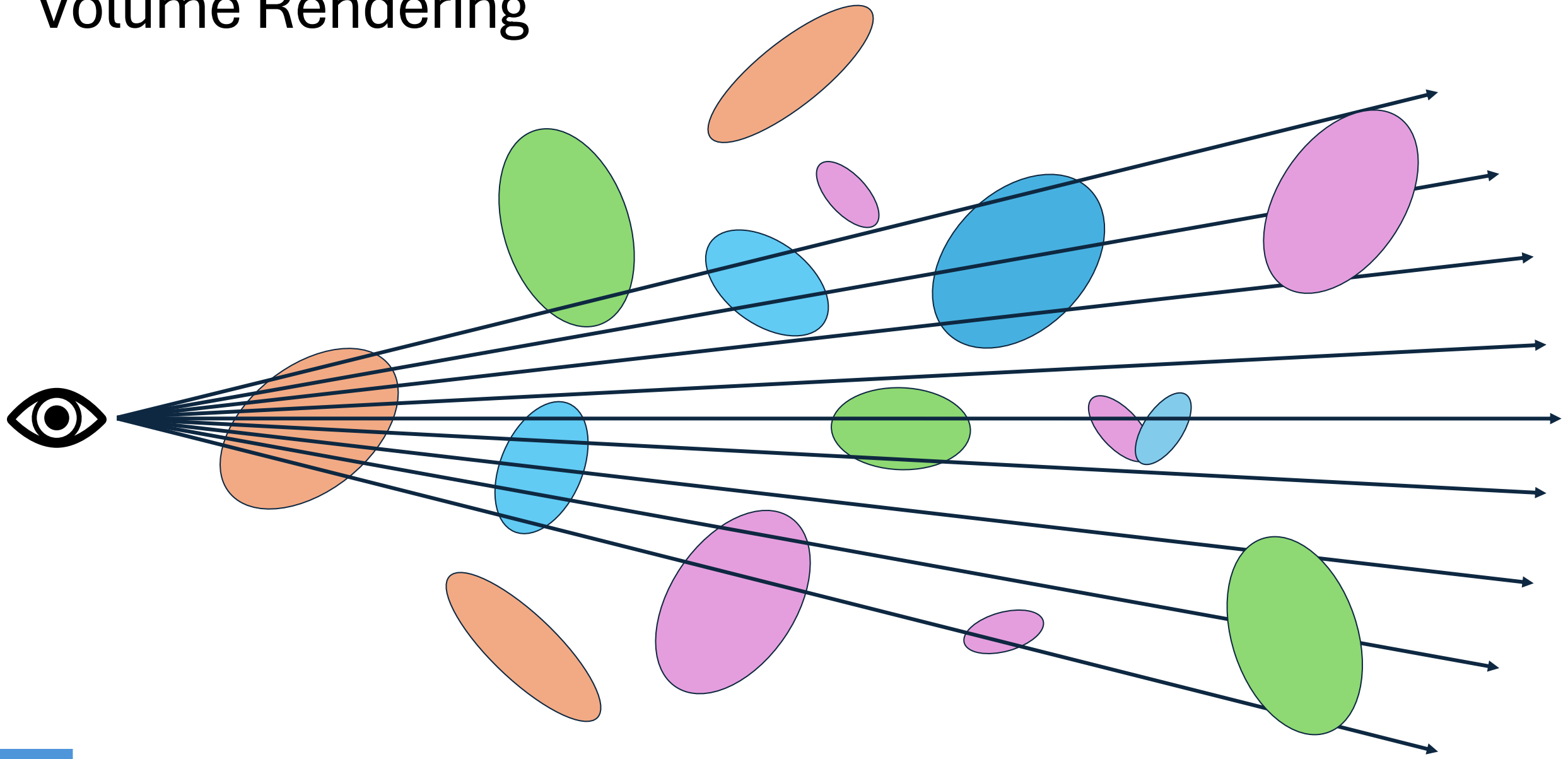


3. Flattening Gaussians

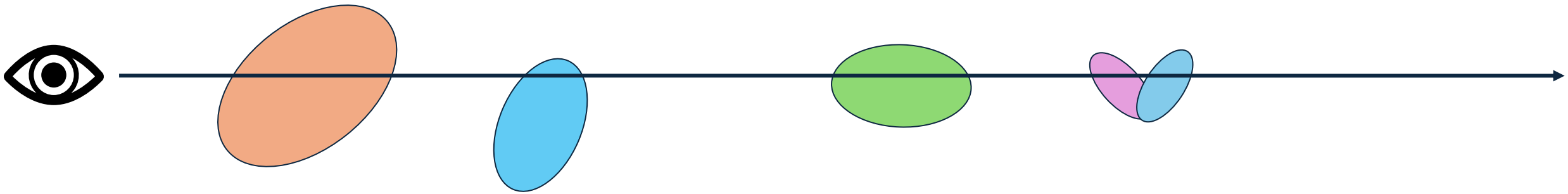


4. Multi-View Loss

# Volume Rendering

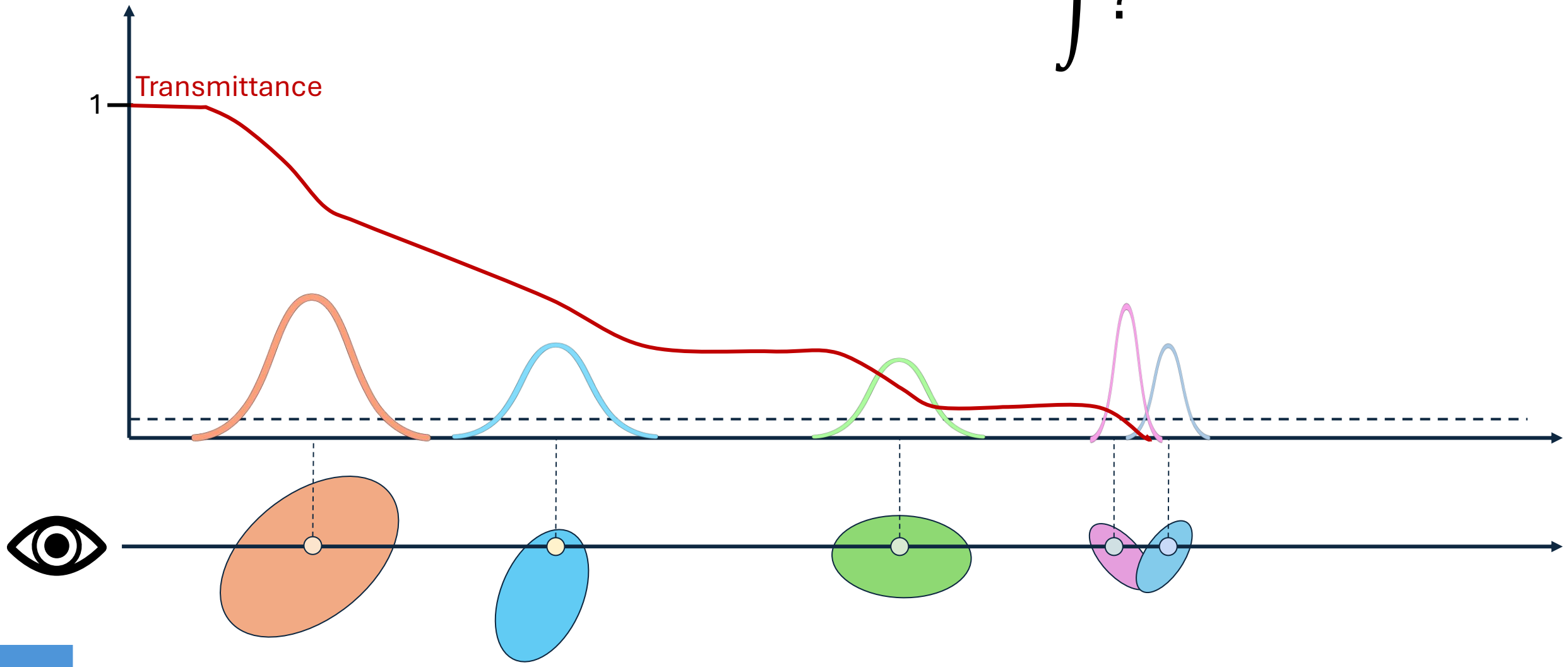


# Volume Rendering



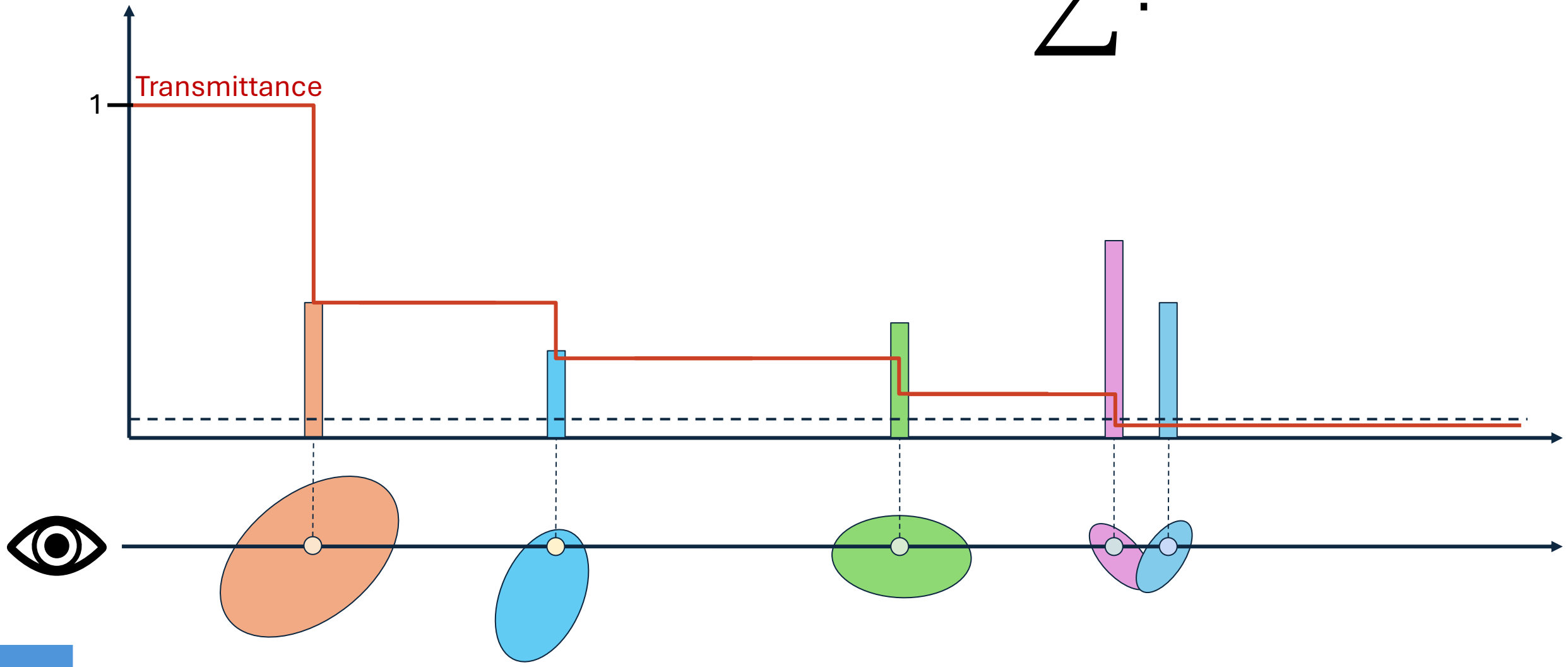
# Volume Rendering

$$\int ?$$

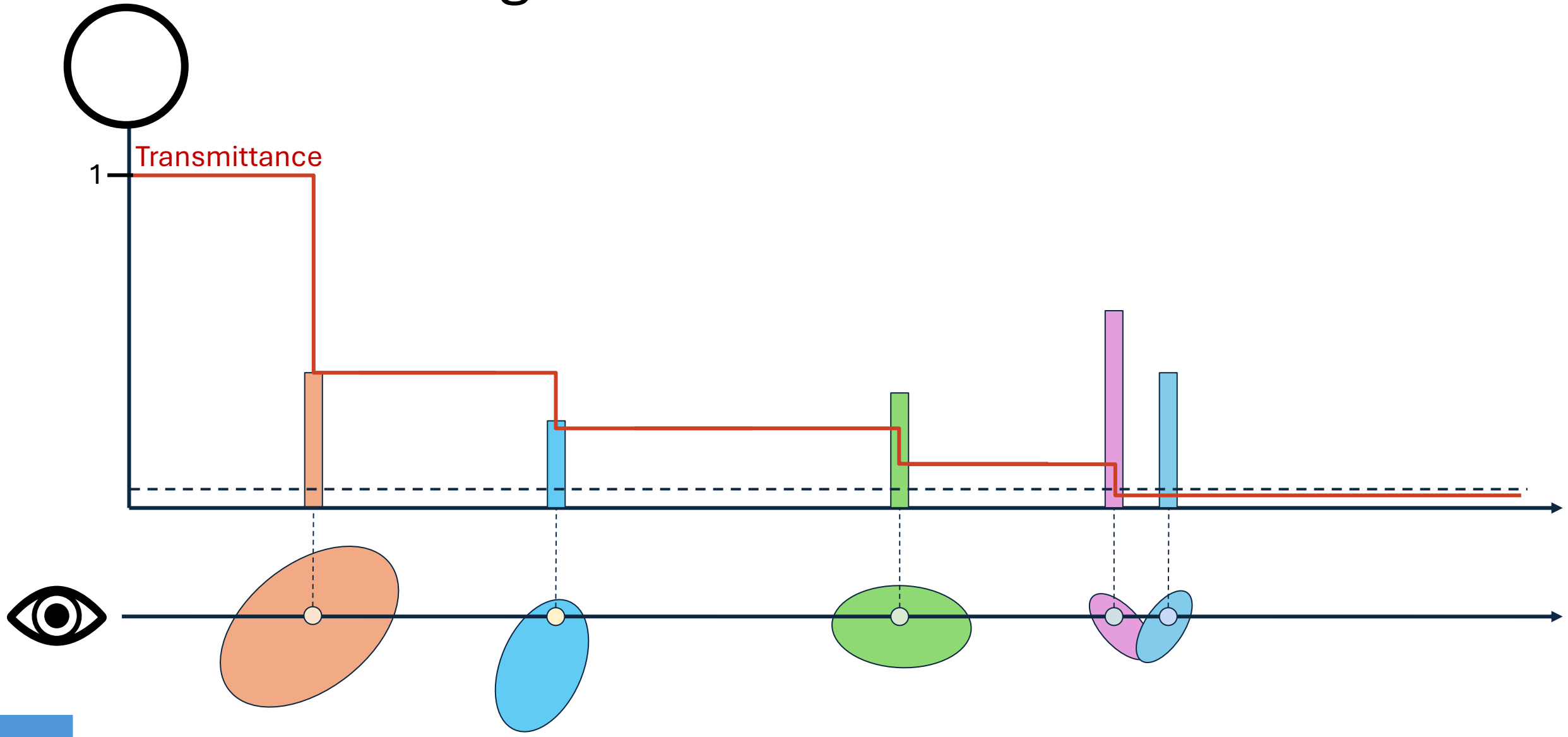


# Volume Rendering

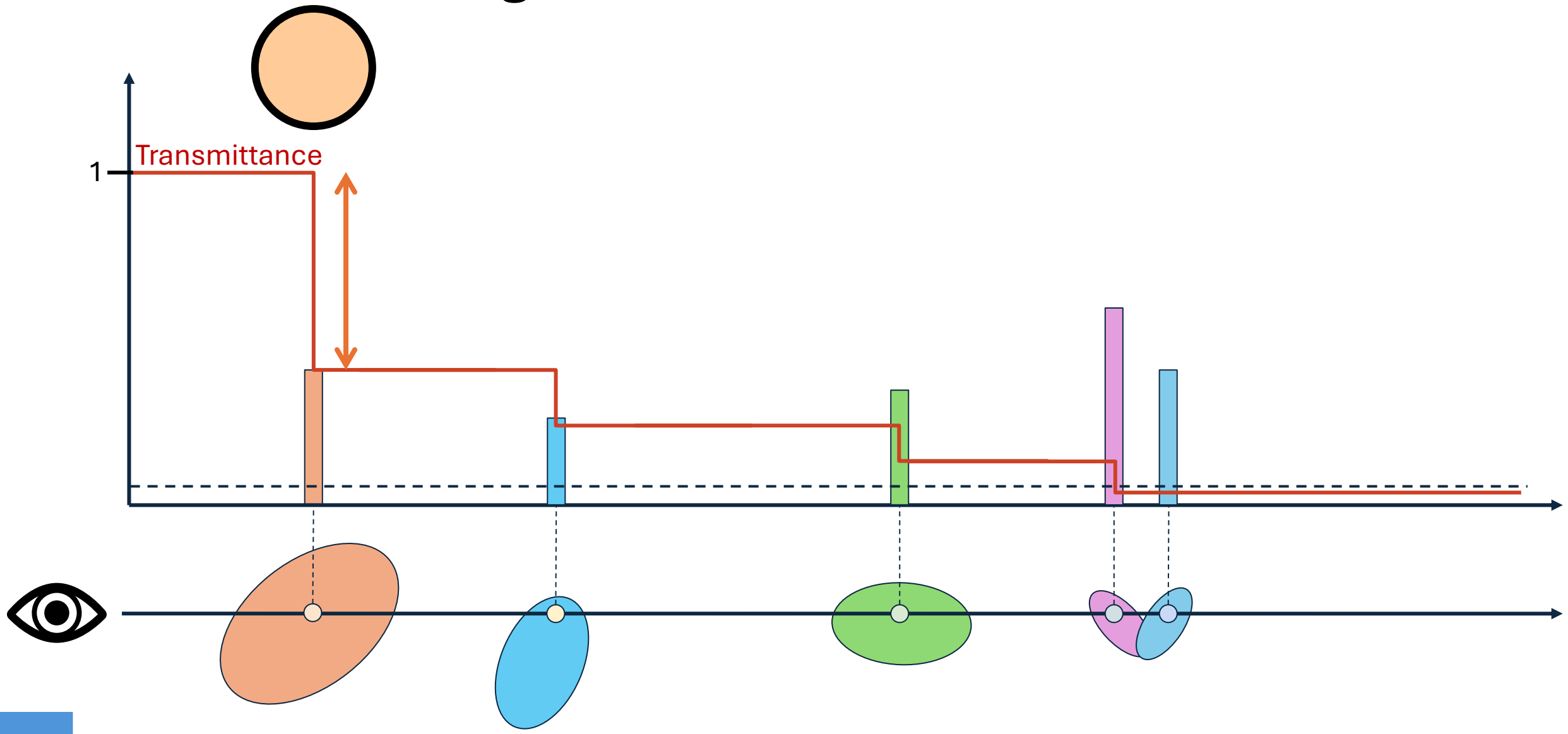
$$\sum !$$



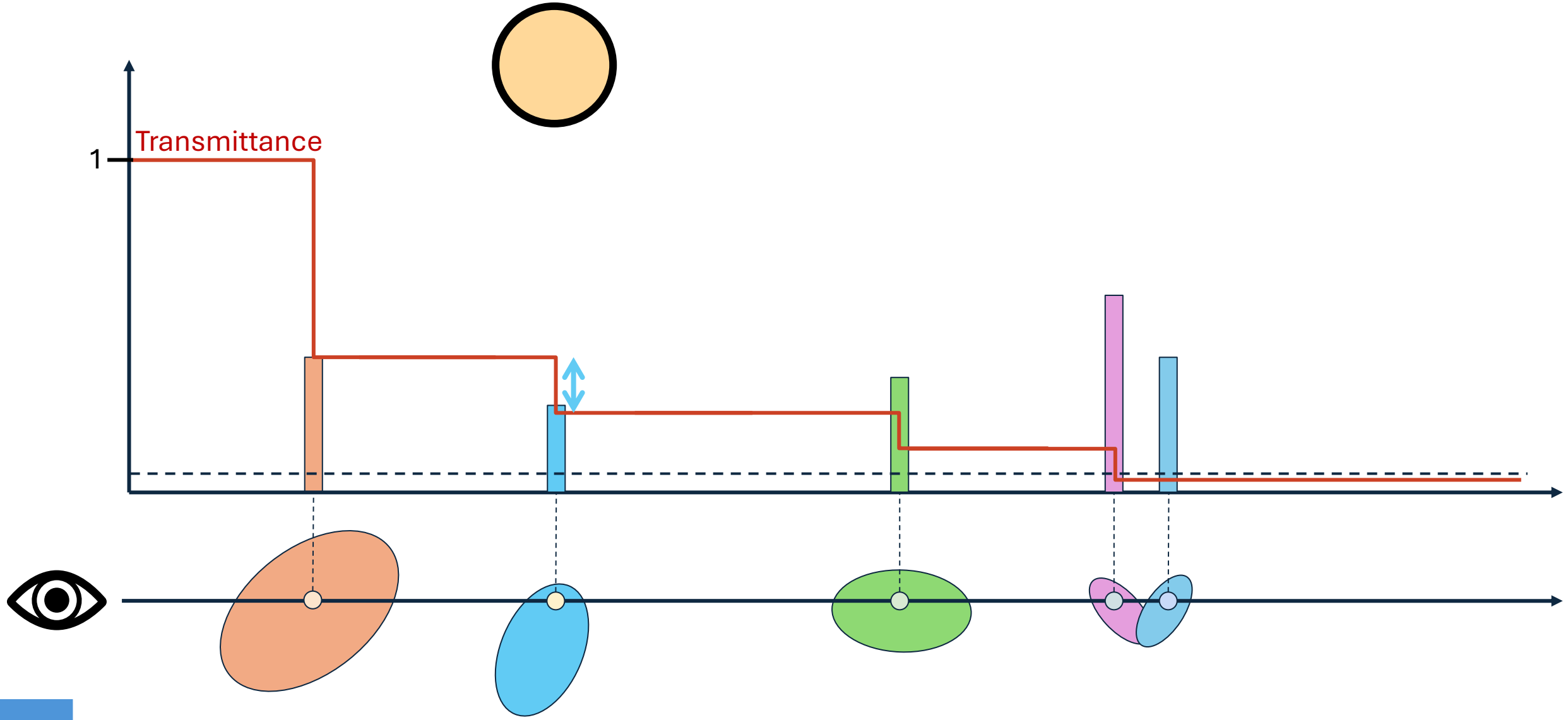
# Volume Rendering



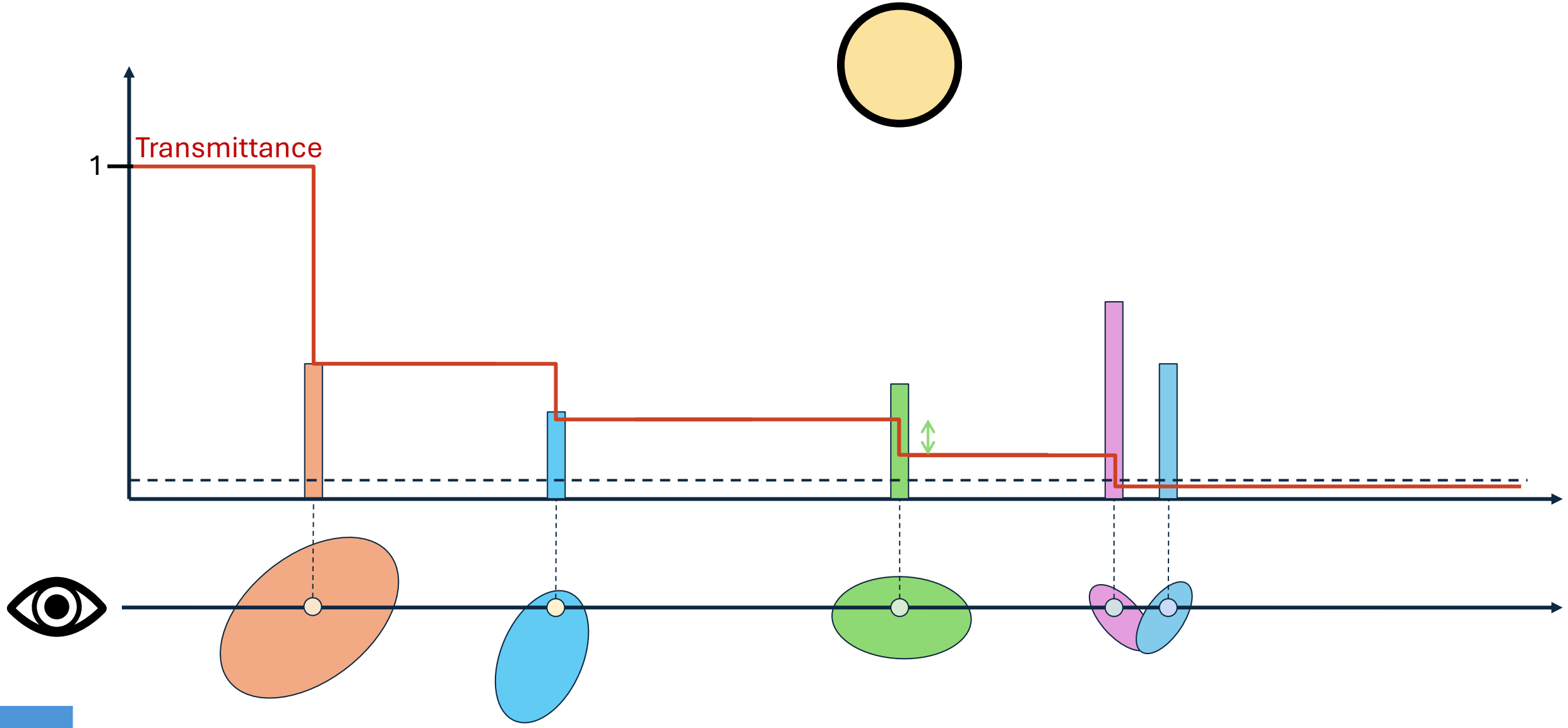
# Volume Rendering



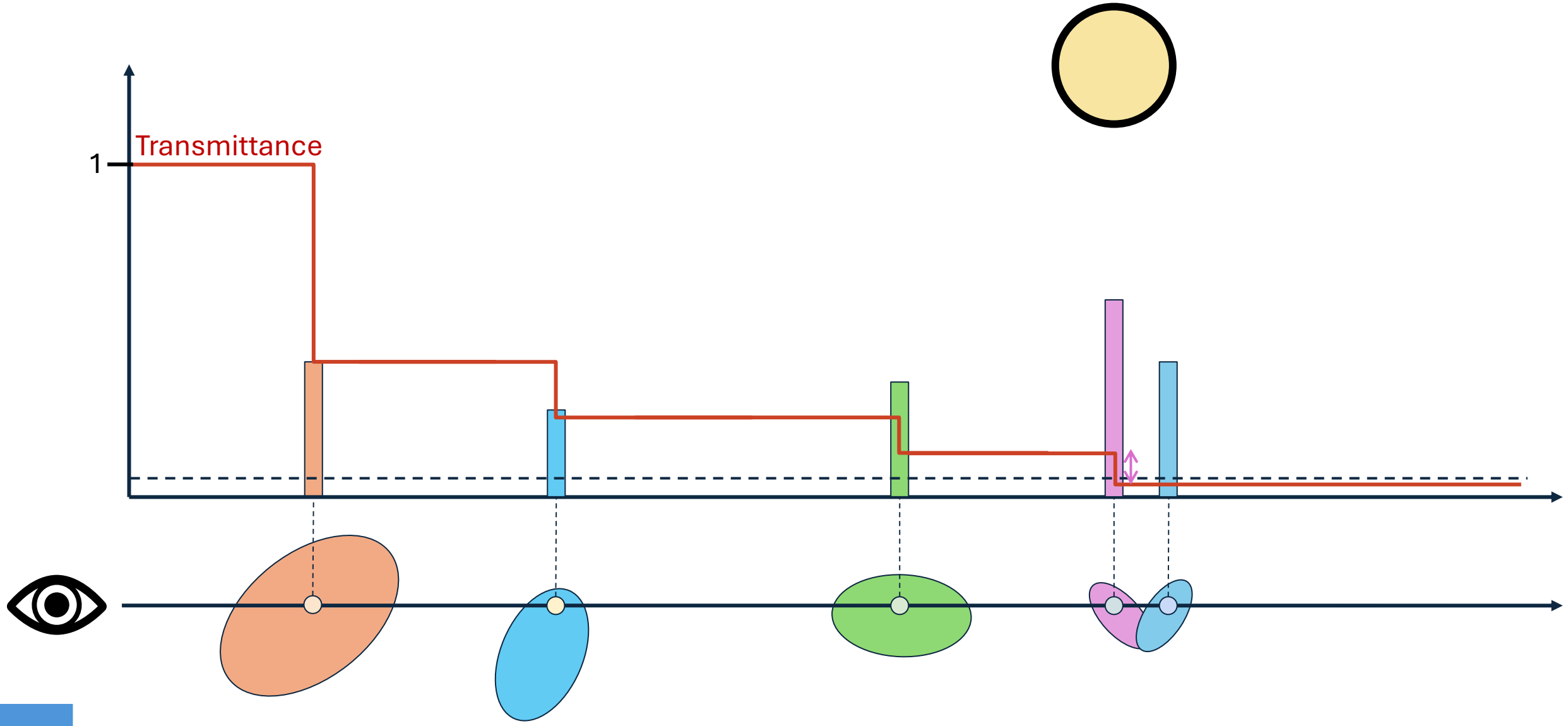
# Volume Rendering



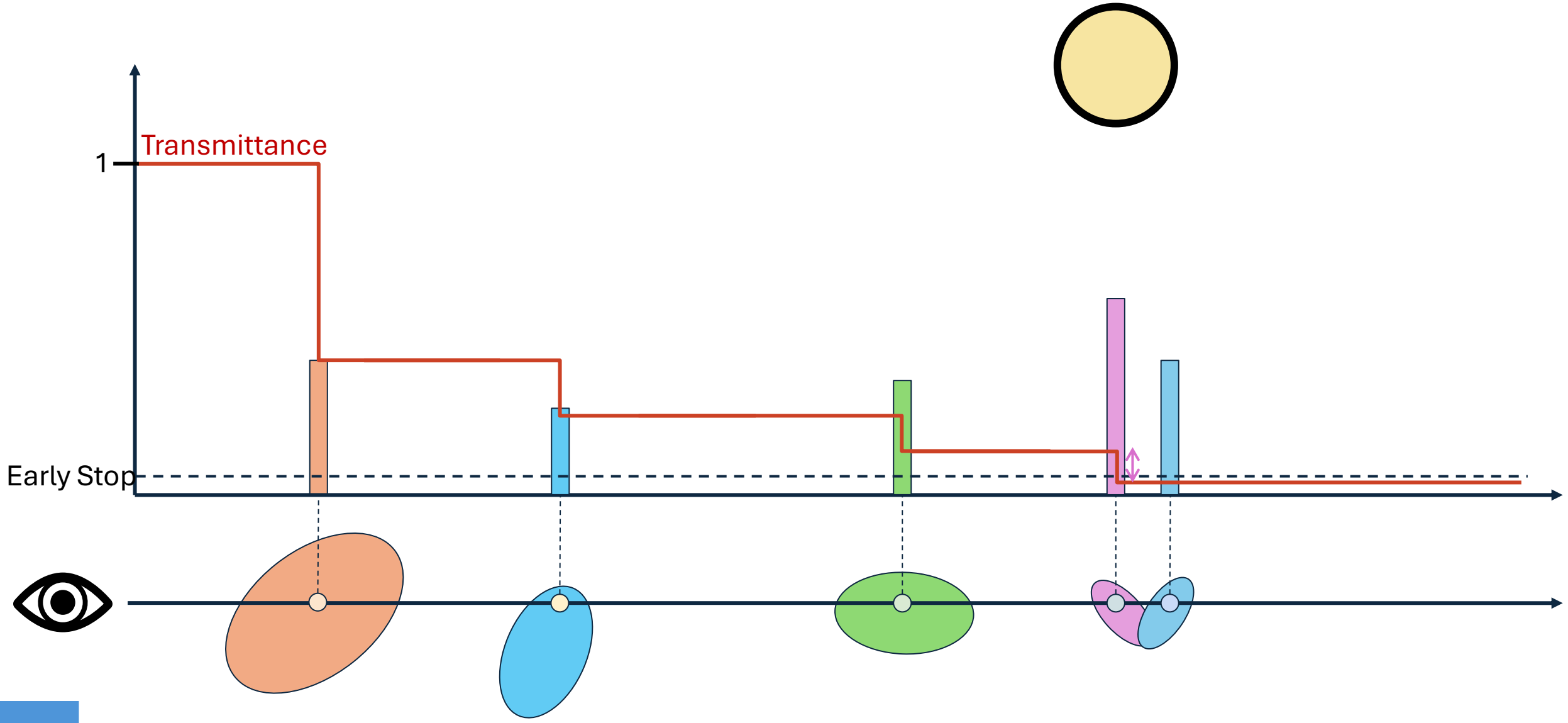
# Volume Rendering



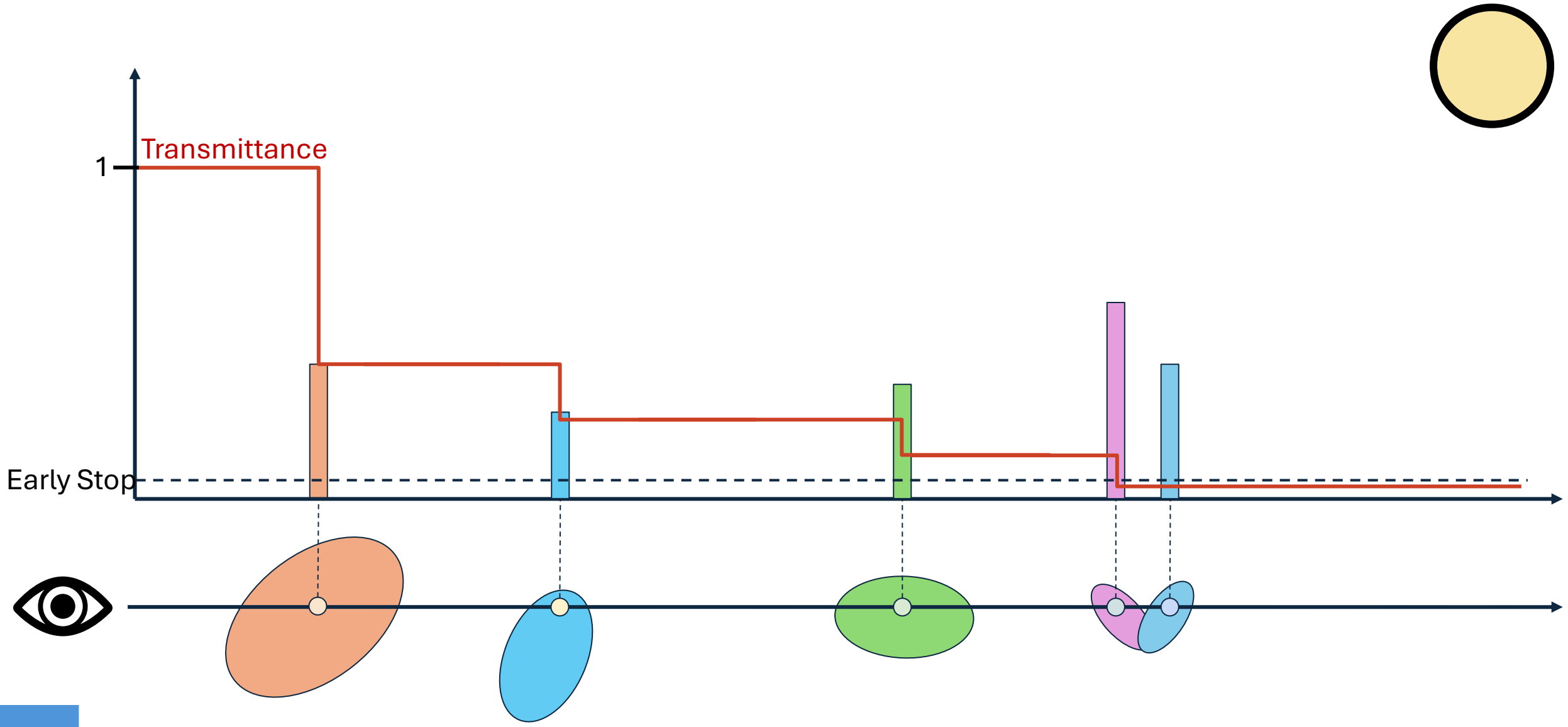
# Volume Rendering



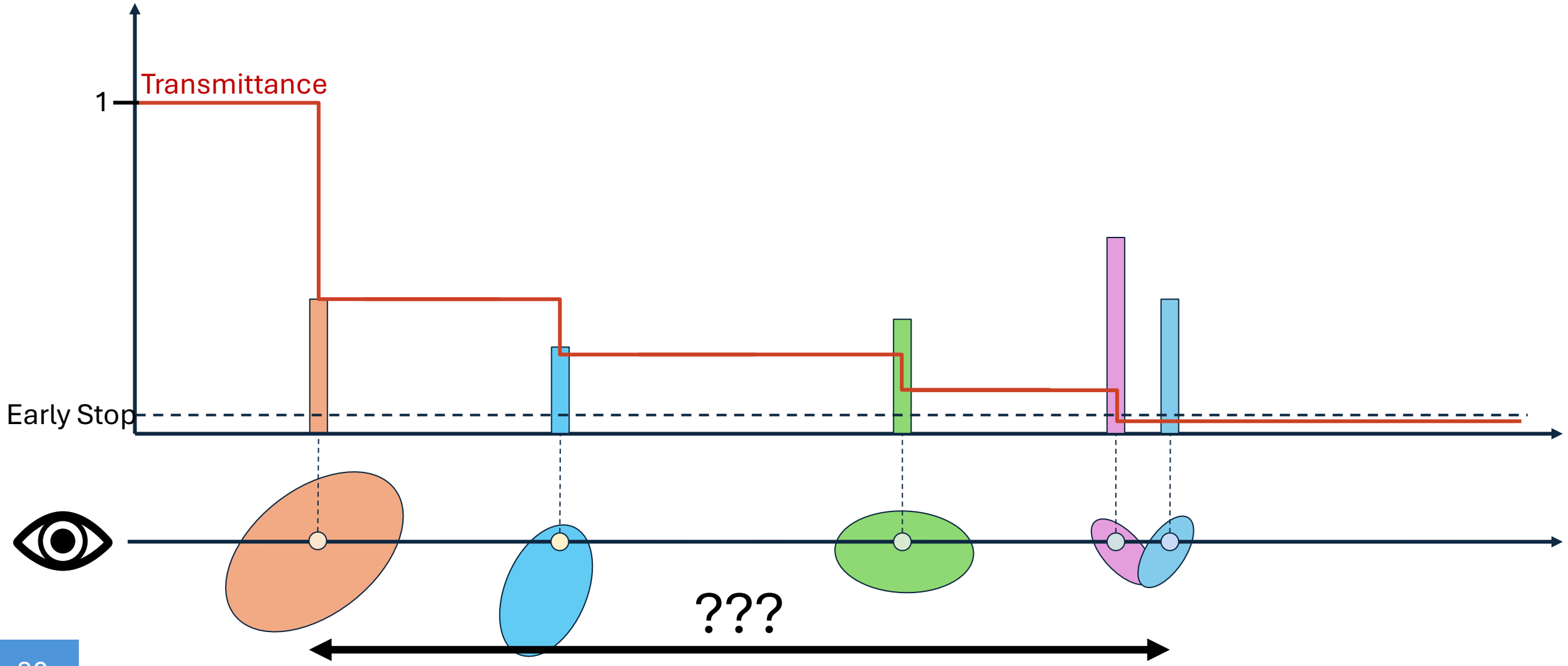
# Volume Rendering



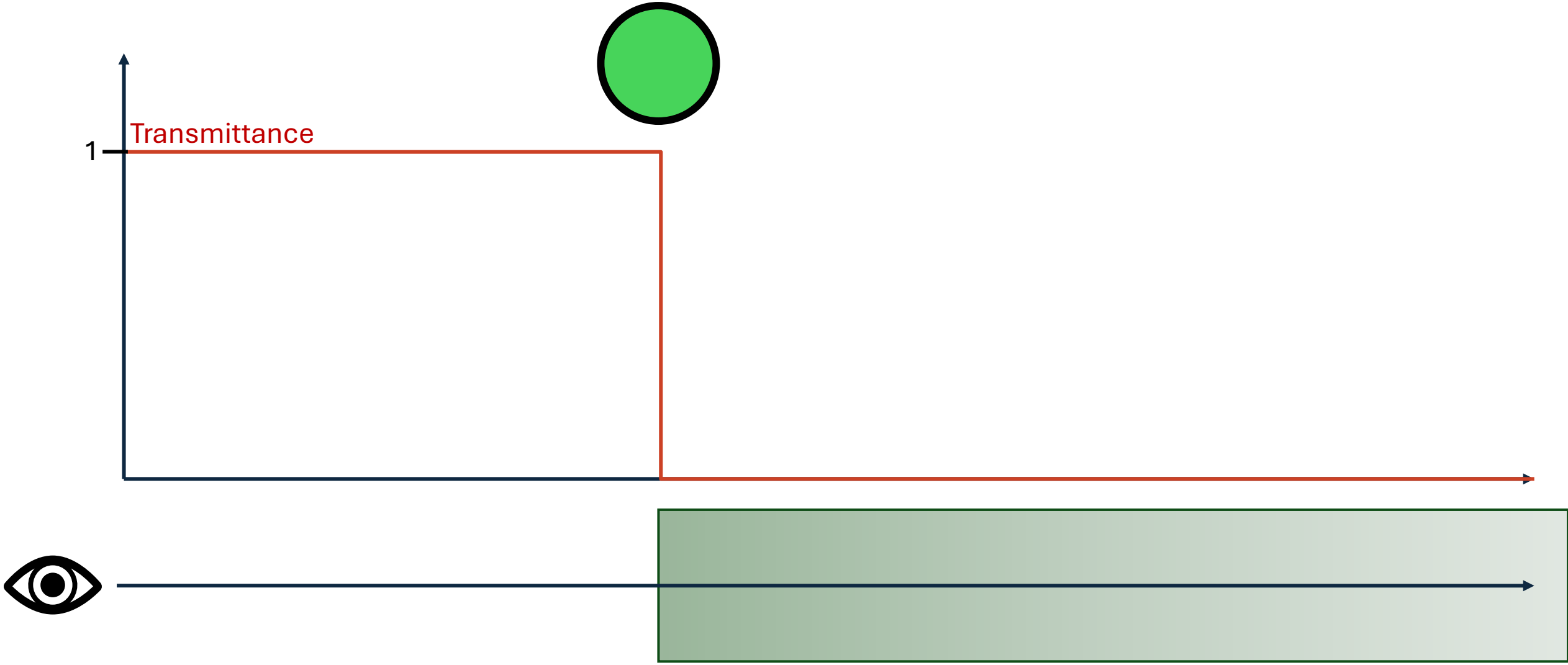
# Volume Rendering



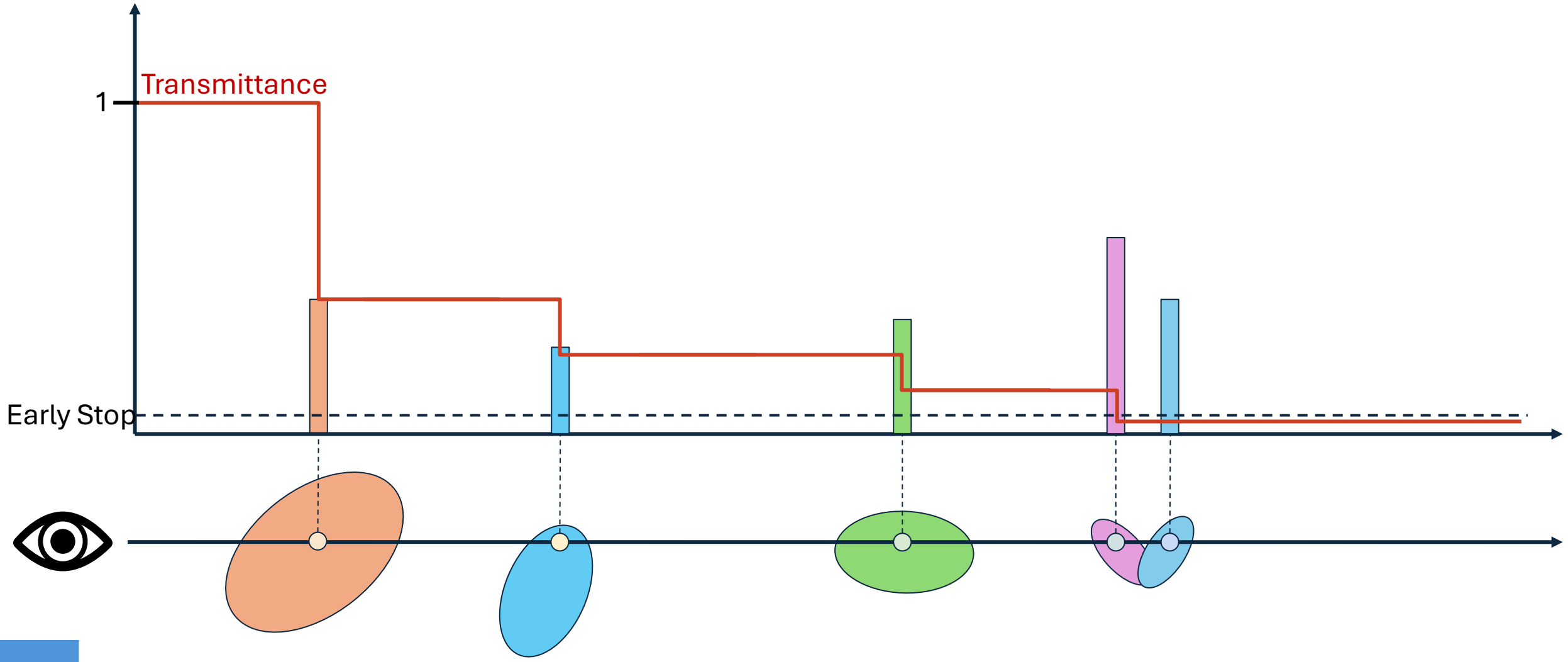
# Volume Rendering



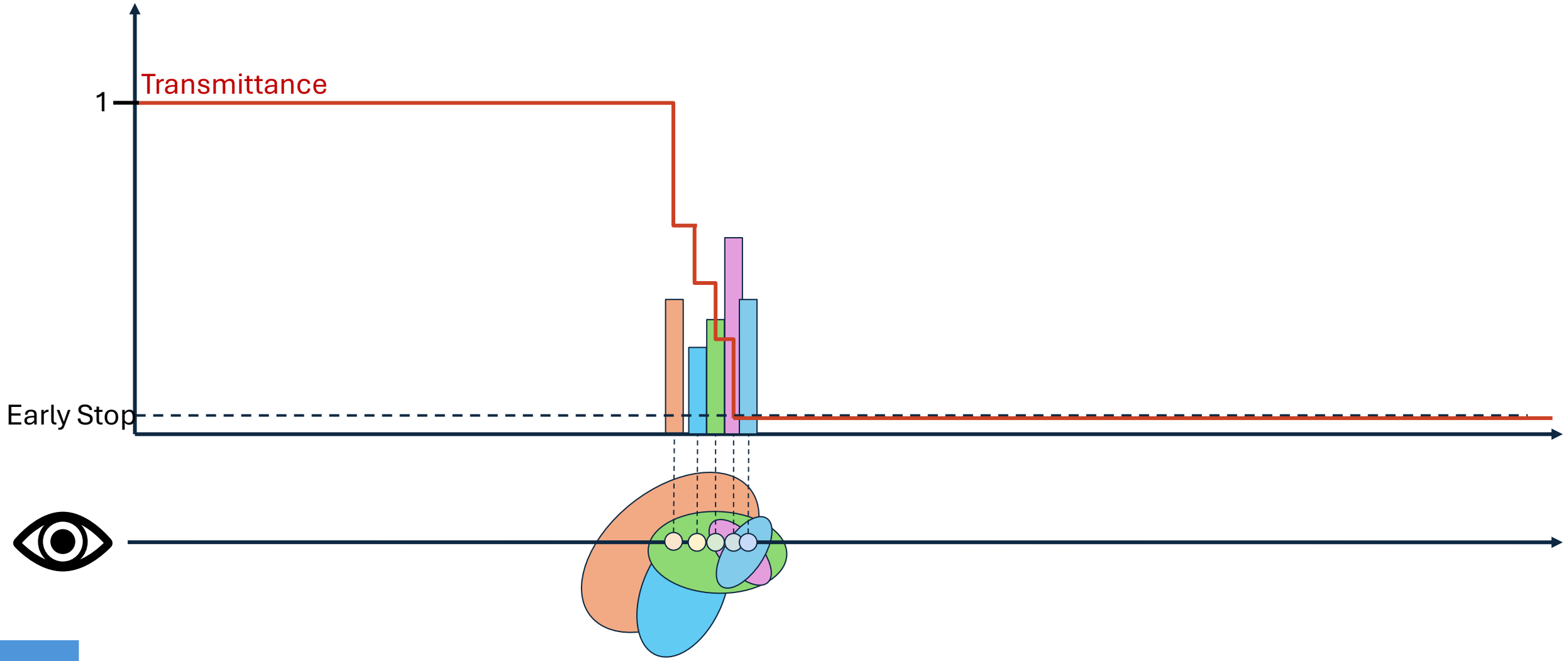
# Surface Rendering



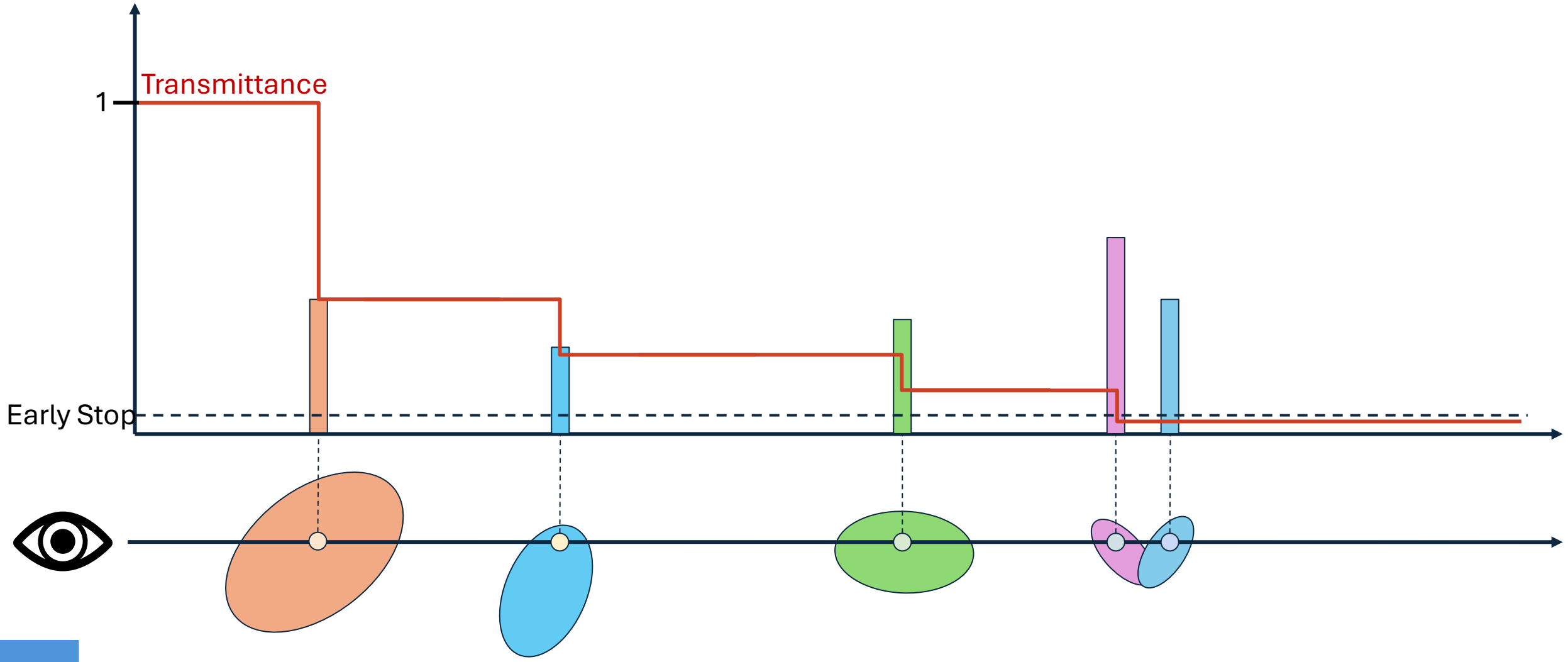
# Volume Rendering



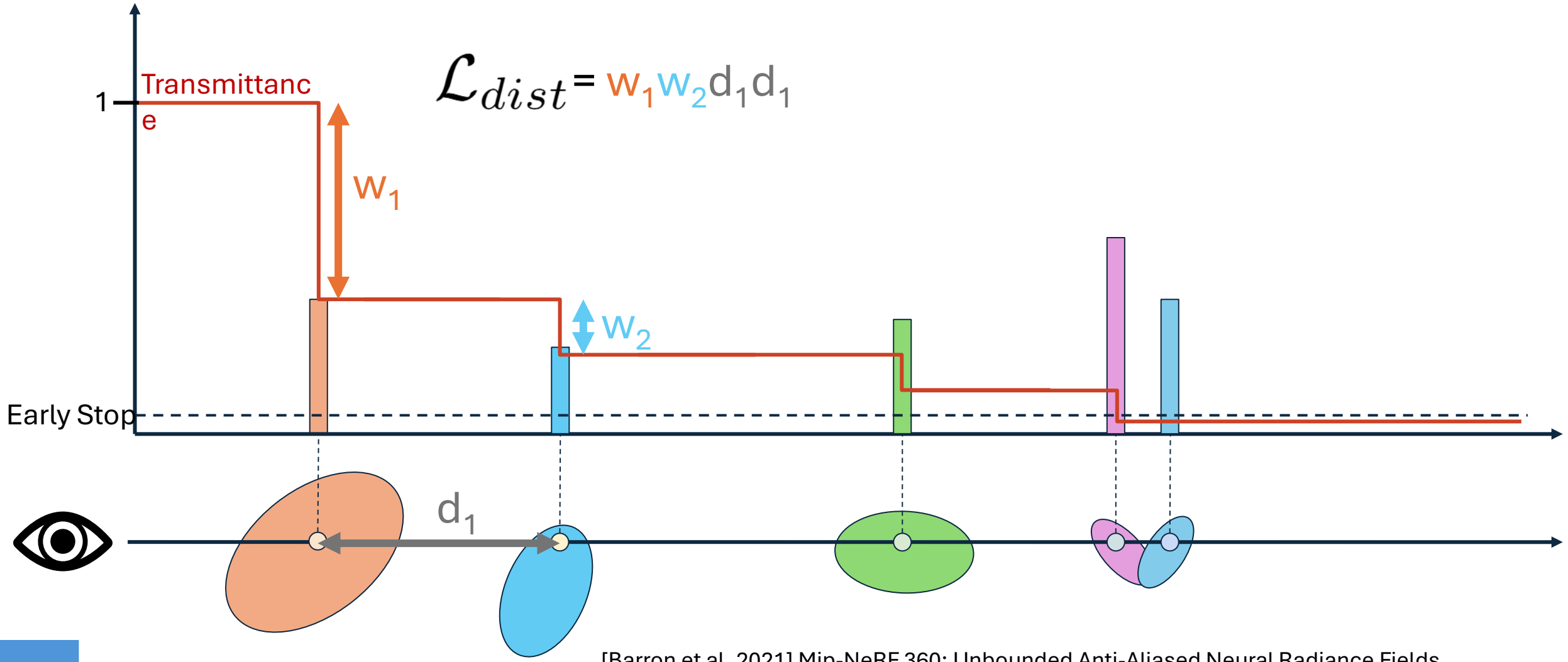
# Volume Rendering



# Volume Rendering

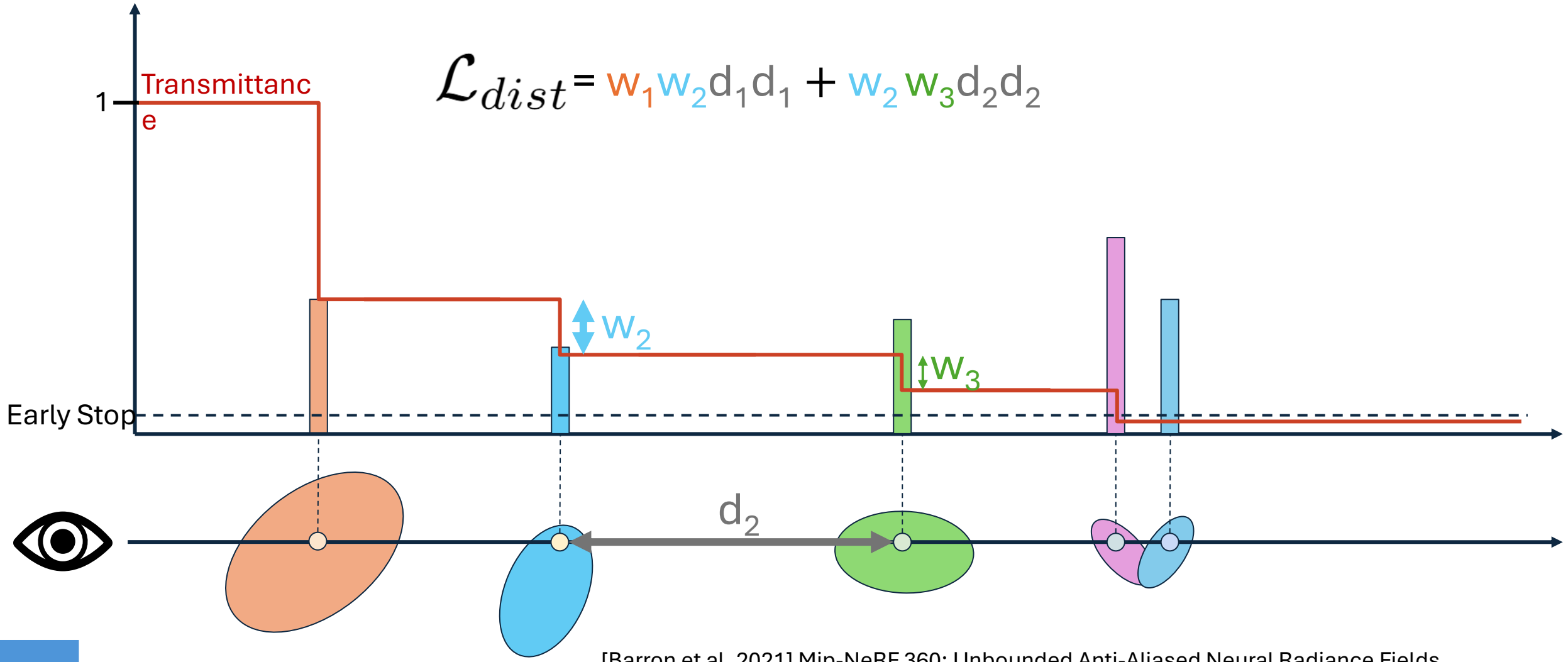


# Volume Rendering




[Barron et al. 2021] Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields
[Huang et al. 2024] 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. SIGGRAPH

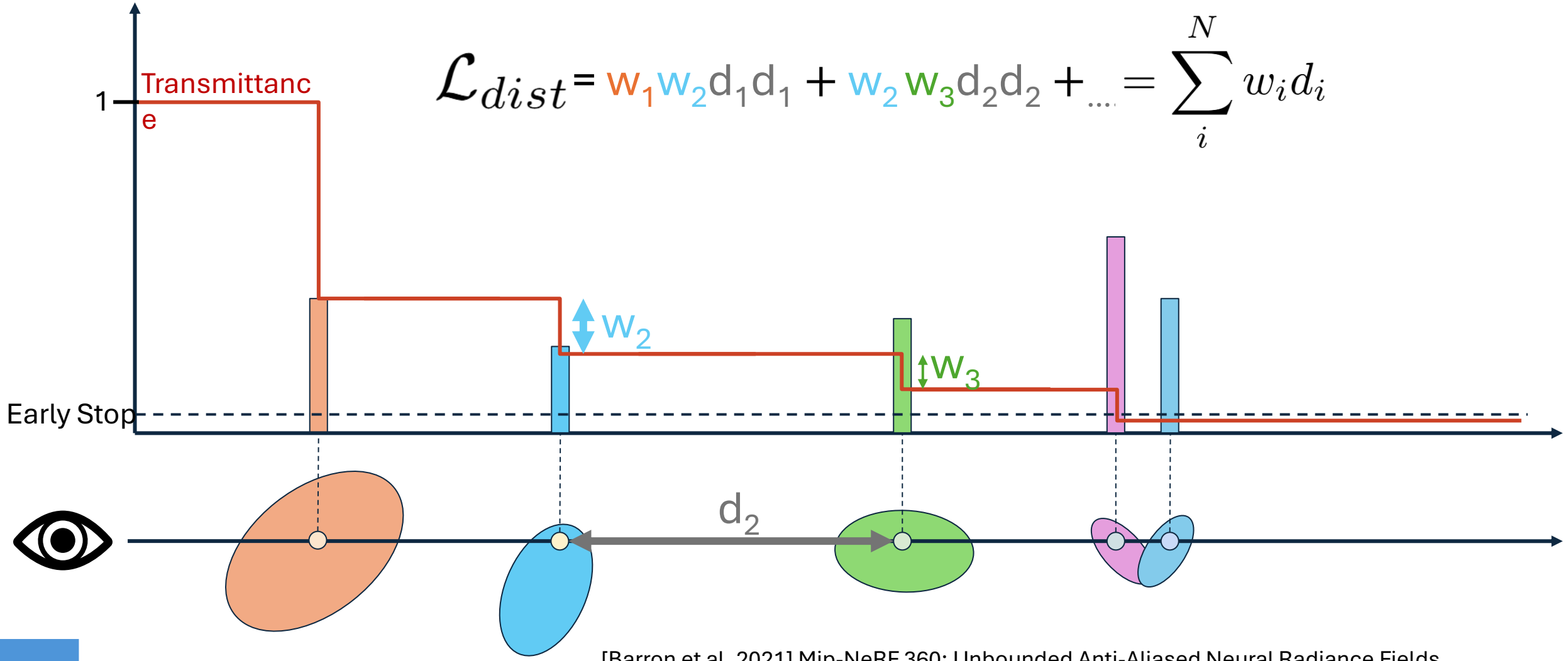
# Volume Rendering




[Barron et al. 2021] Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields
[Huang et al. 2024] 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. SIGGRAPH

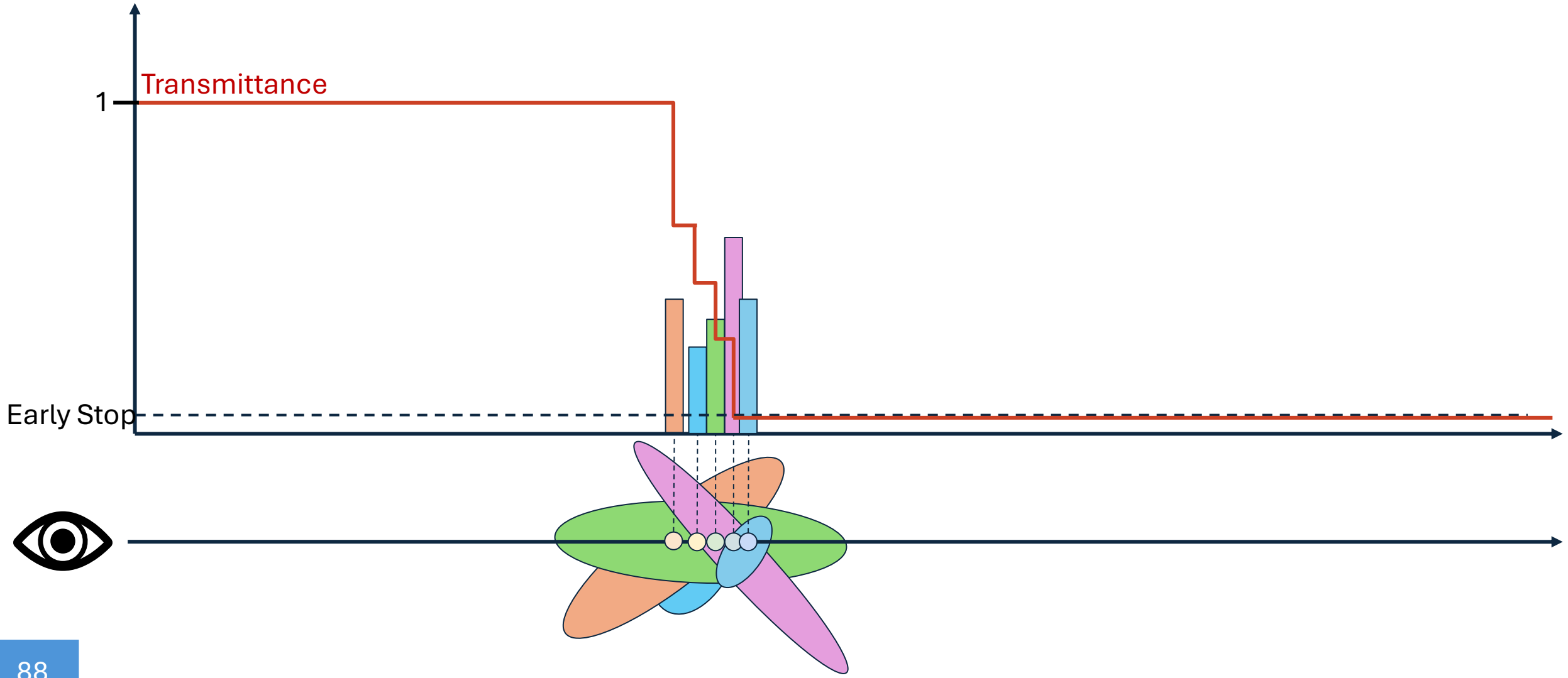
# Volume Rendering

$$\mathcal{L}_{dist} = w_1 w_2 d_1 d_1 + w_2 w_3 d_2 d_2 + \ldots = \sum_i^N w_i d_i$$



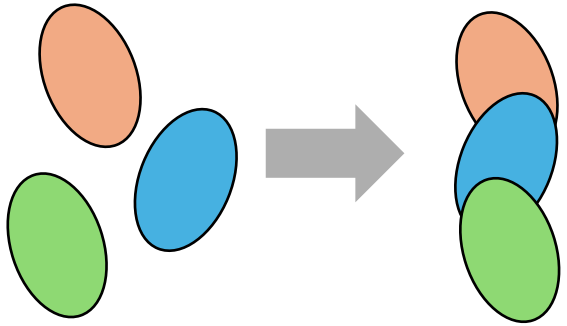

[Barron et al. 2021] Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields
[Huang et al. 2024] 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. SIGGRAPH

# Volume Rendering

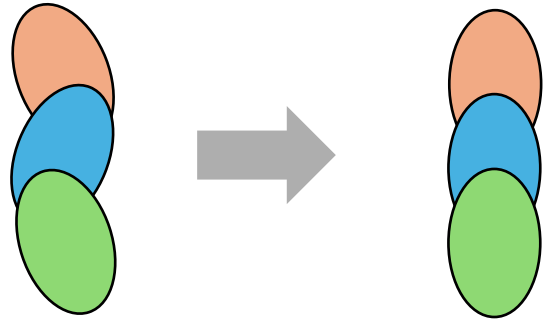


# Geometric Losses

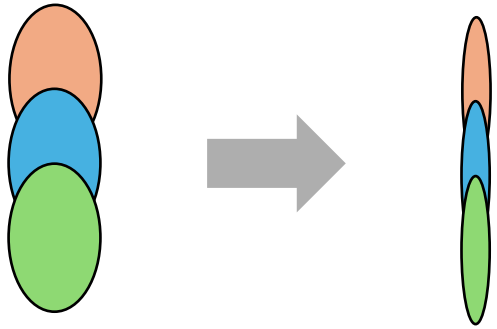
$$\mathcal{L}_{geometry} = \mathcal{L}_{dist} + \mathcal{L}_{normal} + \mathcal{L}_{flat} + \mathcal{L}_{multiview}$$



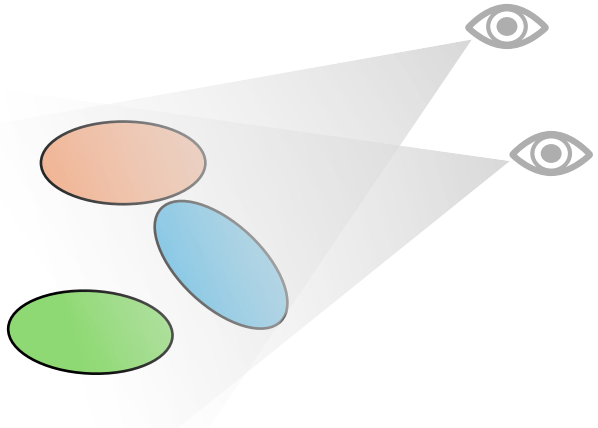
1. Depth Distortion Loss



**2. Normal Consistency Loss**

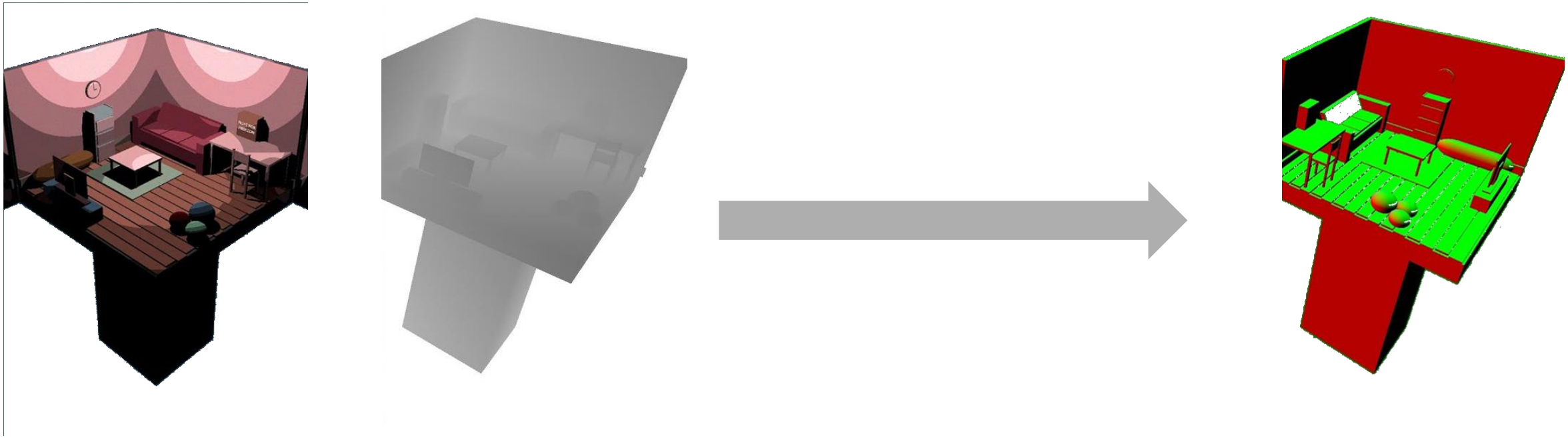


3. Flattening Gaussians



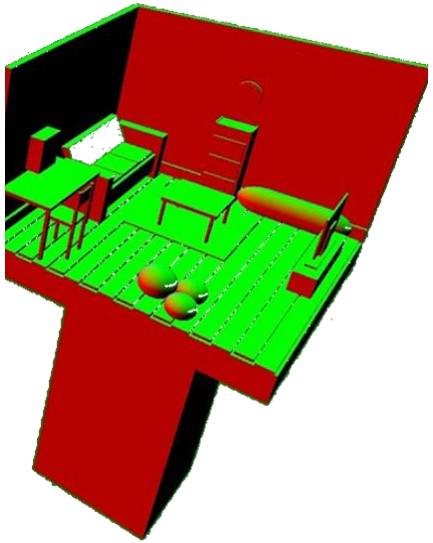
4. Multi-View Loss

# Normal Consistency



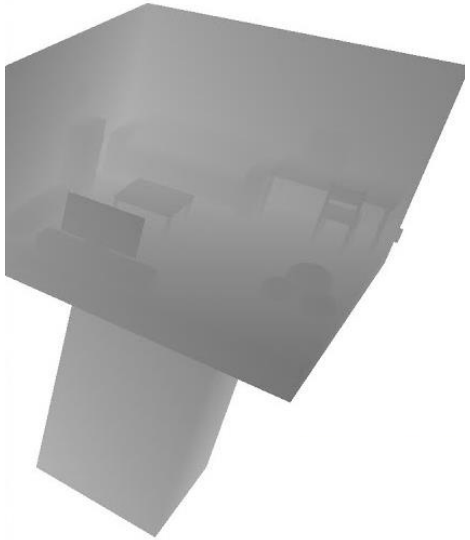
[FM Coding] How to Render Depth & World Space Normals on Game Screen? (Unity3D Beginner Tutorial)

# Normal Consistency



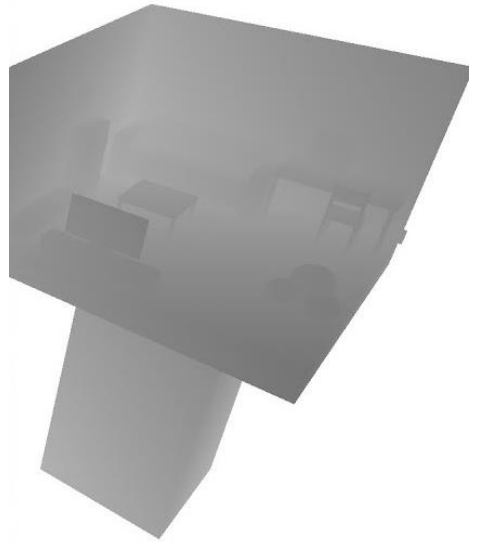
$=$

$\nabla y$



$\times$

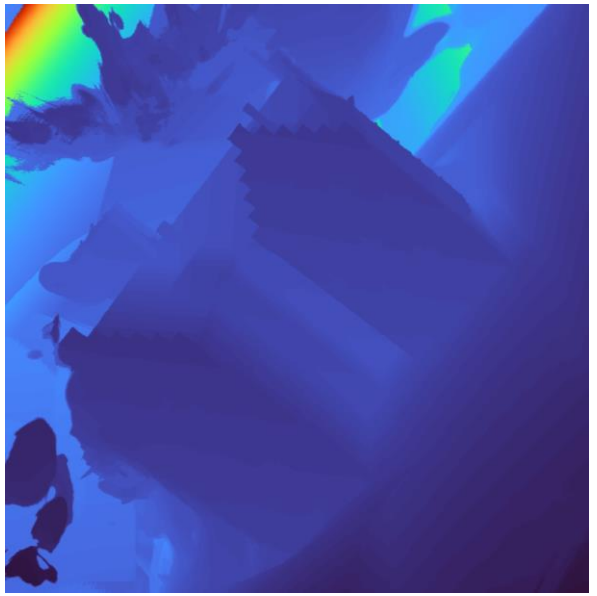
$\nabla x$



[FM Coding] How to Render Depth & World Space Normals on Game Screen? (Unity3D Beginner Tutorial)

# 3D Gaussian Normals / Depths

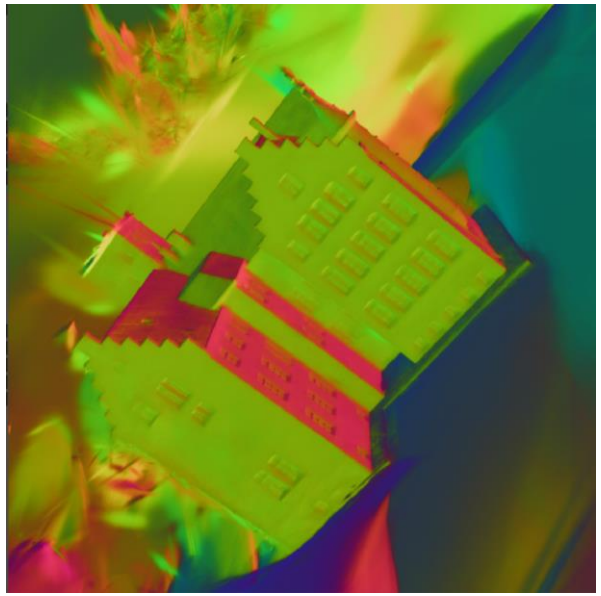
Render



Depth

› Mean?

› Median?

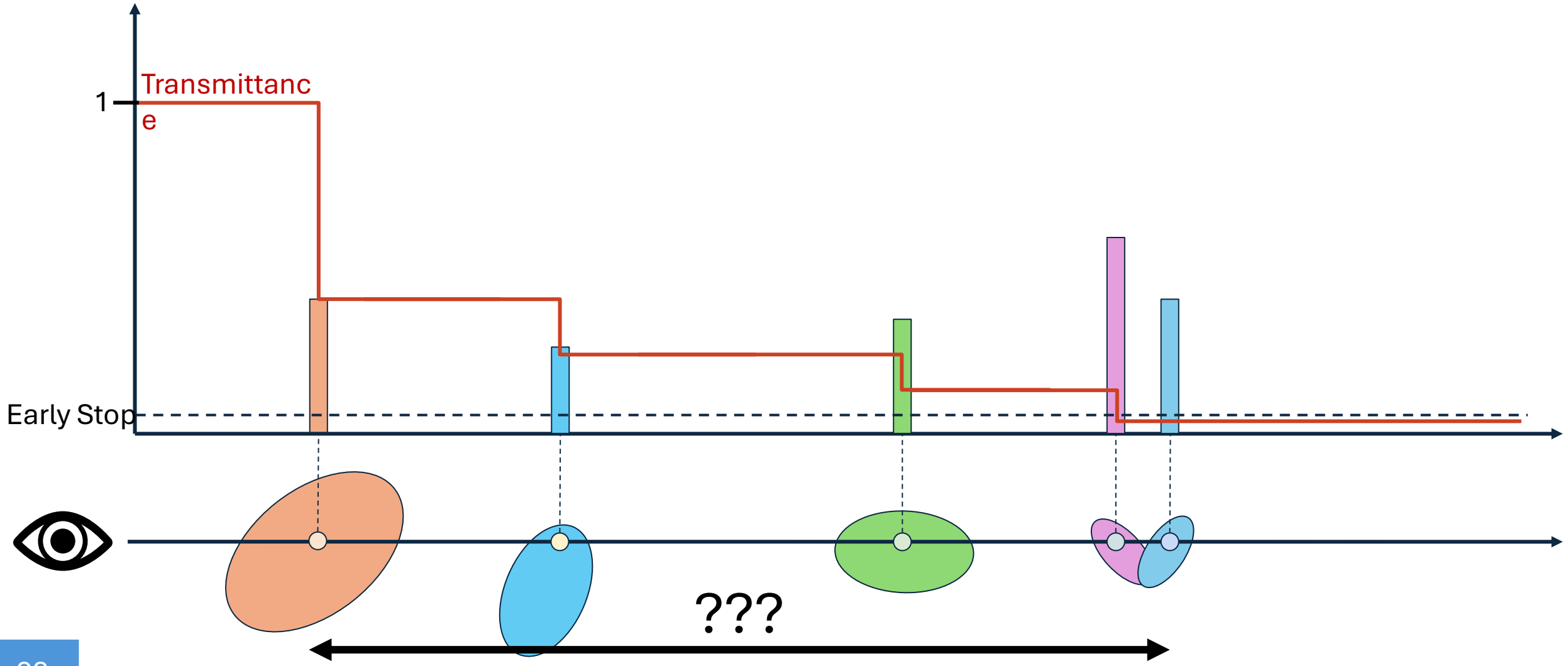


Normal

› Min Scale?

› Intersection Plane?

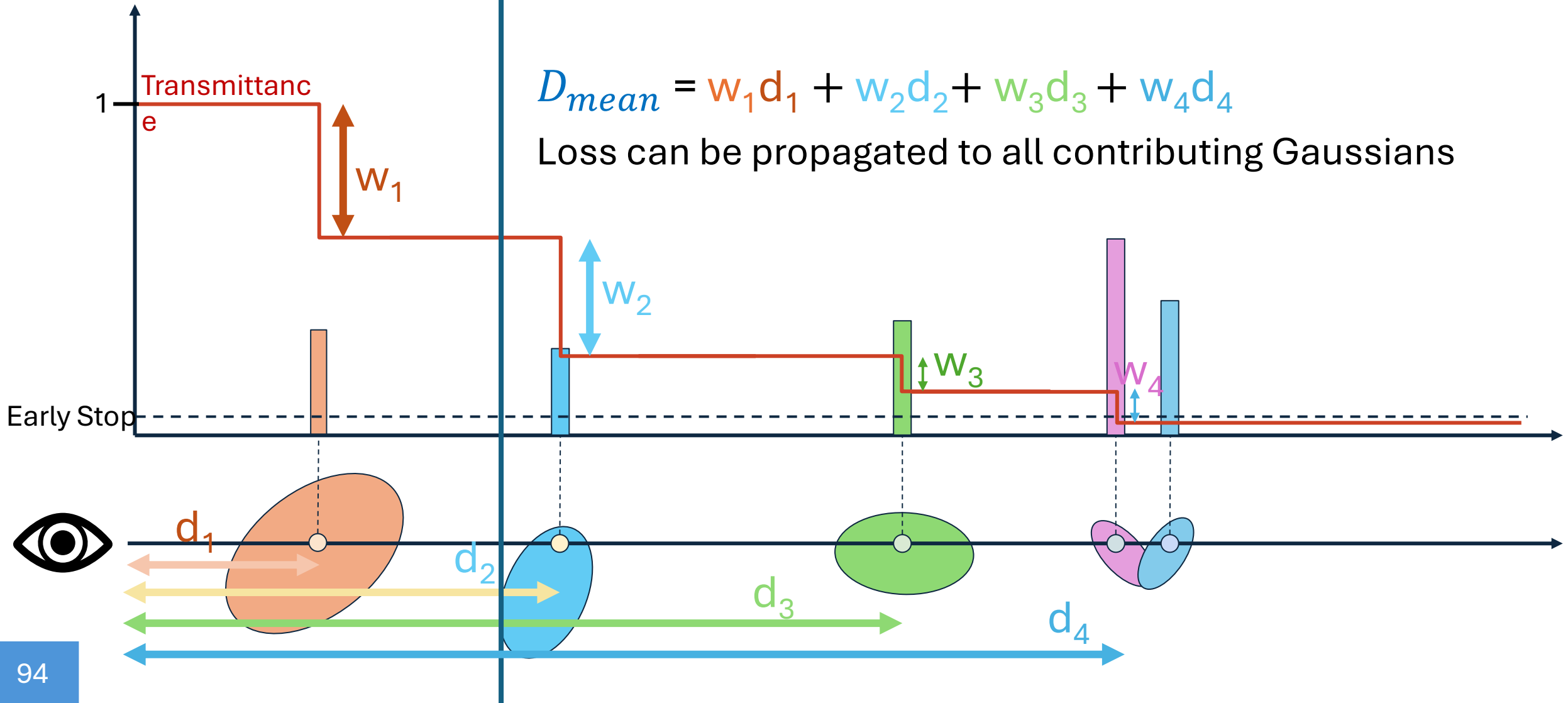
# Gaussian Depth



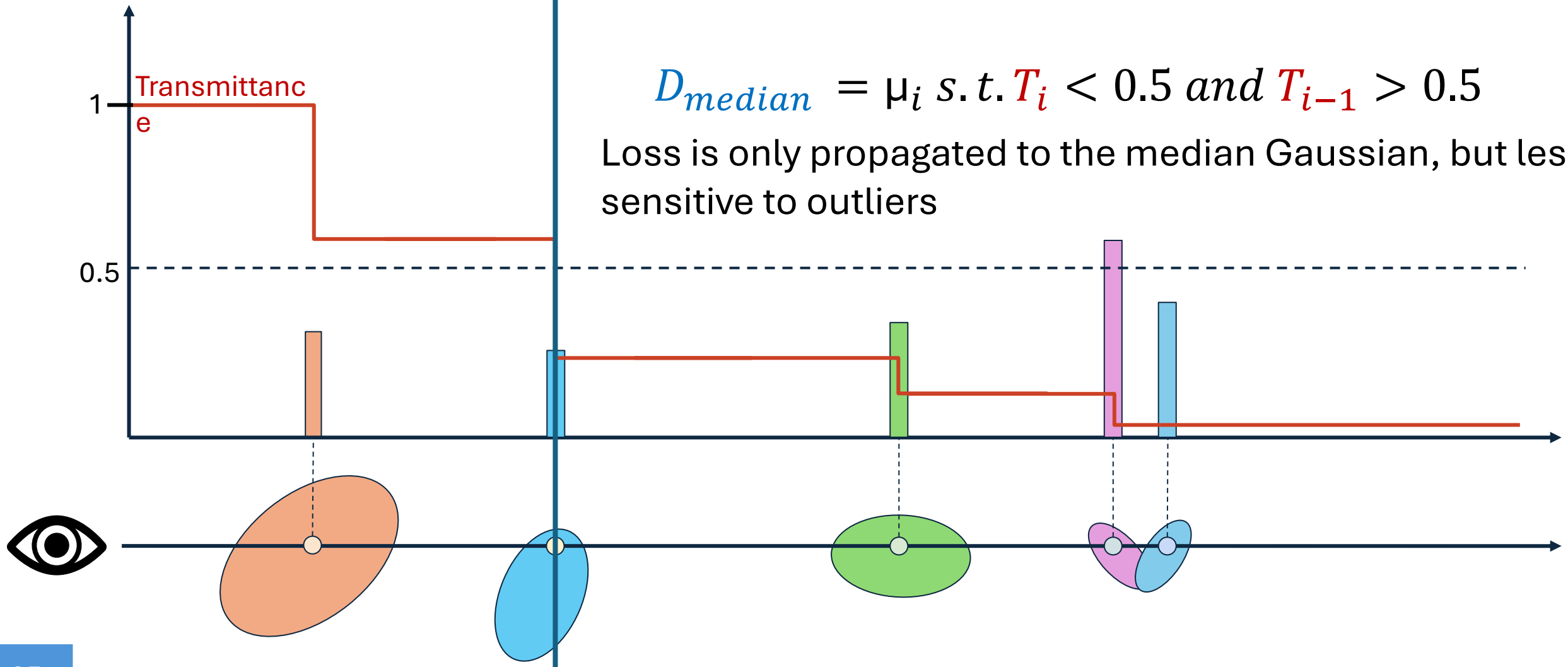
# Mean Depth

$D_{mean} = w_1 d_1 + w_2 d_2 + w_3 d_3 + w_4 d_4$

Loss can be propagated to all contributing Gaussians



# Median Depth

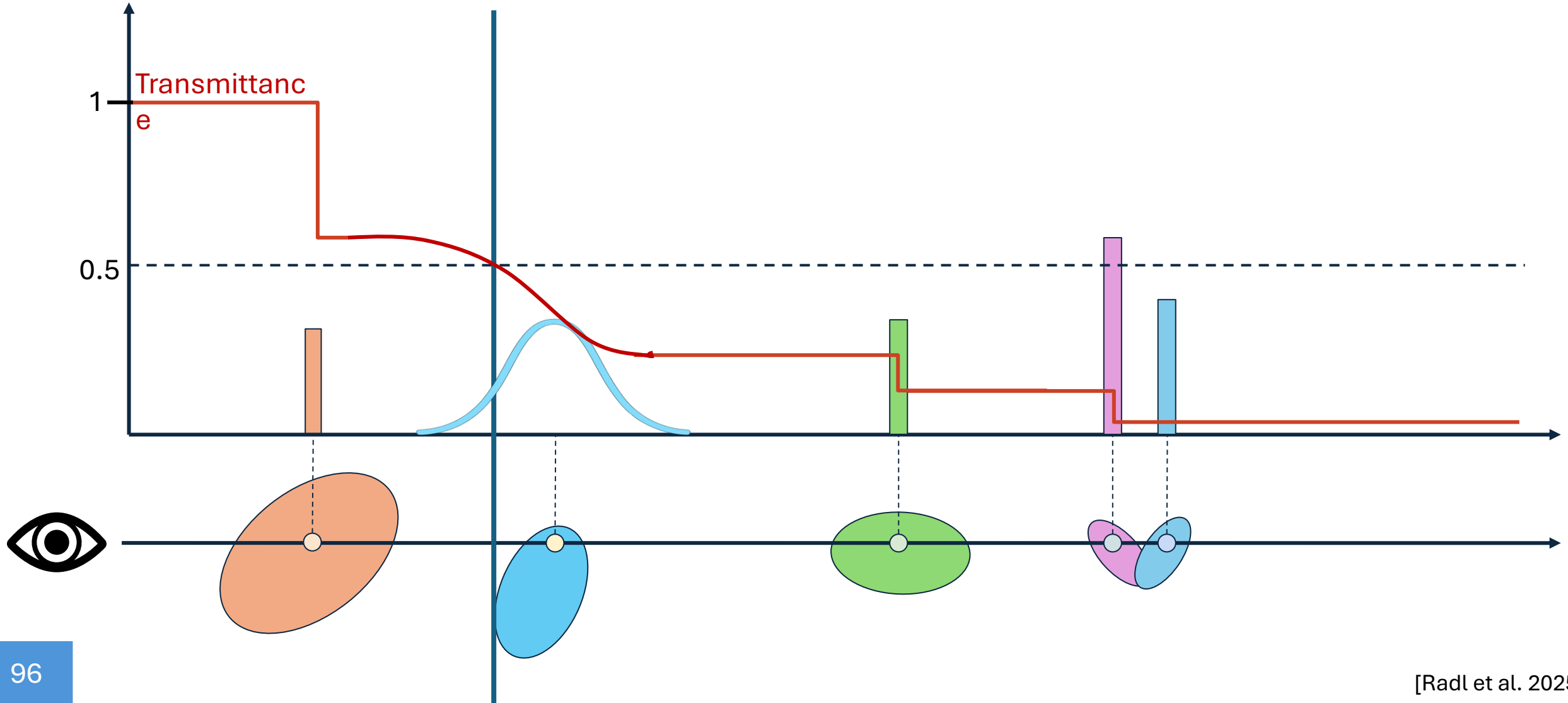



$$D_{median} = \mu_i \; s.t. \; T_i < 0.5 \; and \; T_{i-1} > 0.5$$

Loss is only propagated to the median Gaussian, but less sensitive to outliers

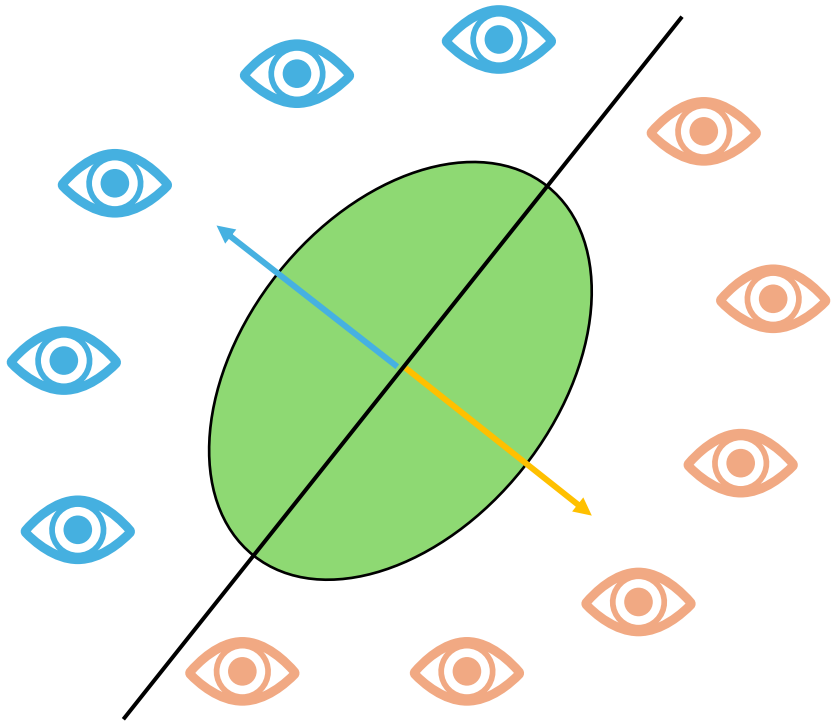
[Yu et al. 2024] Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded

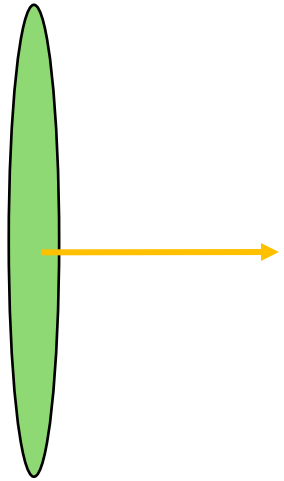
# Exact Median Depth




[Radl et al. 2025]

# Min Scale Normal



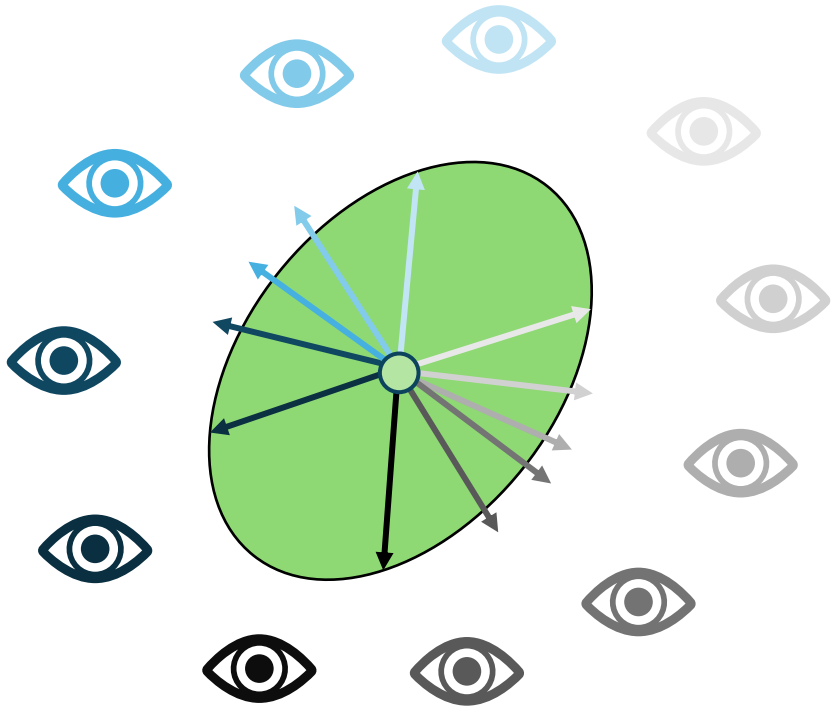


Normals point in the flattest direction



[Guedon and Lepetit 2023] SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rende

# Intersection Plane Normal



Normals should point outwards

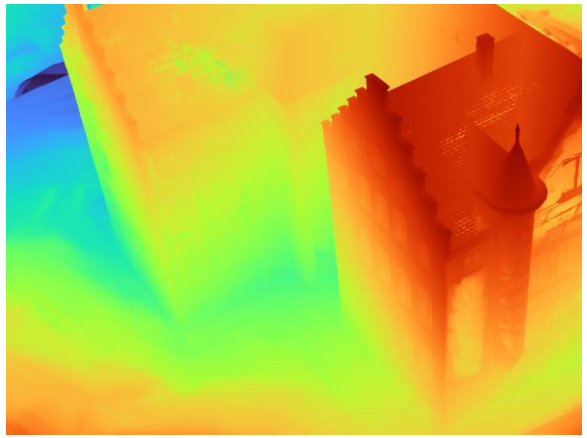
[Yu et al. 2024]

# Normal Consistency Loss $\mathcal{L}_{normal}$

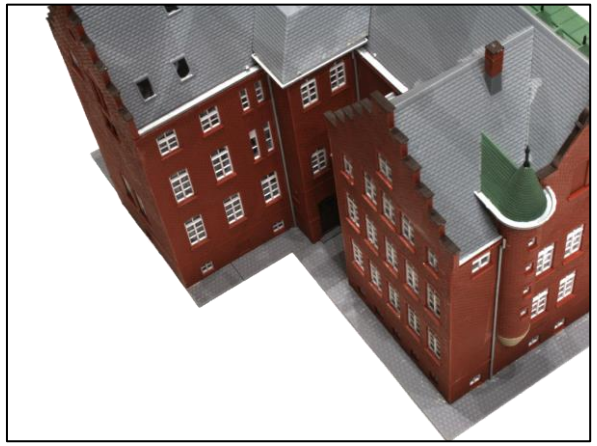



[Huang et al. 2024] 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. SIGGRAPH

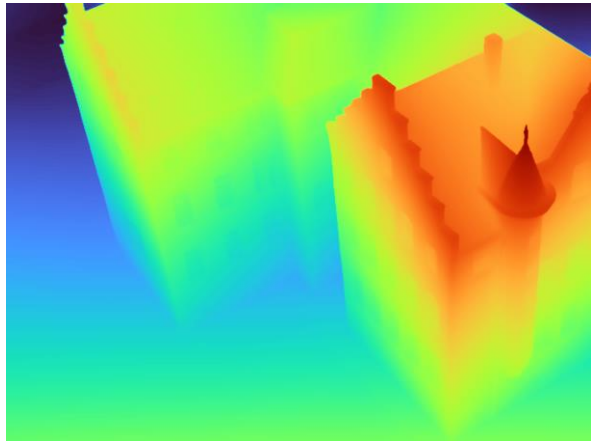
# Monocular Supervision



Depth from Gaussian Model

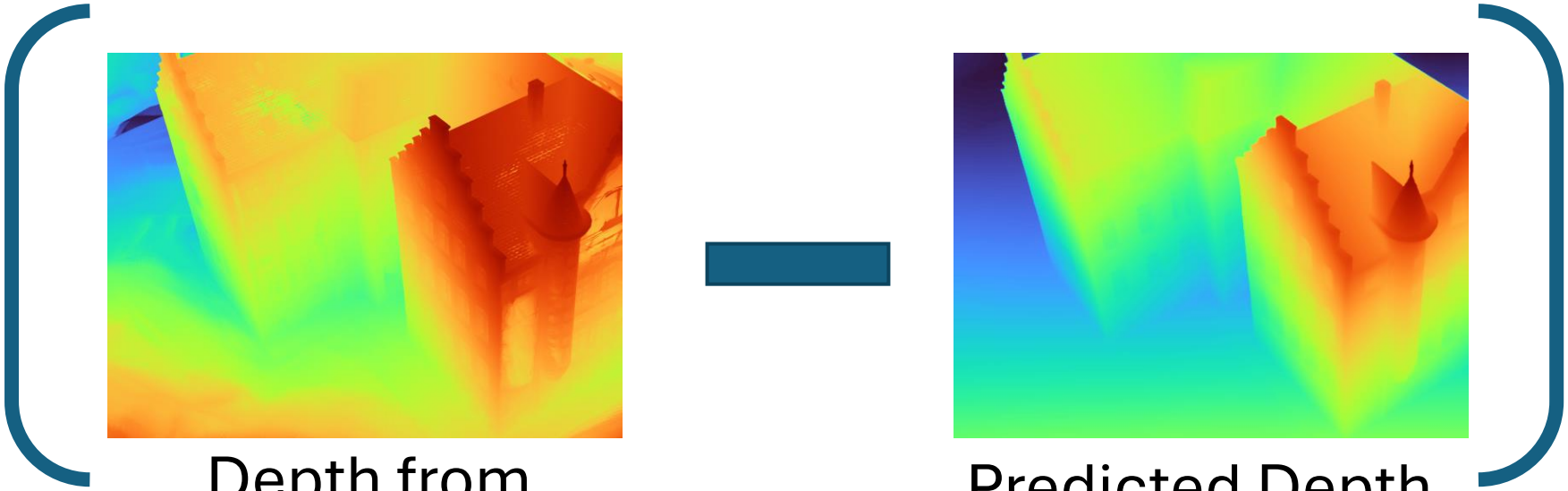


Input Image



Predicted Depth from DPT 2

# Monocular Supervision Loss

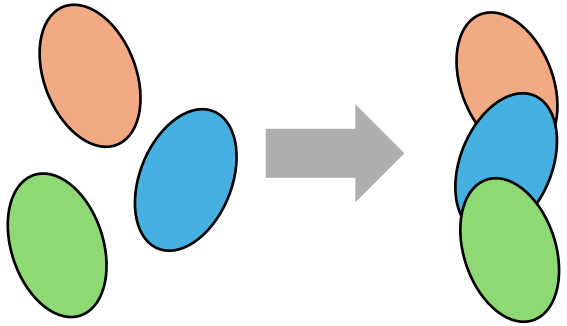


Depth from Gaussian Model

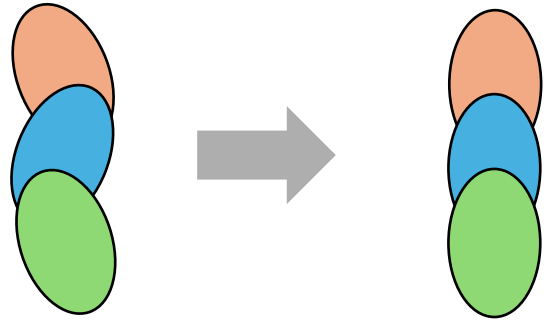
Predicted Depth from DPT 2

# Geometric Losses

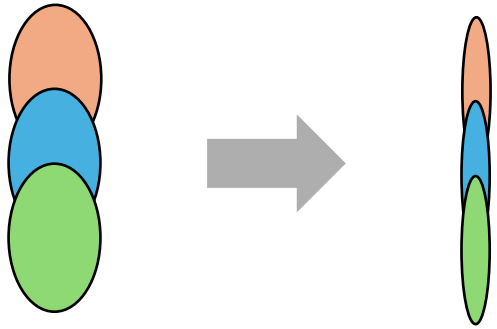
$$\mathcal{L}_{geometry} = \mathcal{L}_{dist} + \mathcal{L}_{normal} + \mathcal{L}_{flat} + \mathcal{L}_{multiview}$$



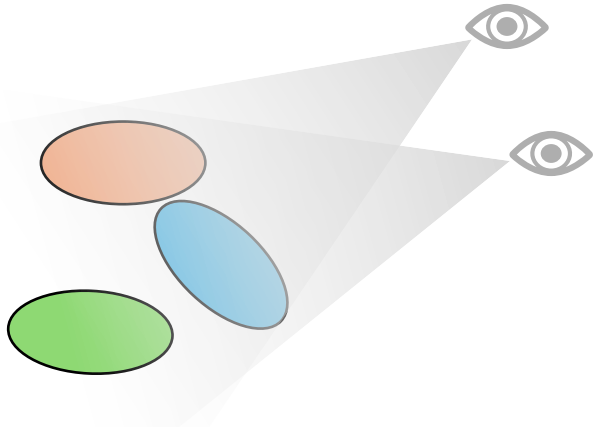
1. Depth Distortion Loss



2. Normal Consistency Loss

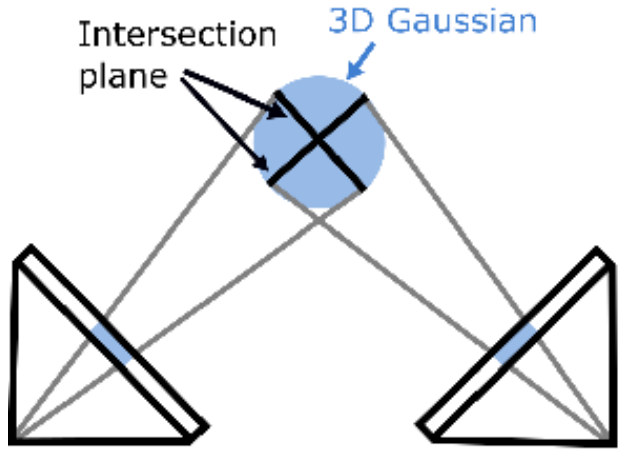


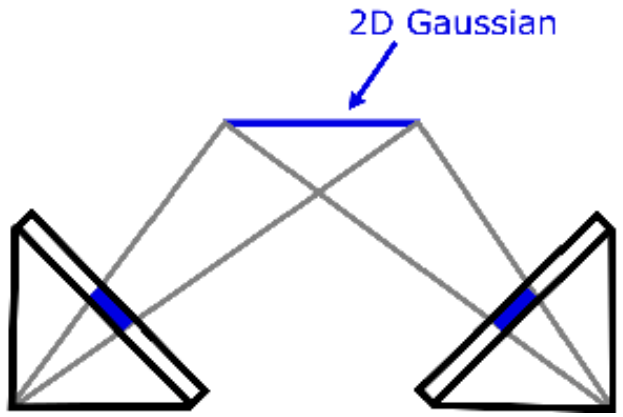
**3. Flattening Gaussians**

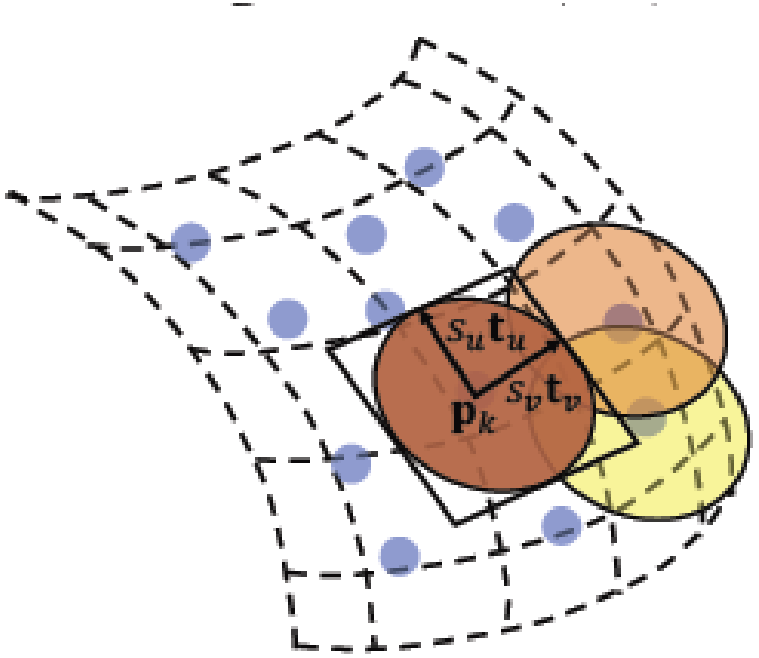


4. Multi-View Loss

# Flatten Gaussians: 2D Gaussian Splatting

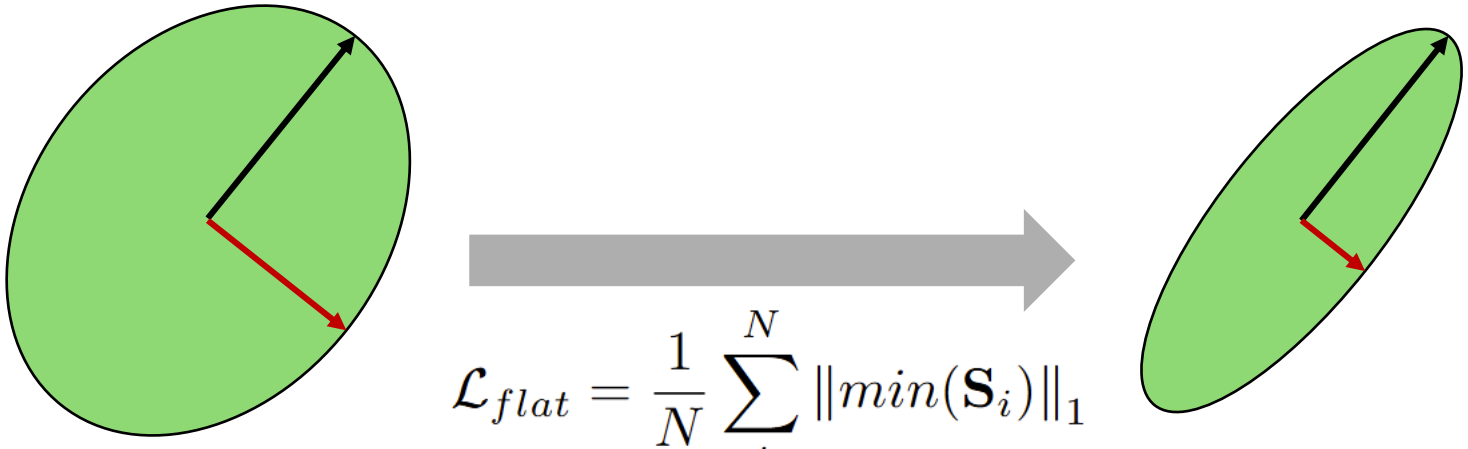



[Huang et al. 2024]

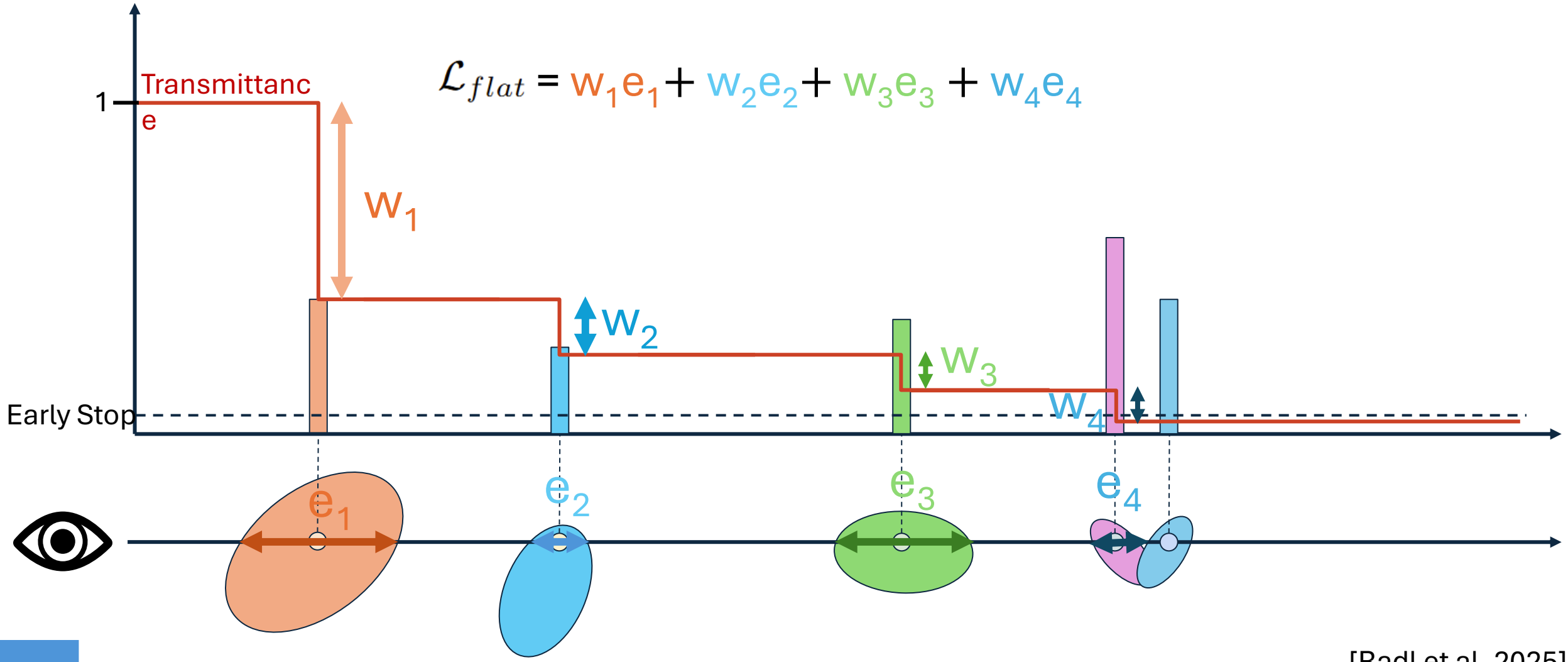
# Flatten Gaussians: Minimize Scale



$$\mathcal{L}_{flat} = \frac{1}{N}\sum_{i}^{N} \|min(\mathbf{S}_i)\|_1$$

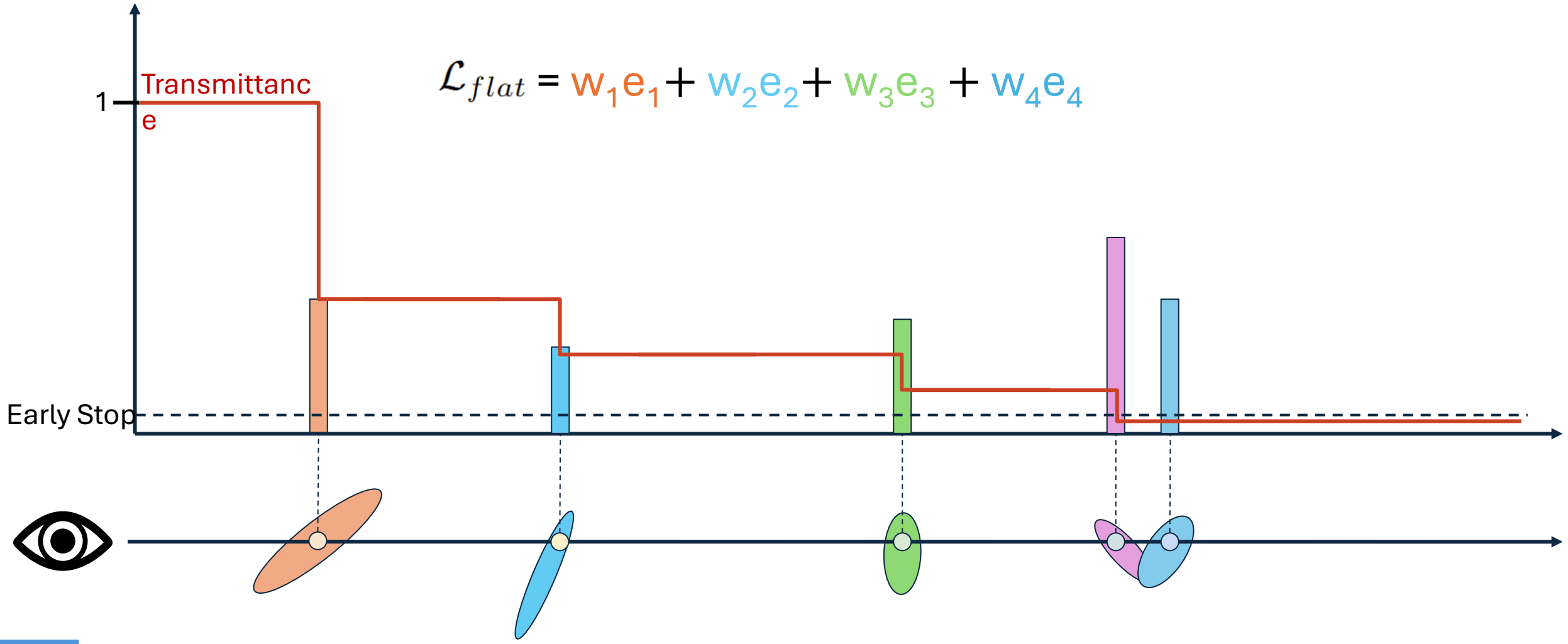
[Guedon and Lepetit, 2023]

# Flatten Gaussians: Extent Loss




[Radl et al. 2025]

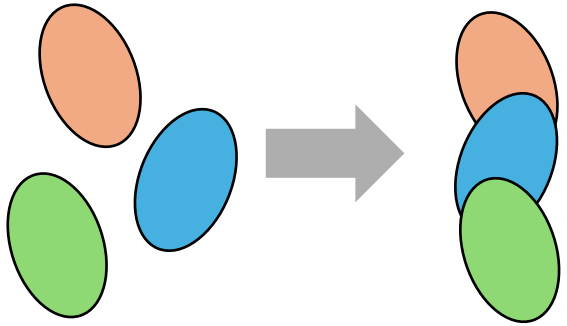
# Flatten Gaussians: Extent Loss



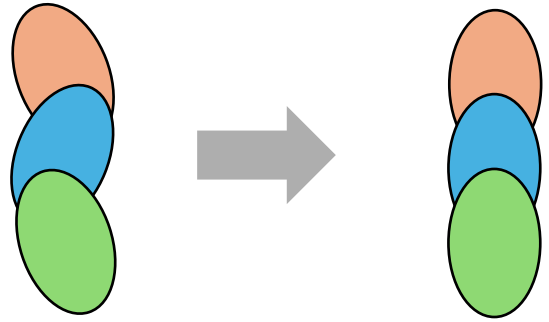

[Radl et al. 2025]

# Geometric Losses

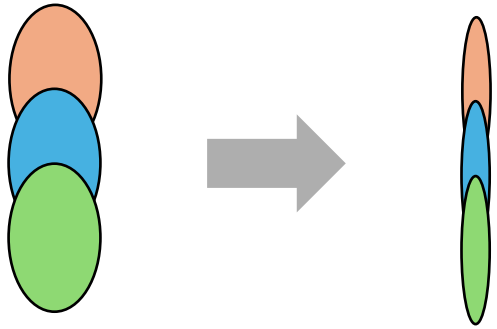
$$\mathcal{L}_{geometry} = \mathcal{L}_{dist} + \mathcal{L}_{normal} + \mathcal{L}_{flat} + \mathcal{L}_{multiview}$$



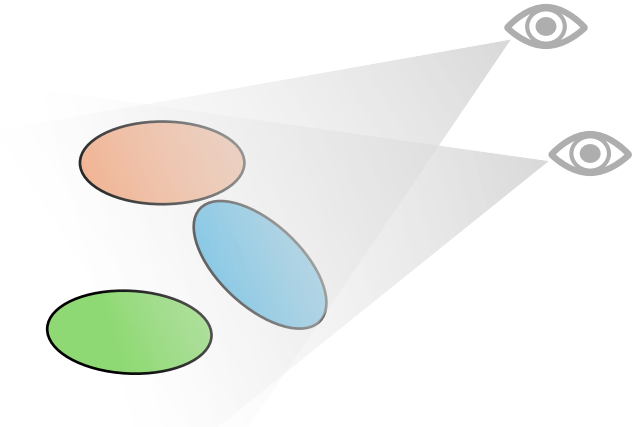
1. Depth Distortion Loss



2. Normal Consistency Loss

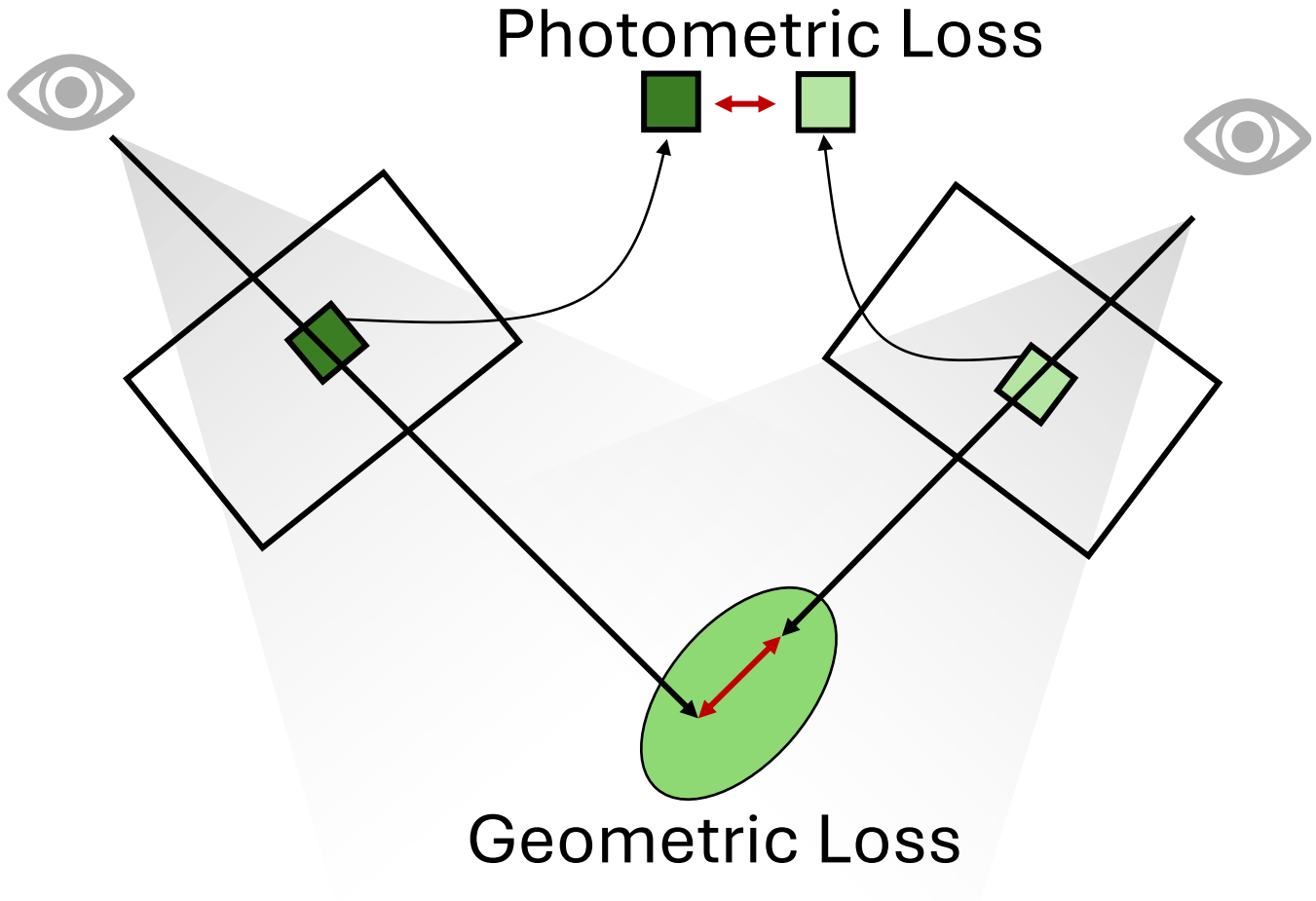


3. Flattening Gaussians



**4. Multi-View Loss**

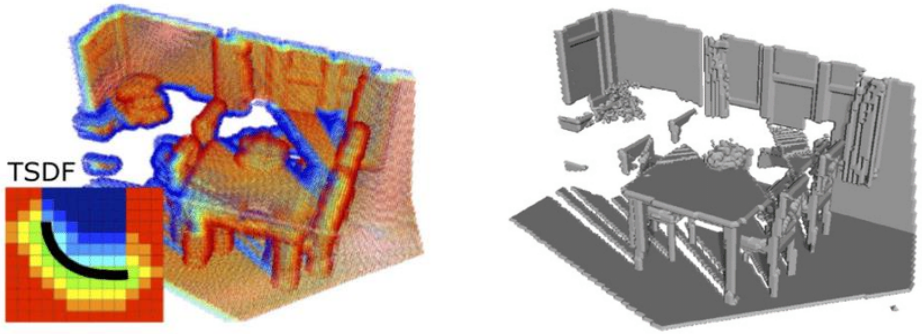
# Multi-View Loss

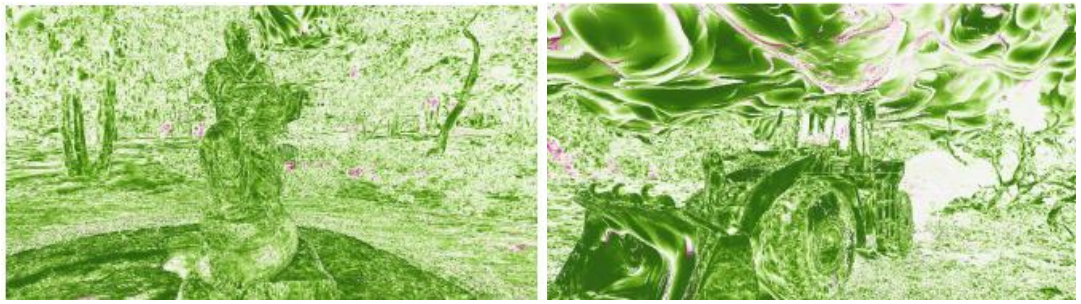


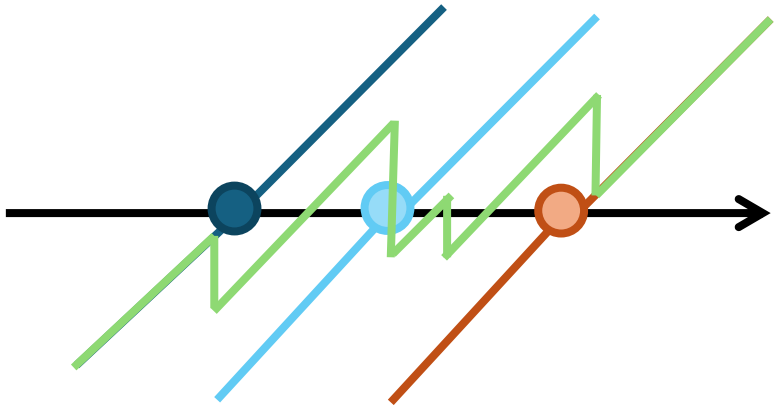


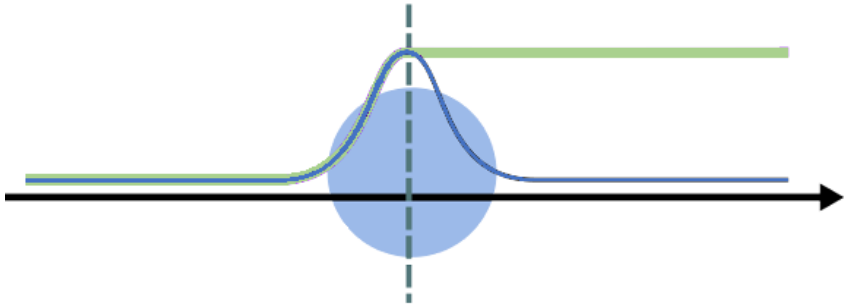

[Chen et al. 2025] PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstr

# Part 2: Meshing





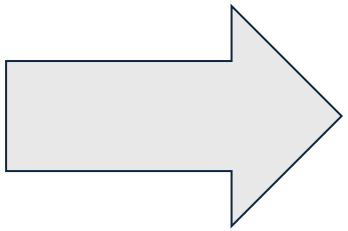


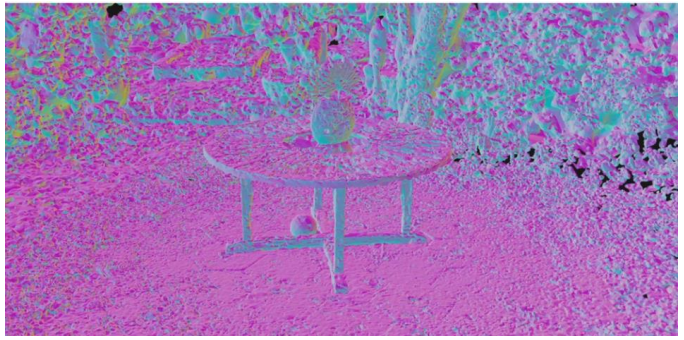
TSDF Volume Fusion

Gaussian Opacity Fields

[Roldao, 2022] 3D Semantic Scene Completion: A Survey
[Radl et al. 2025]

# Meshing: Marching Cubes

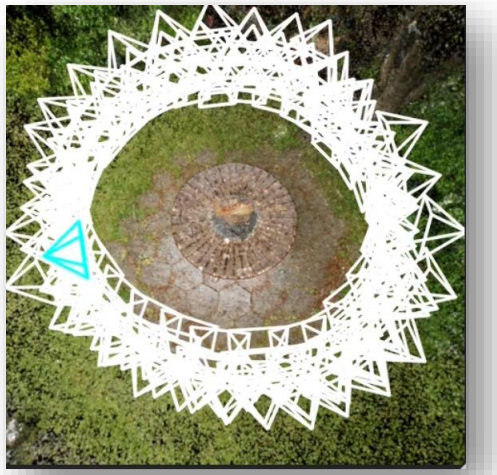




[Guedon and Lepetit,

# Meshing: Camera Poses

3D Gaussians

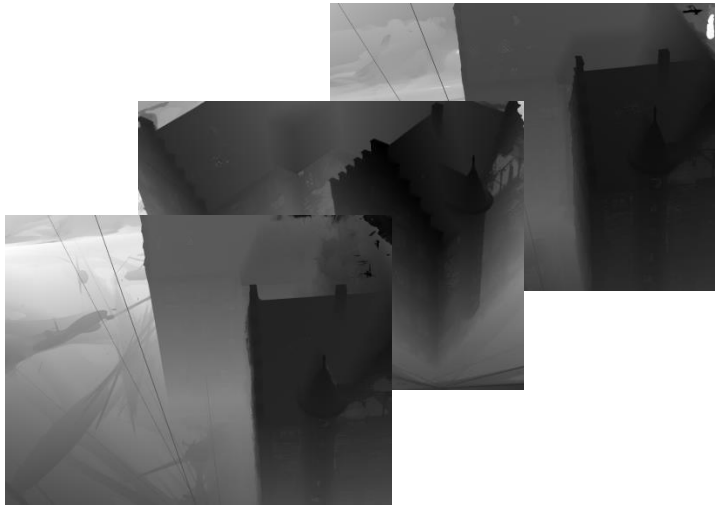


Camera Poses

Better Mesh

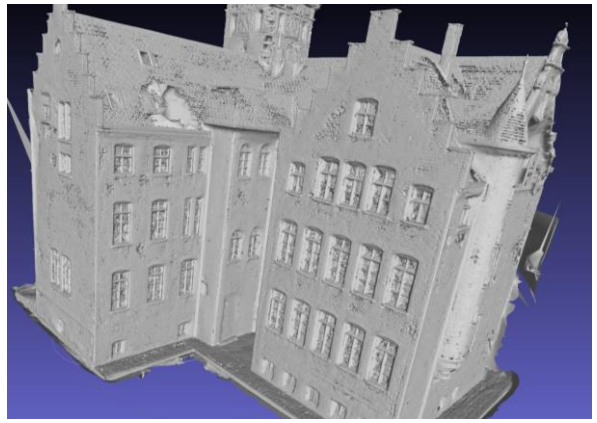
# Truncated Signed Distance Volume Fusion (TSDF)

3D Gaussians



Posed Depth Maps

TSDF

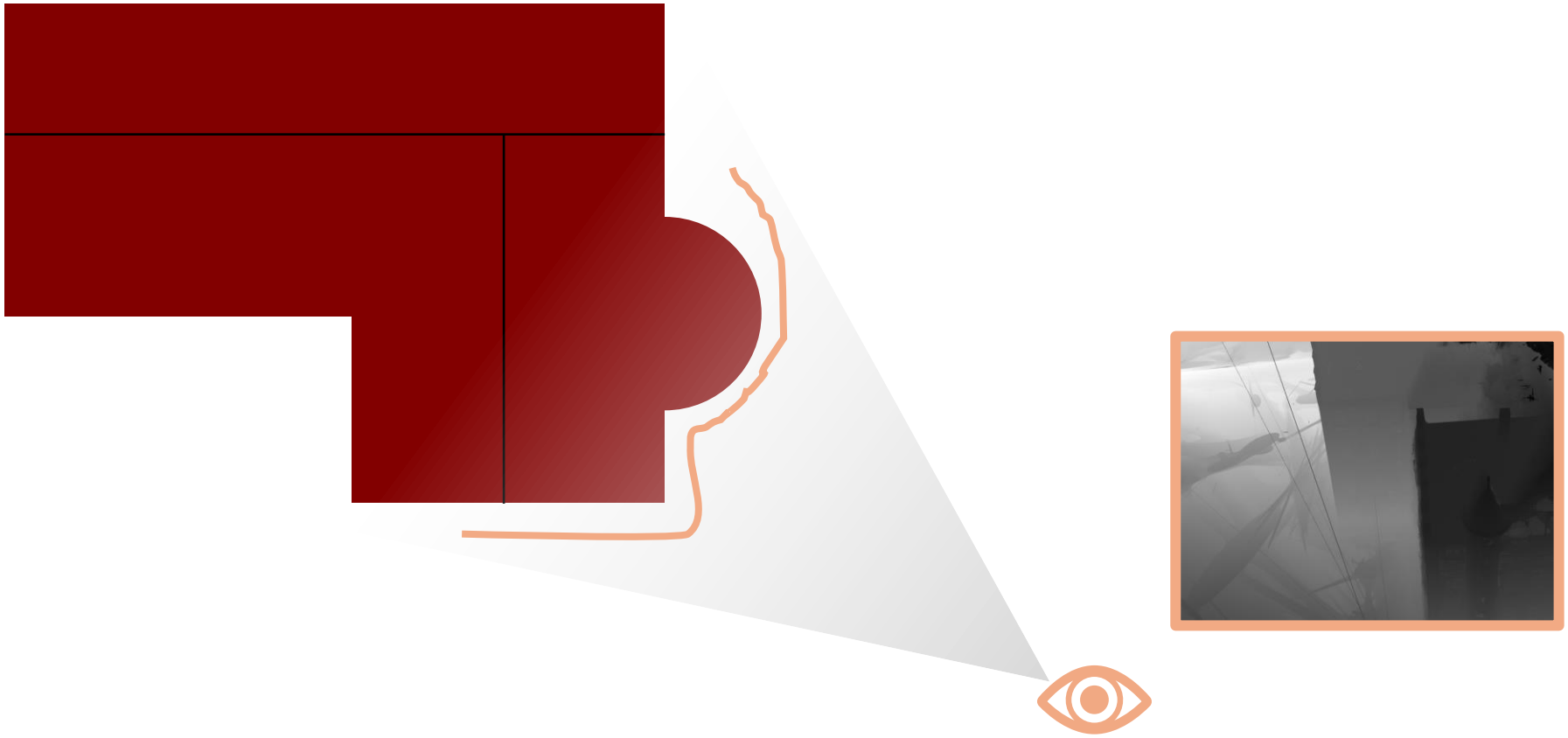


Mesh



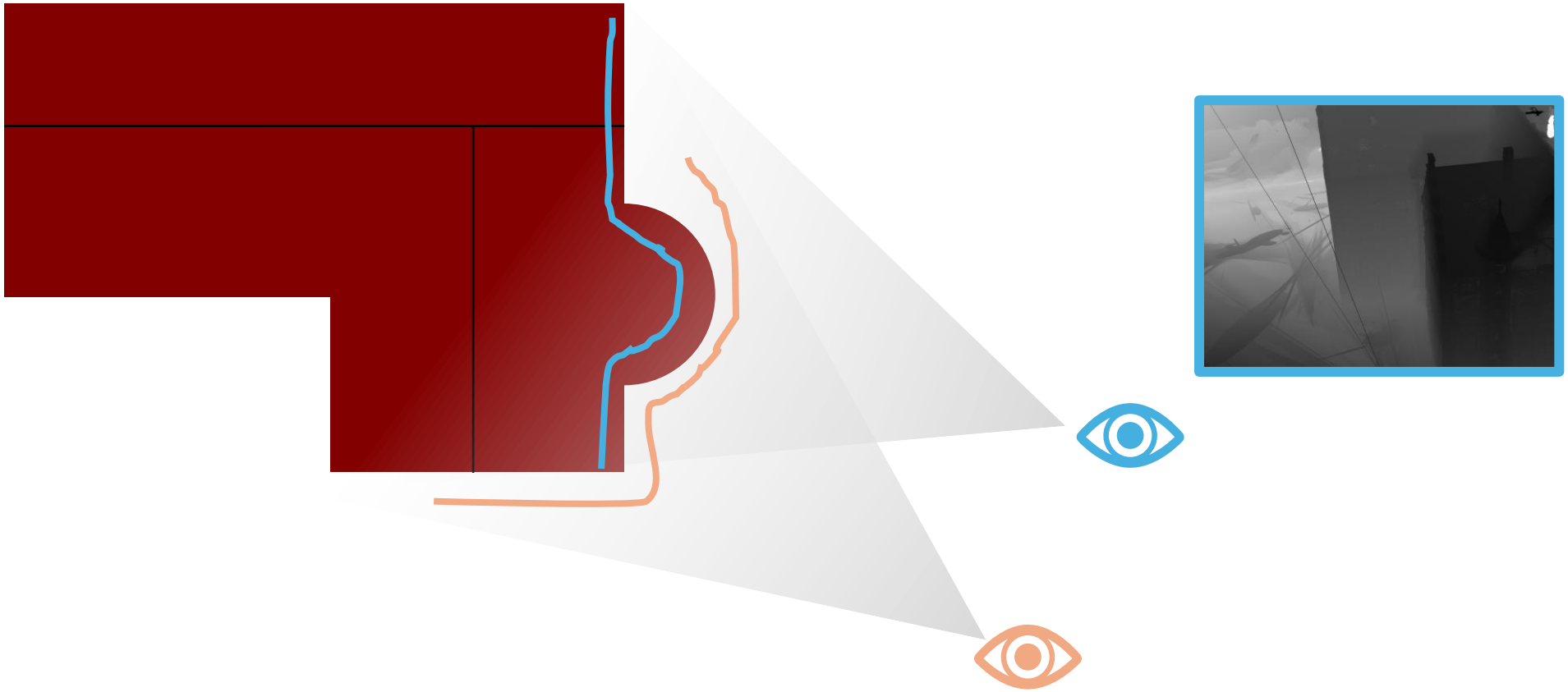
[Curless and Levoy, 1996] A Volumetric Method for Building Complex Models from Range Imag

# Truncated SDF



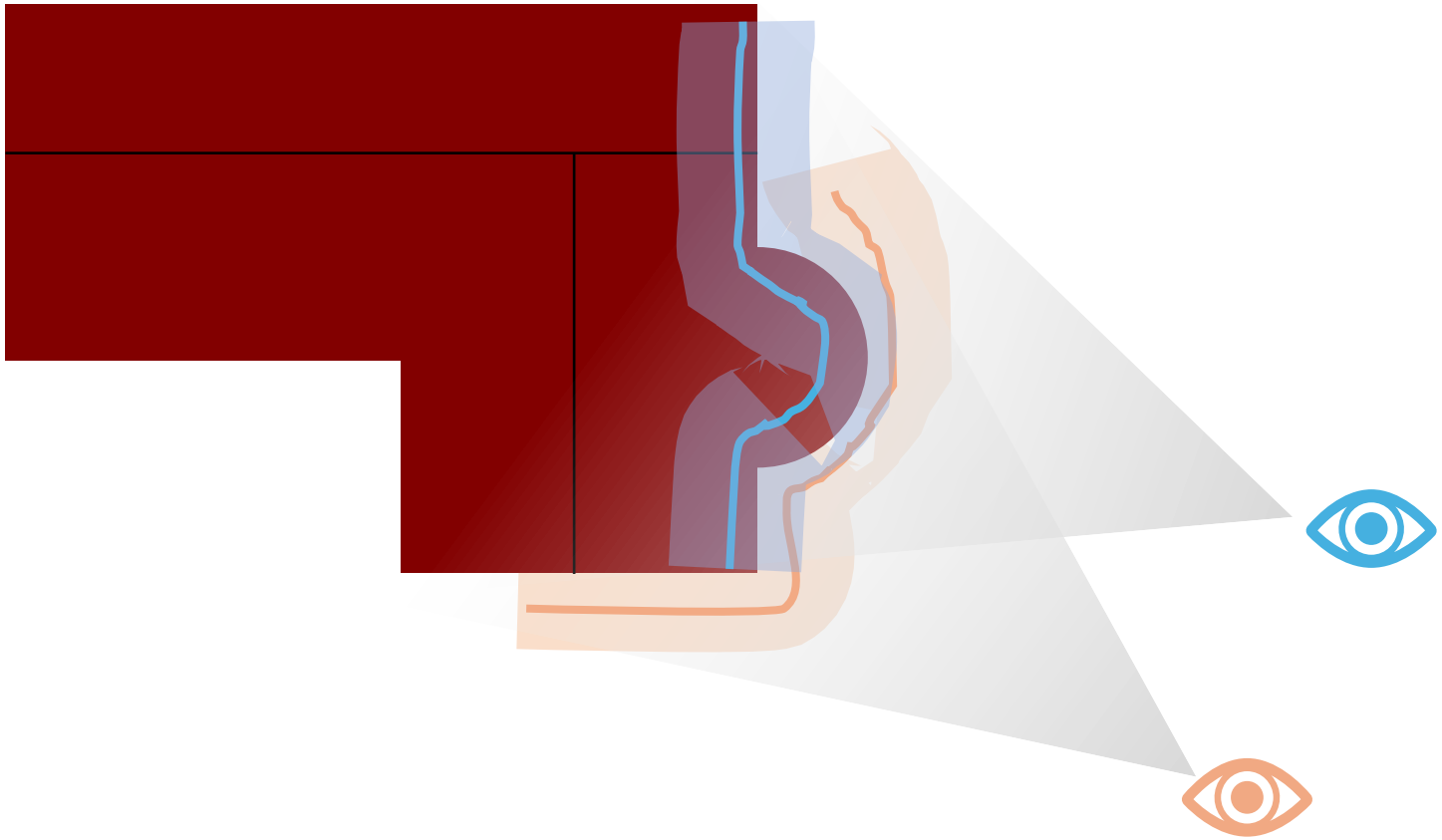
[Curless and Levoy, 1996

# Truncated SDF



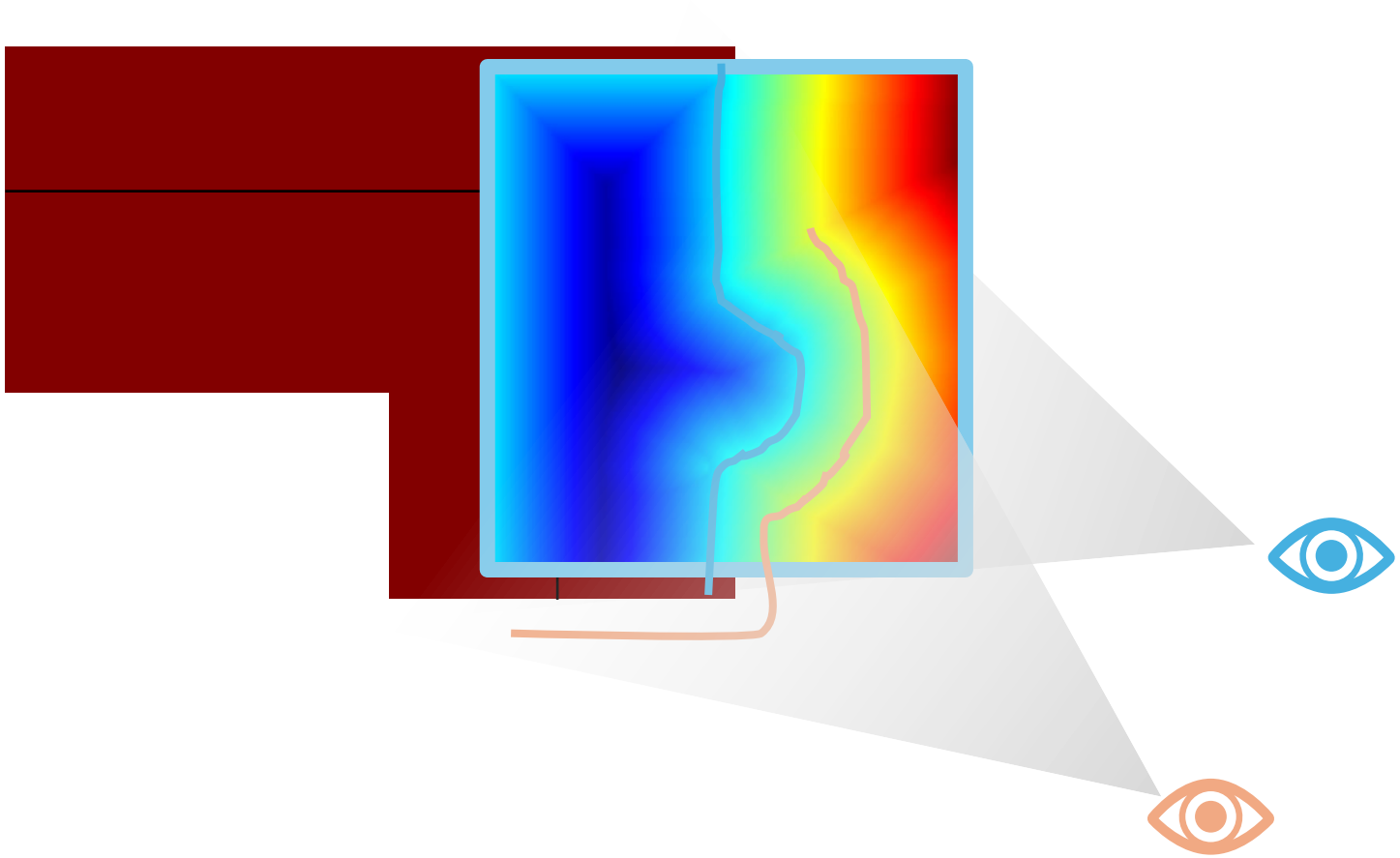
[Curless and Levoy, 1996

# Truncated SDF



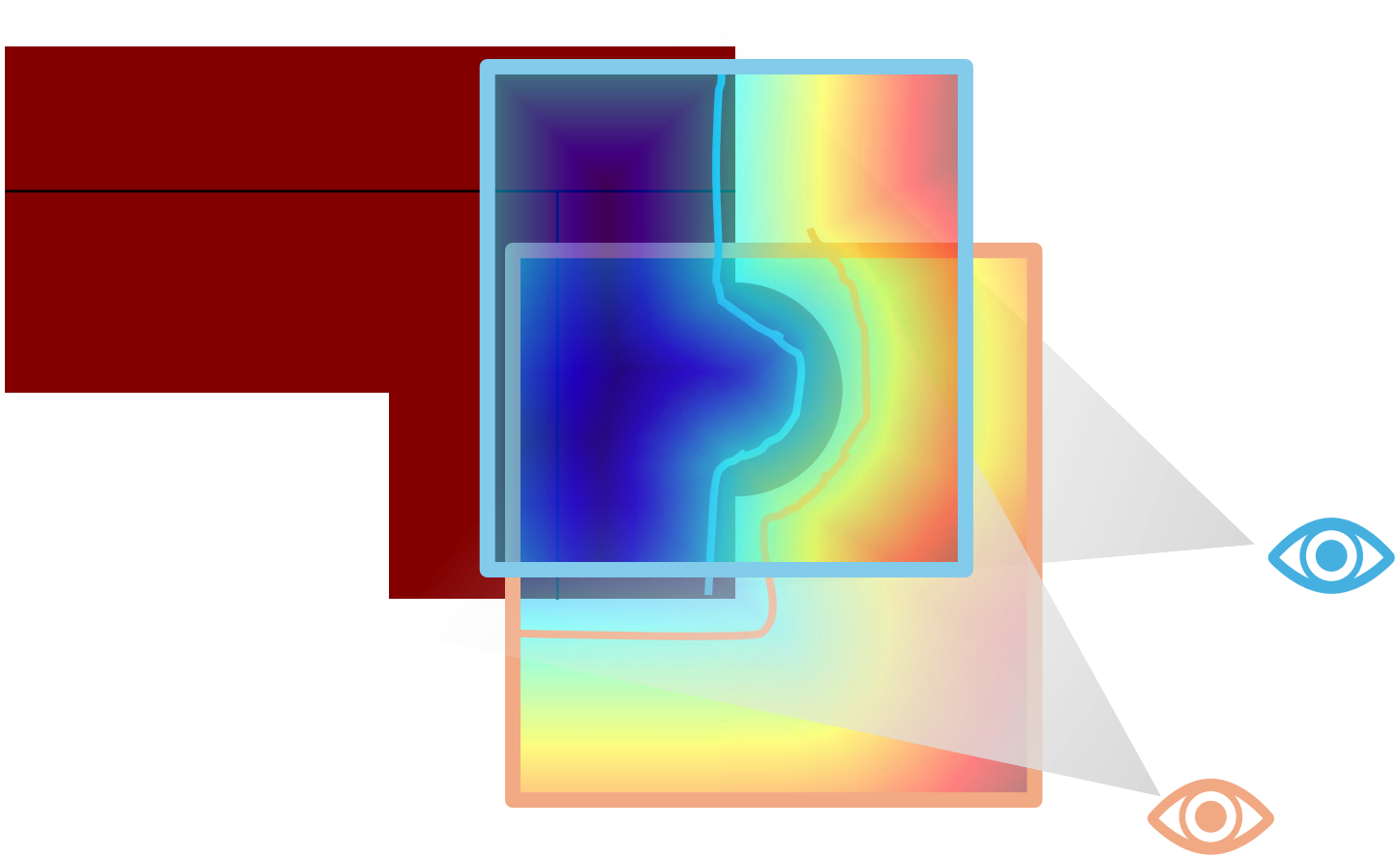
[Curless and Levoy, 1996

# Truncated SDF



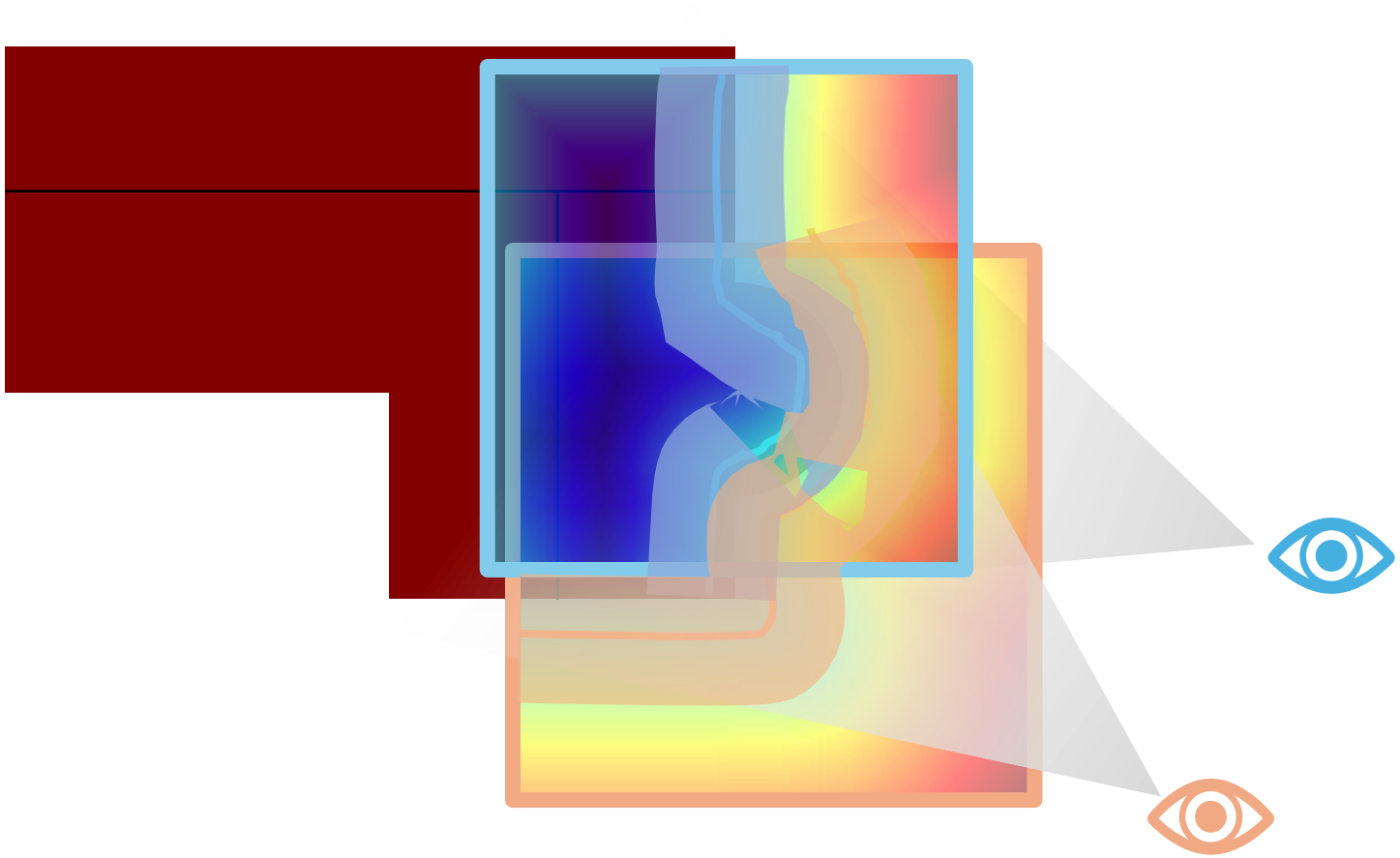
[Curless and Levoy, 1996

# Truncated SDF



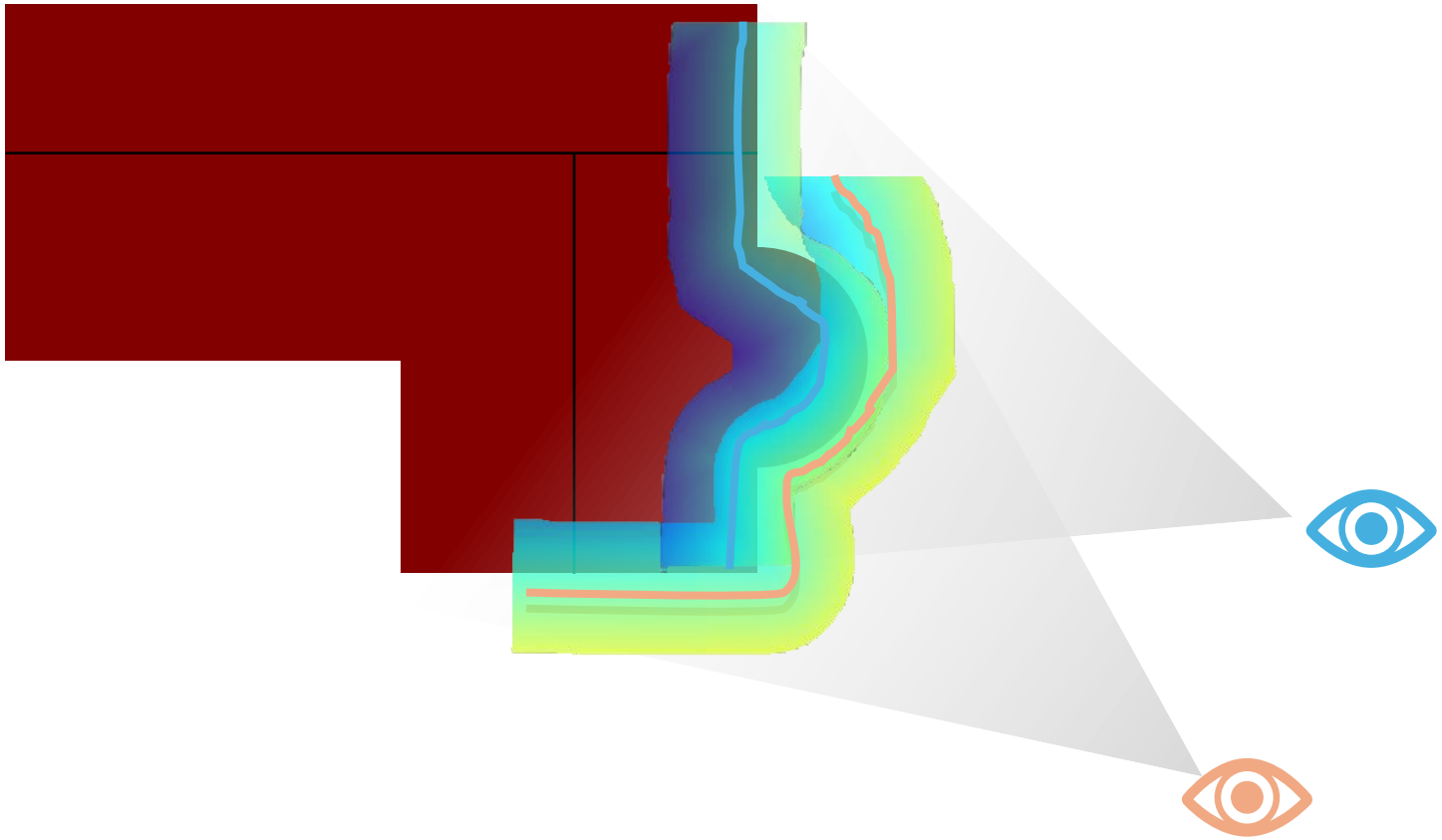
[Curless and Levoy, 1996

# Truncated SDF



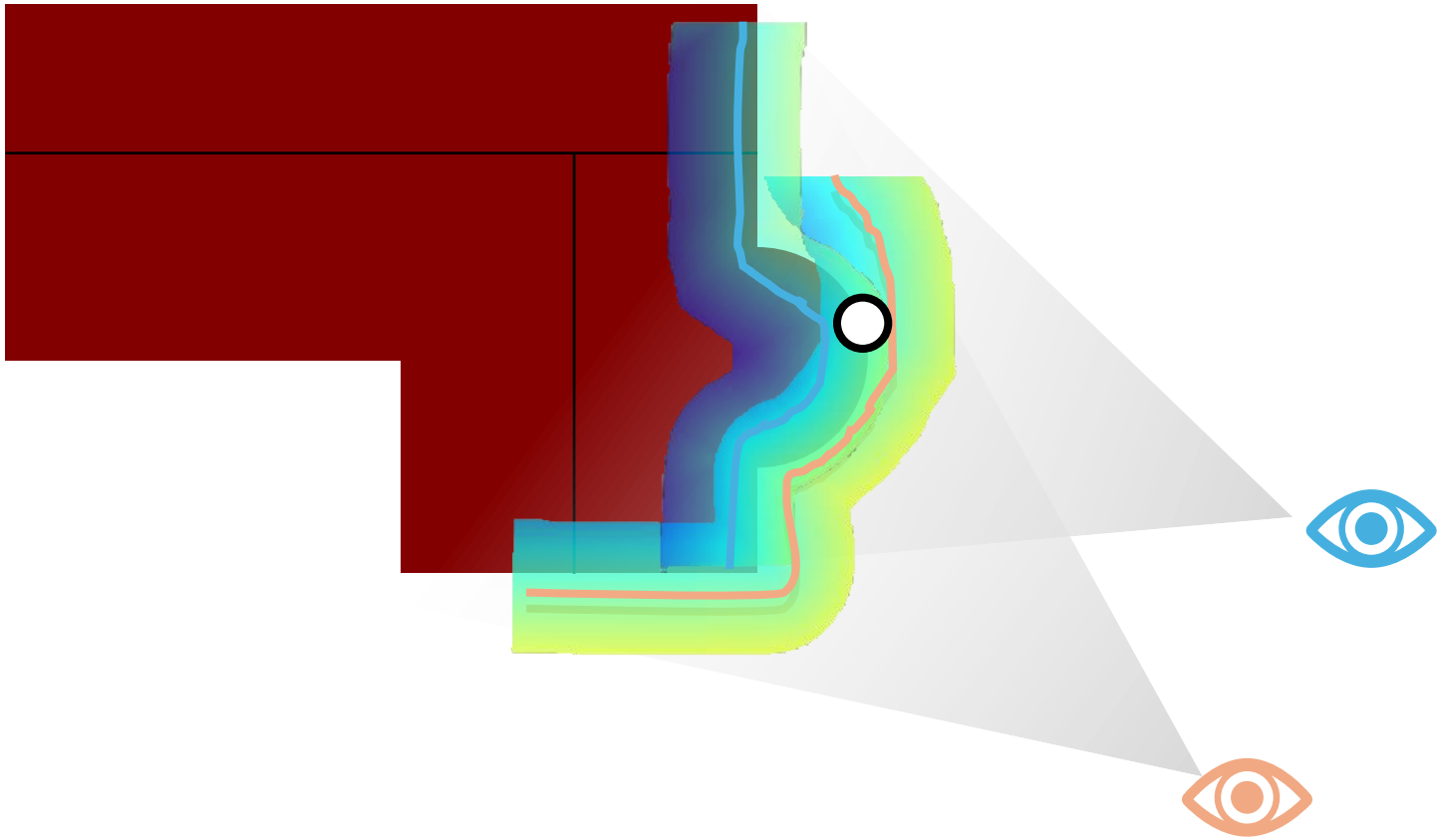
[Curless and Levoy, 1996

# Truncated SDF



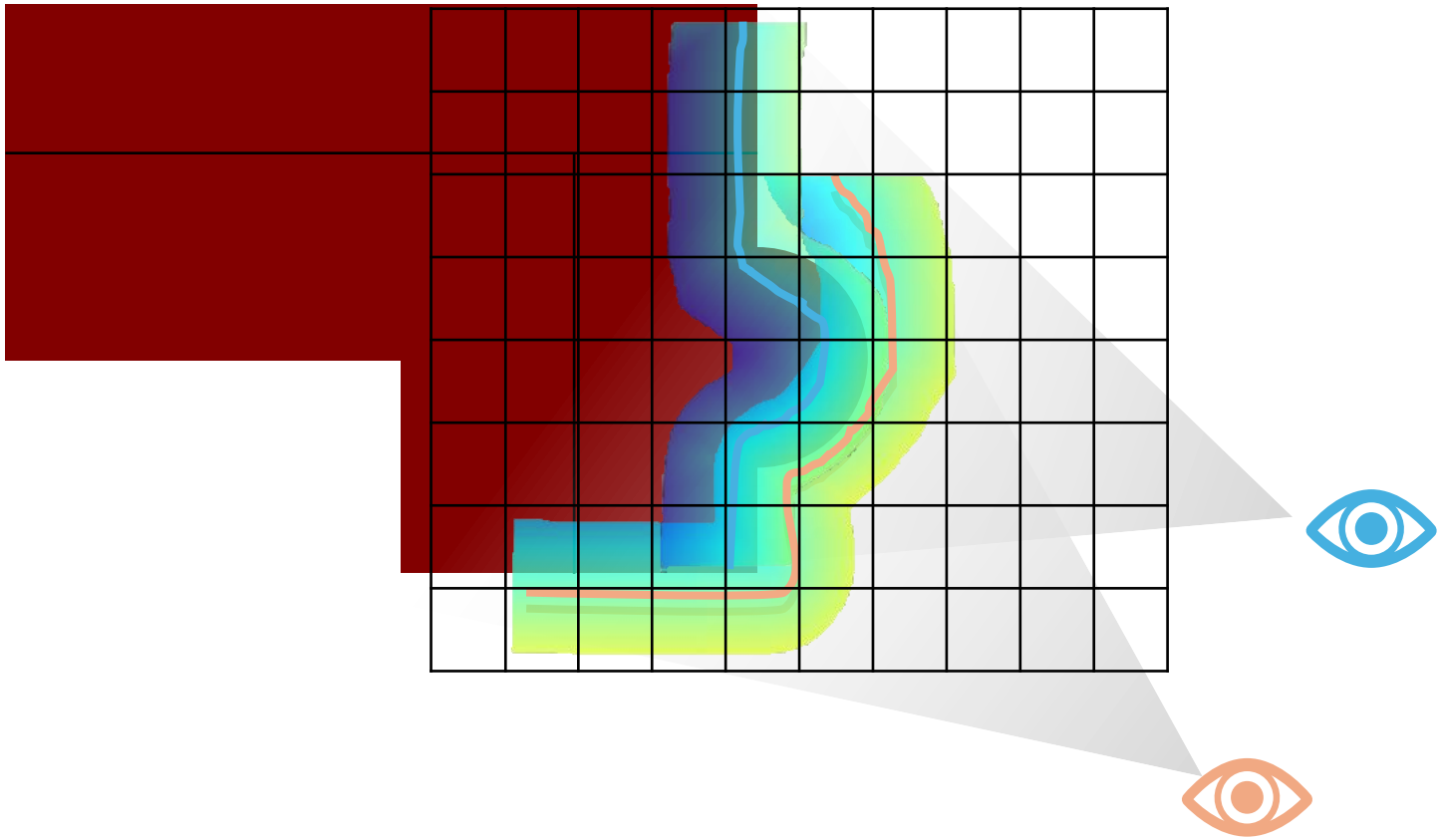
[Curless and Levoy,

# Truncated SDF



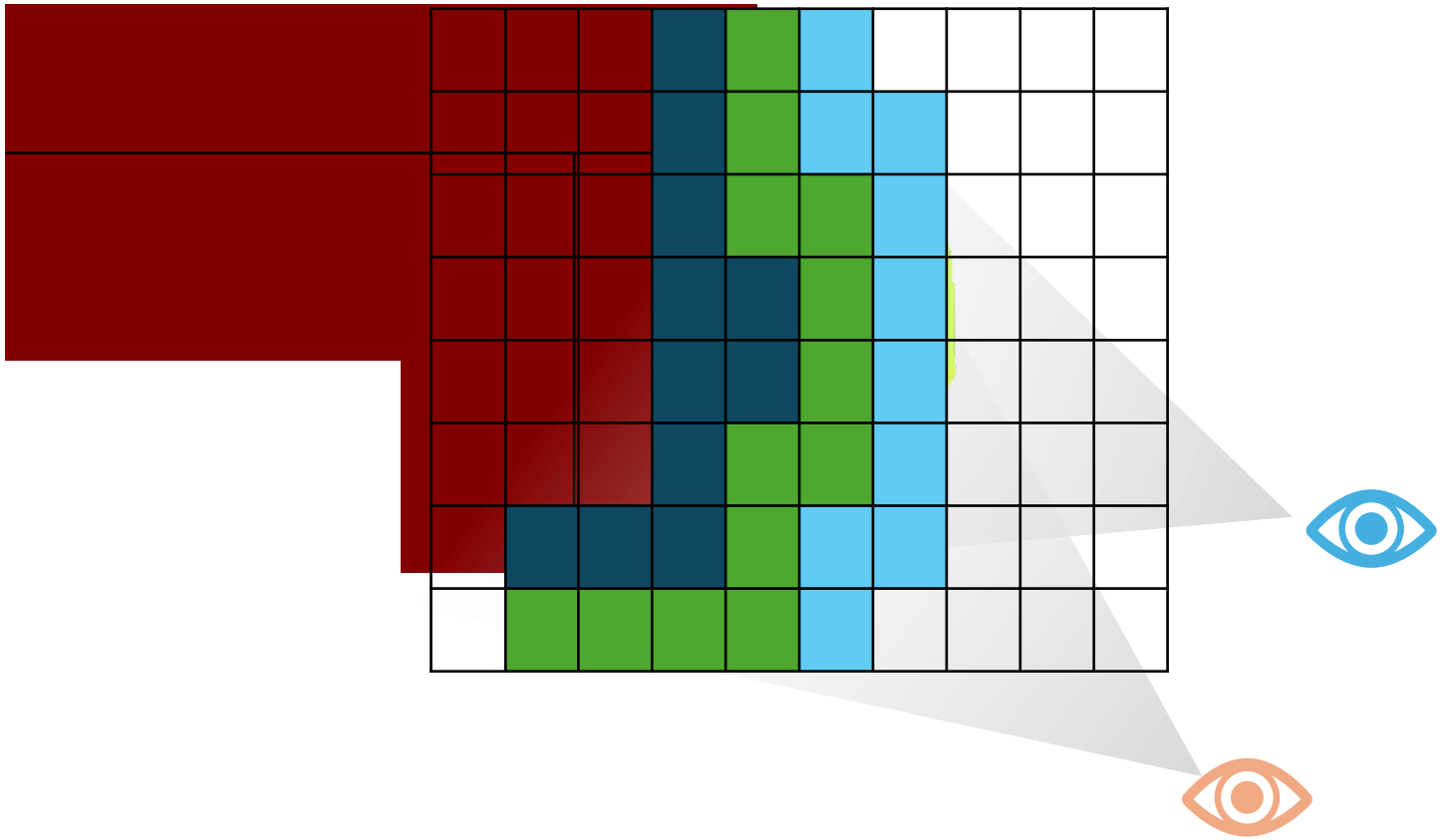
[Curless and Levoy,

# Truncated SDF: Fusion



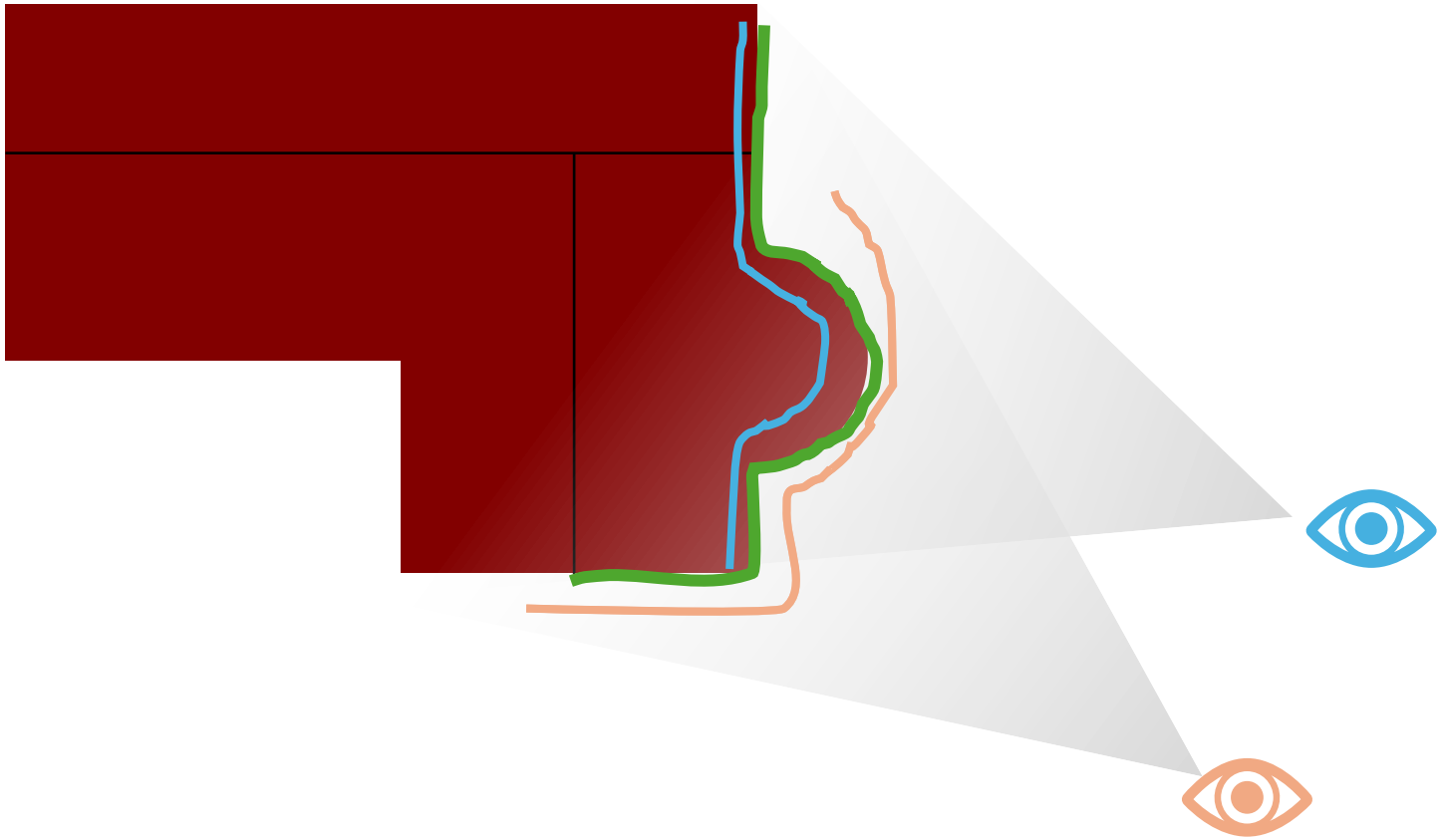
[Curless and Levoy,

# Truncated SDF: Fusion



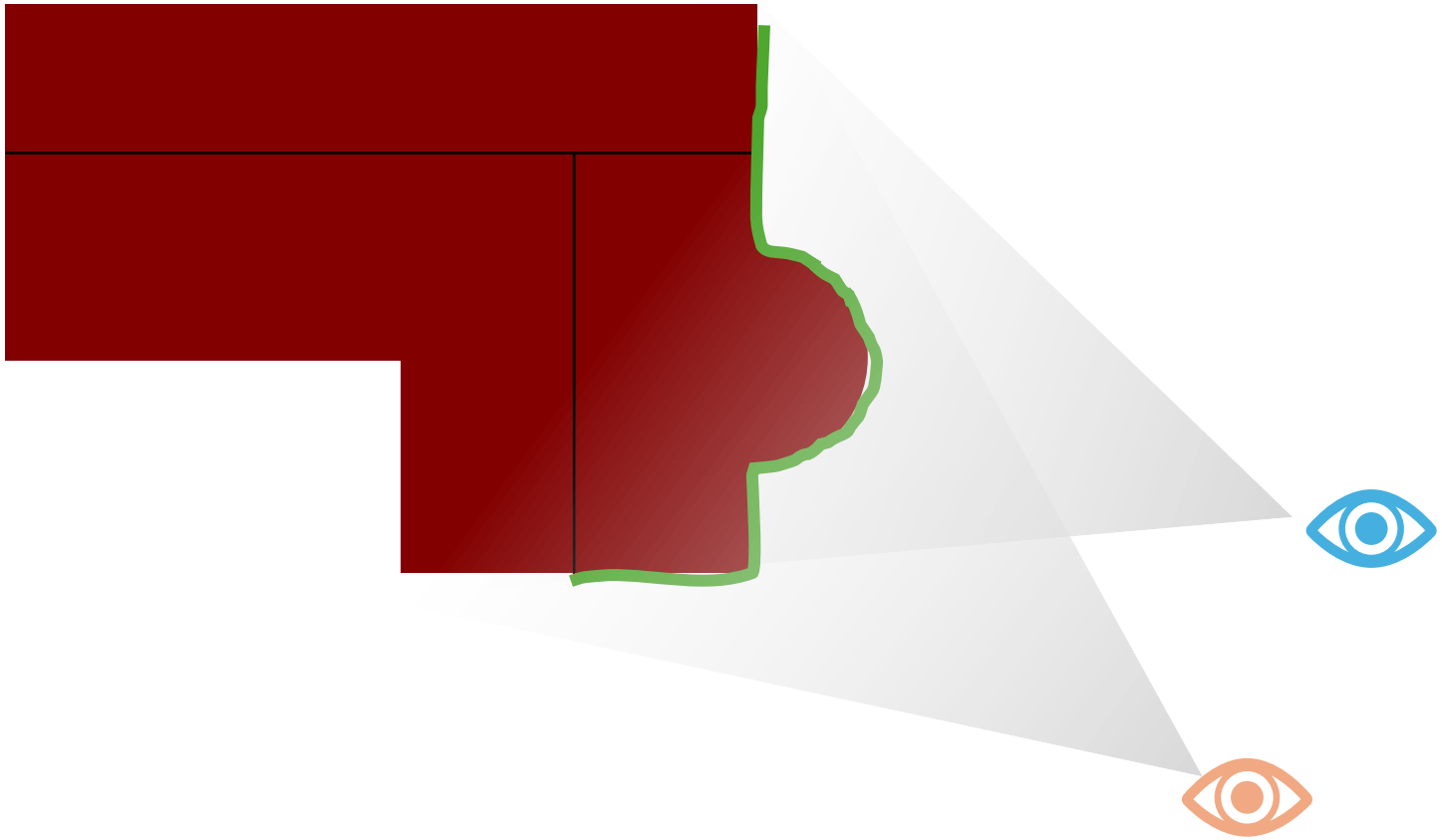
[Curless and Levoy,

# Truncated SDF: Marching Cubes



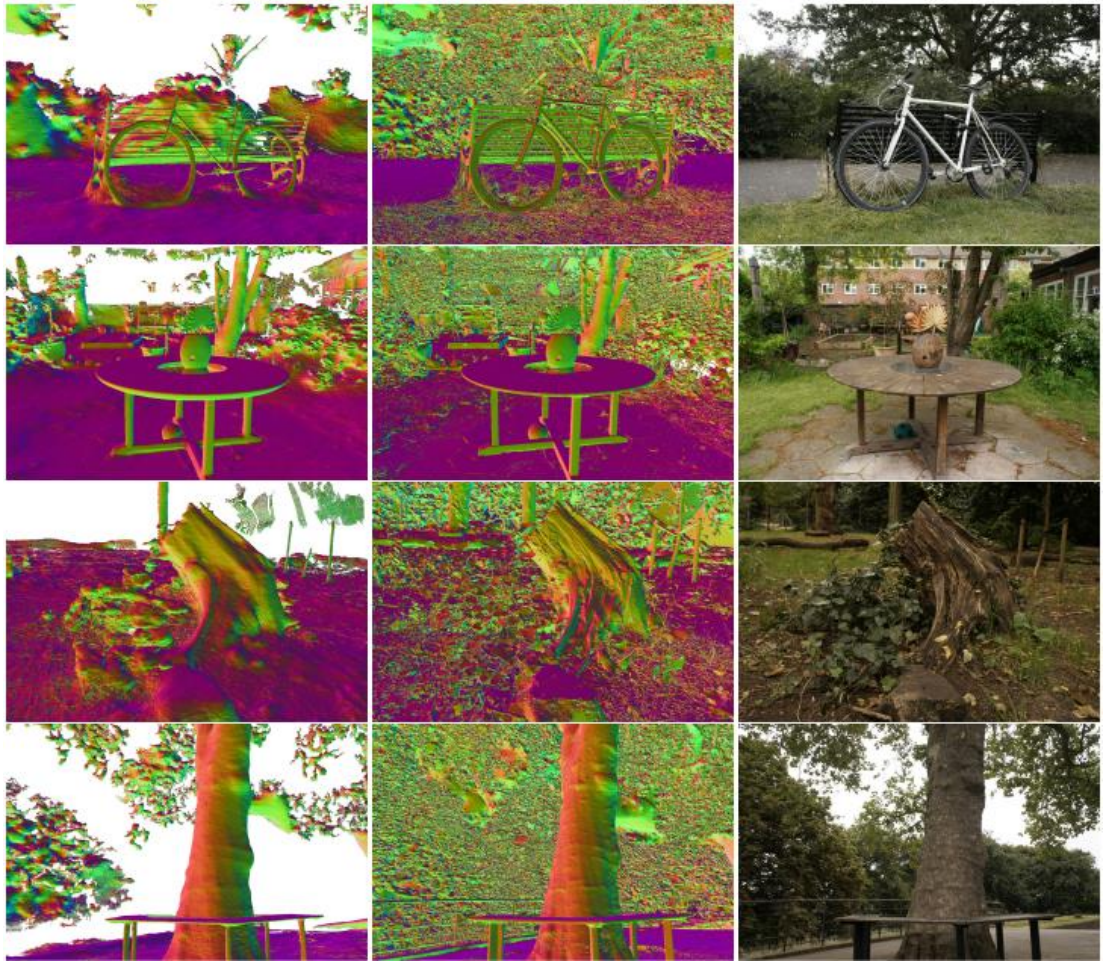
[Curless and Levoy,

# Truncated SDF: Marching Cubes



[Curless and Levoy,

# TSDF struggles with Large, Open Areas



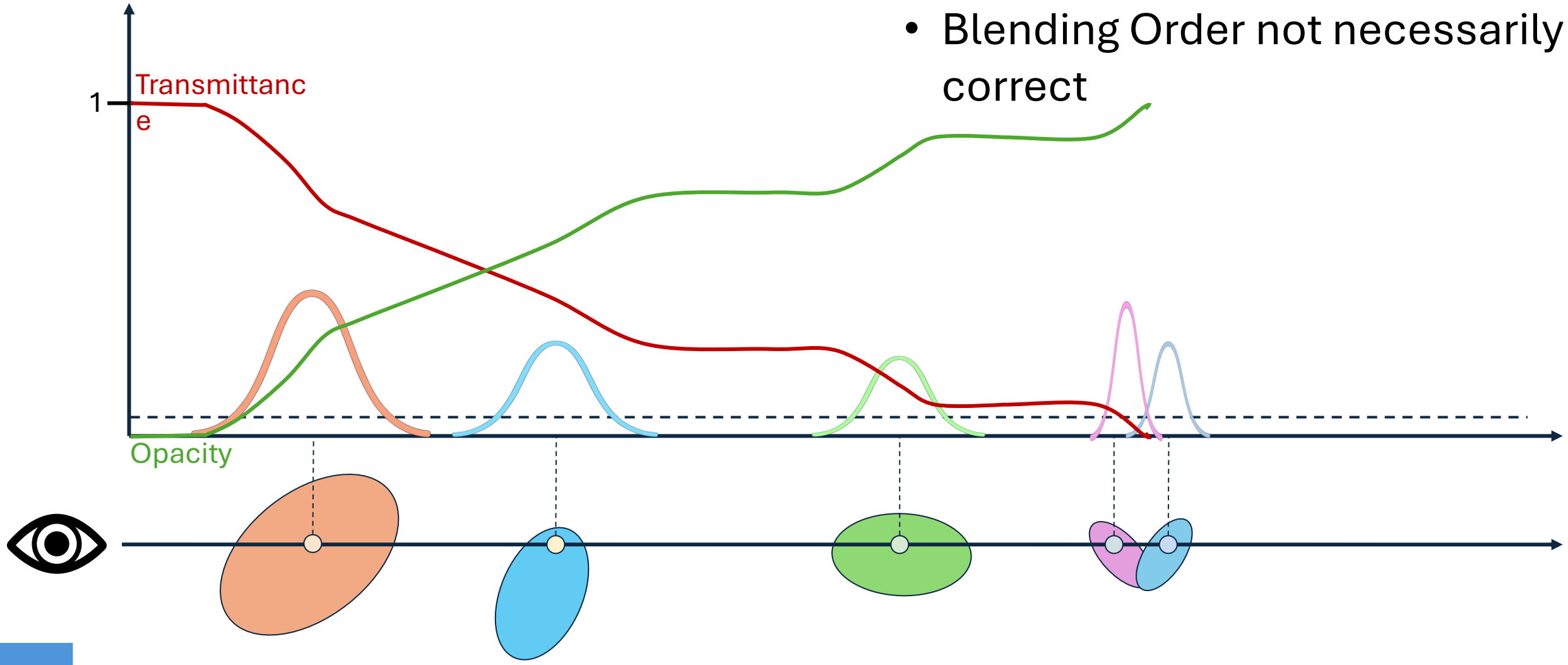
2DGS w/ TSDF | Gaussian Opacity Fields | Ground Truth

[Huang et al. 2024]
[Yu et al. 2024]

# Volume Rendering

Assumptions:

- No Self Occlusion
- No contribution after mean
- Blending Order not necessarily correct

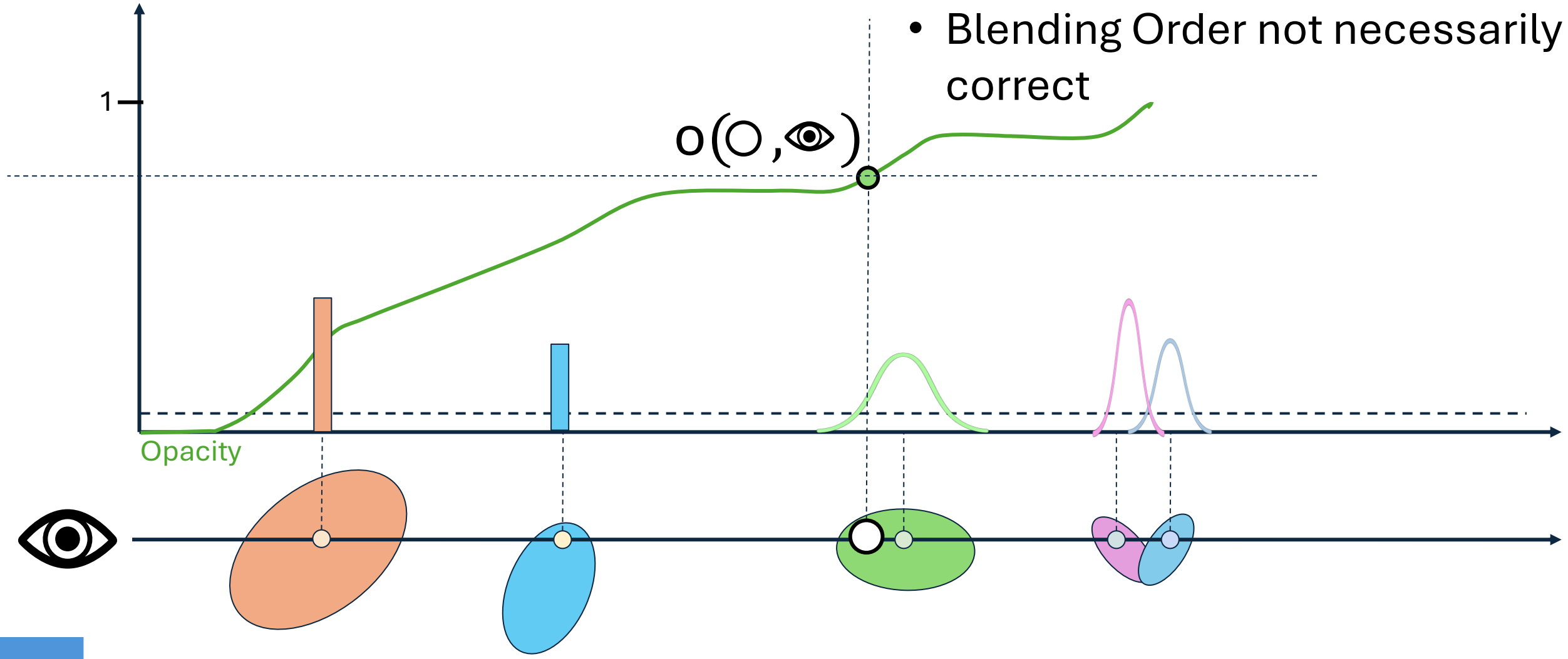



[Yu et al. 2024]

# Volume Rendering

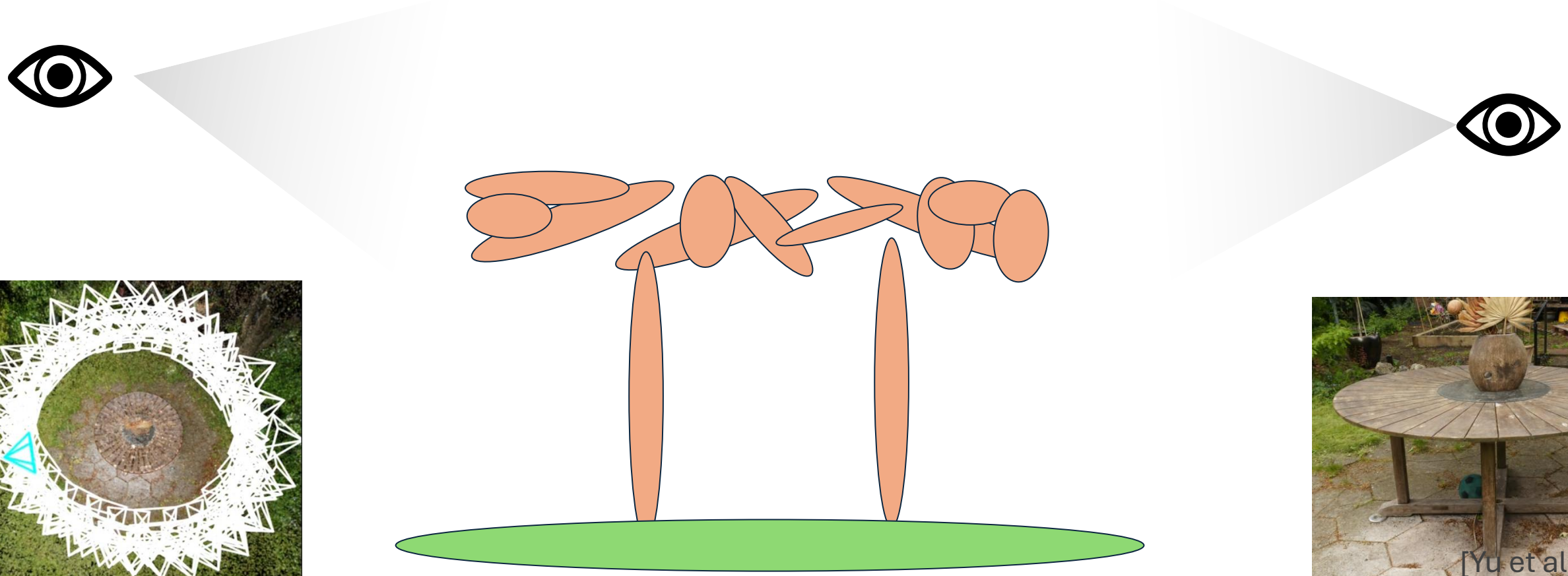
Assumptions:

- No Self Occlusion
- No contribution after mean
- Blending Order not necessarily correct




[Yu et al. 2024]

# Gaussian Opacity Fields

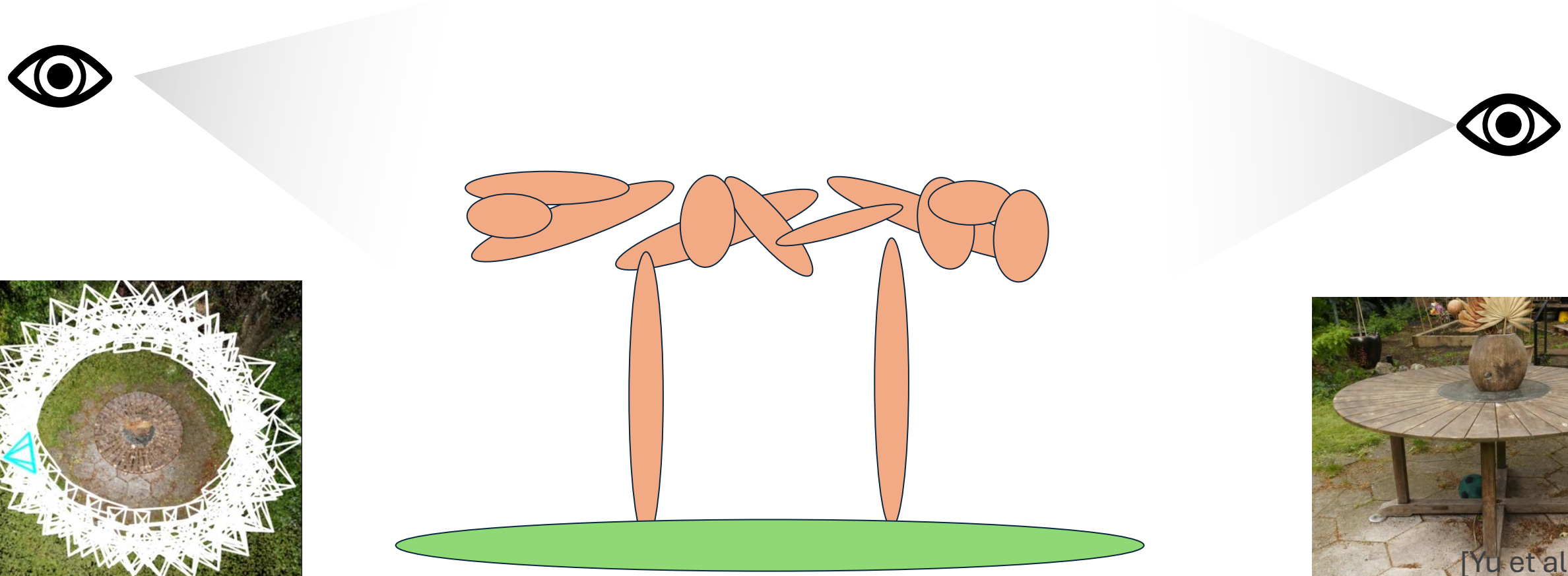






[Yu et al. 2024]

# Gaussian Opacity Fields

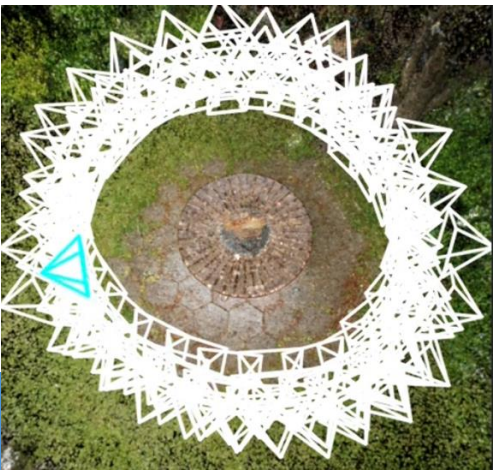






[Yu et al. 2024]

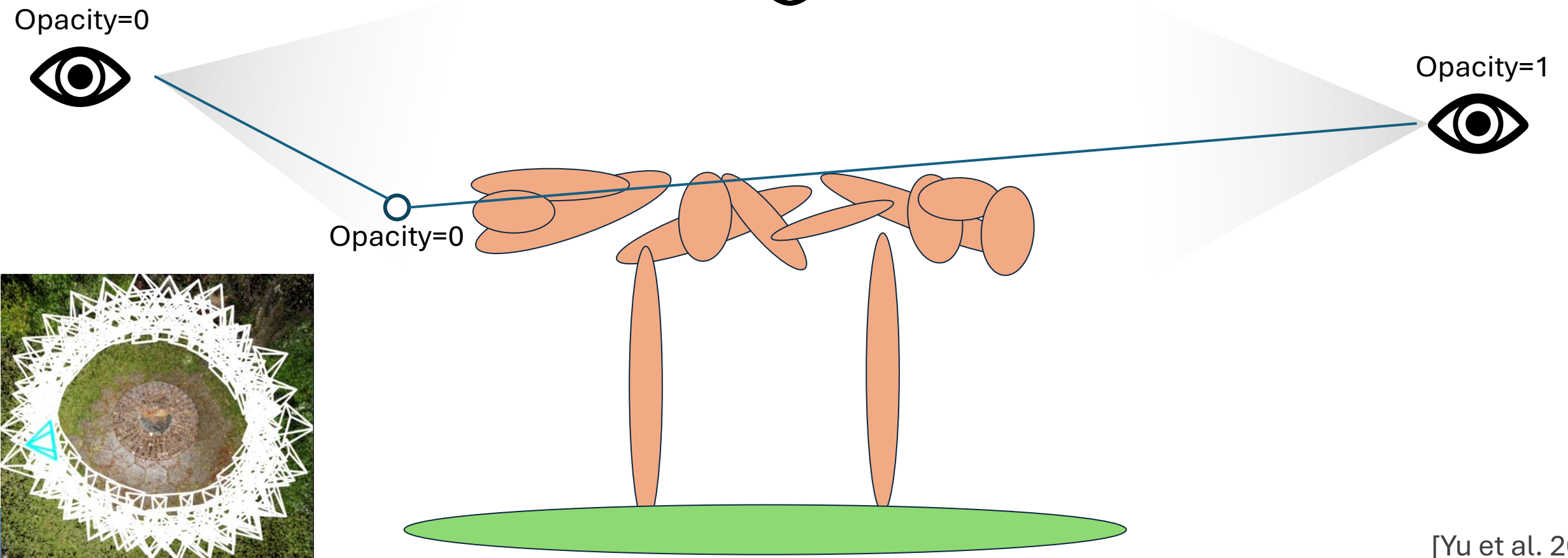
# Gaussian Opacity Fields



[Yu et al. 2024]

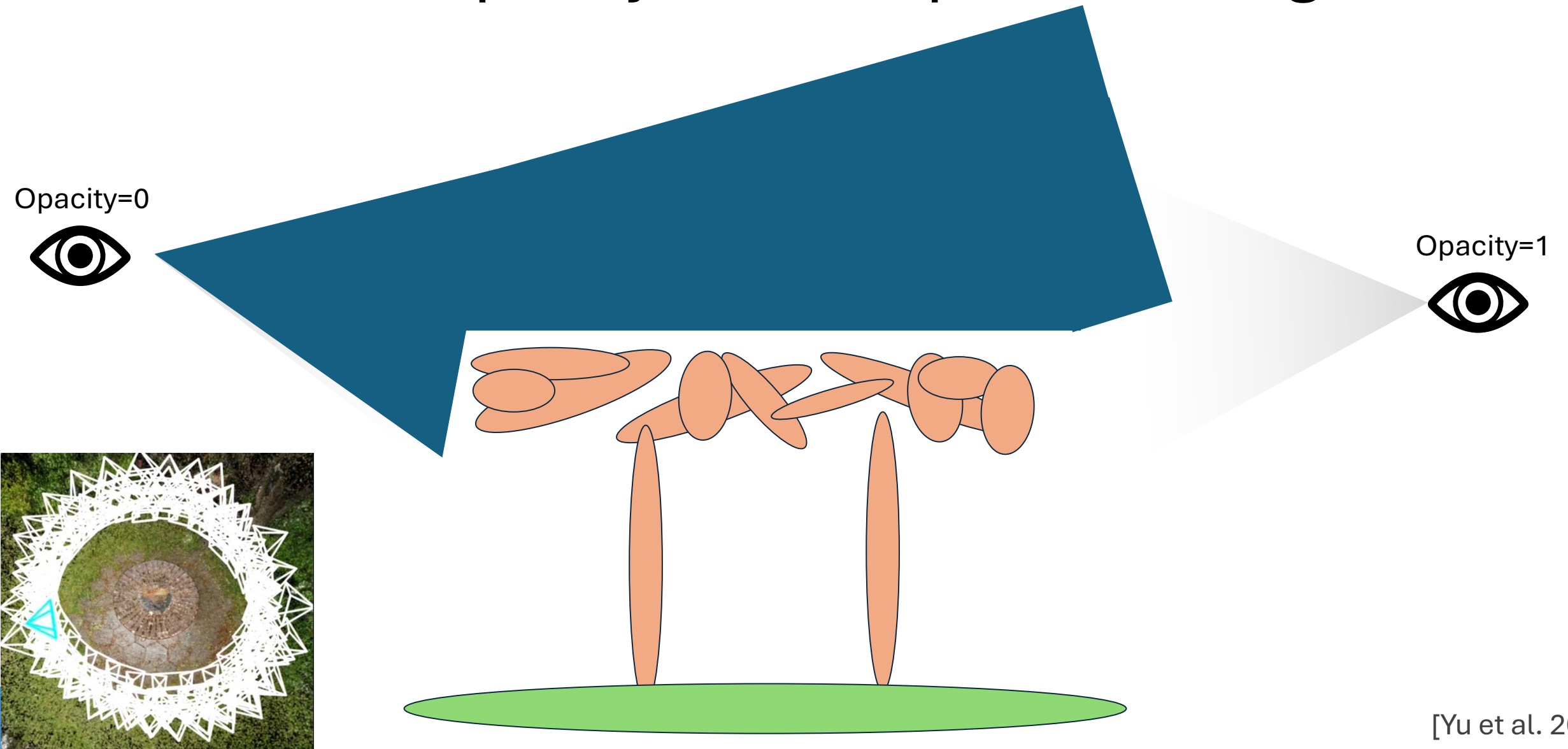
# Gaussian Opacity Fields

$$O(\bigcirc) = \min_{\text{eye}}( o(\bigcirc, \text{eye}) )$$



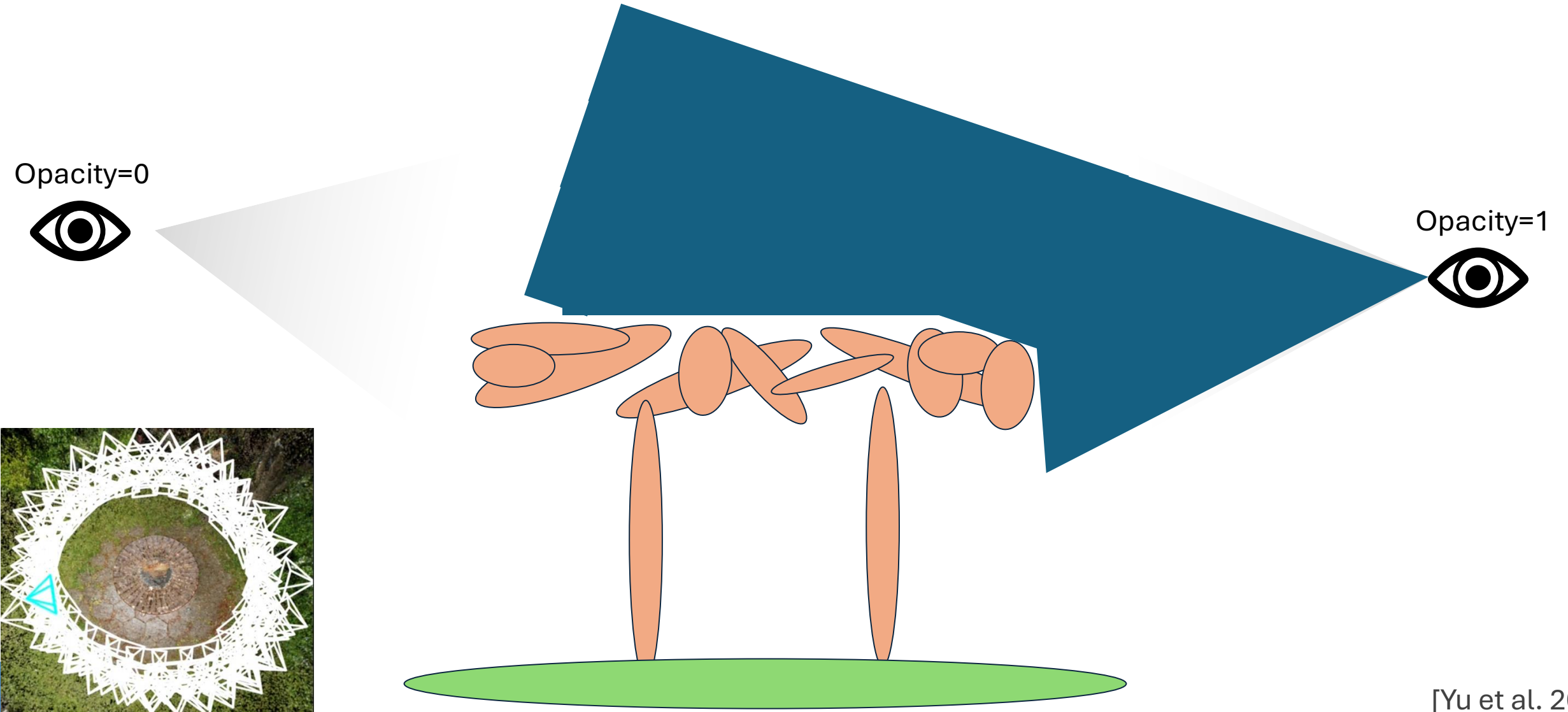

[Yu et al. 2024]

# Gaussian Opacity Fields: Space Carving




[Yu et al. 2024]

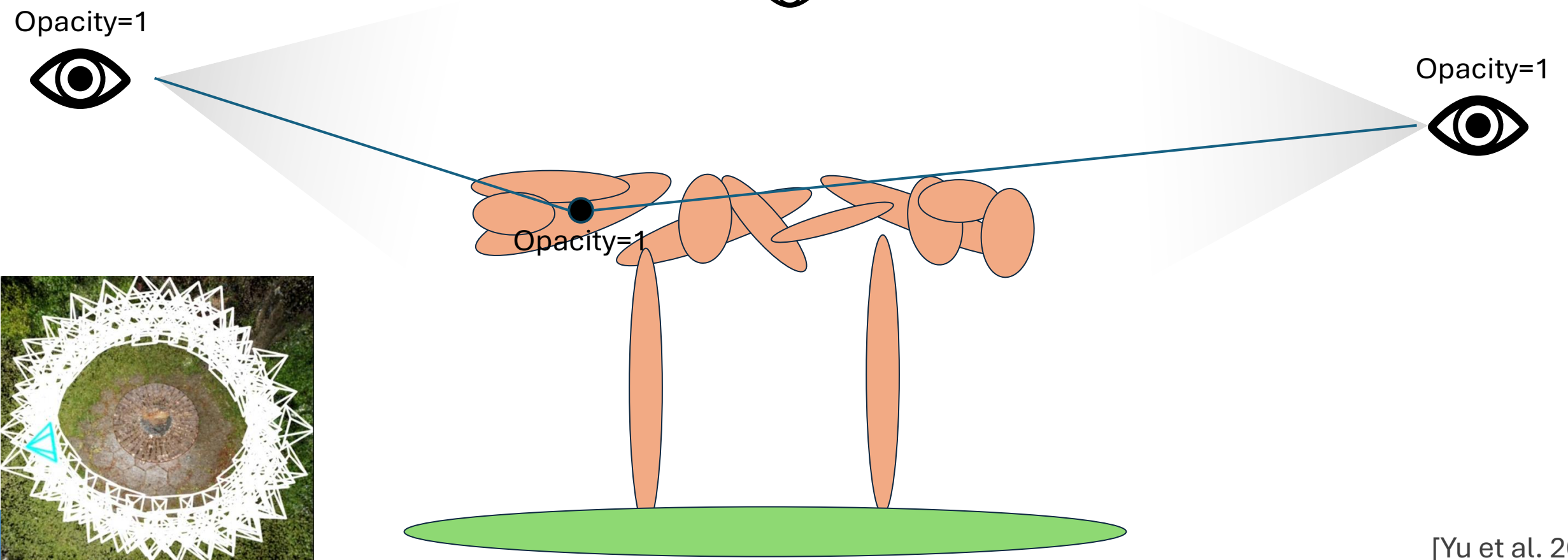
# Gaussian Opacity Fields: Space Carving




[Yu et al. 2024]

# Gaussian Opacity Fields

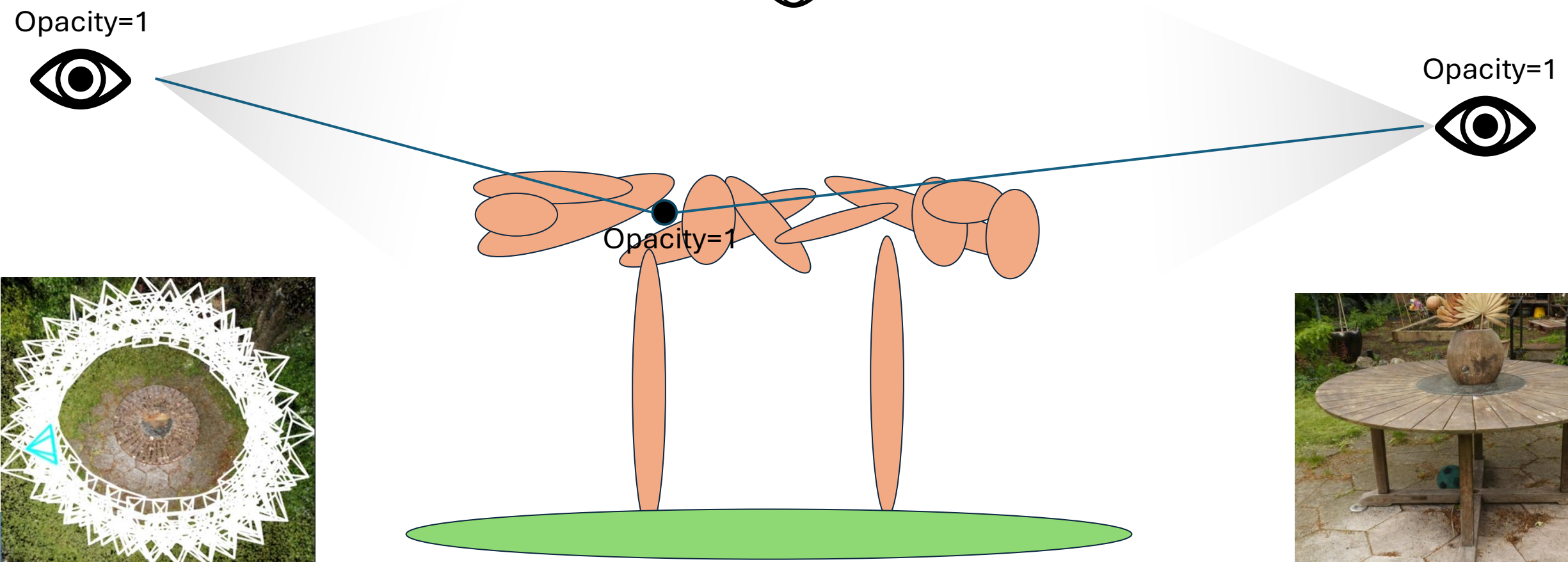
$$O(\bigcirc) = \min_{\text{eye}}( o(\bigcirc, \text{eye}) )$$




[Yu et al. 2024]

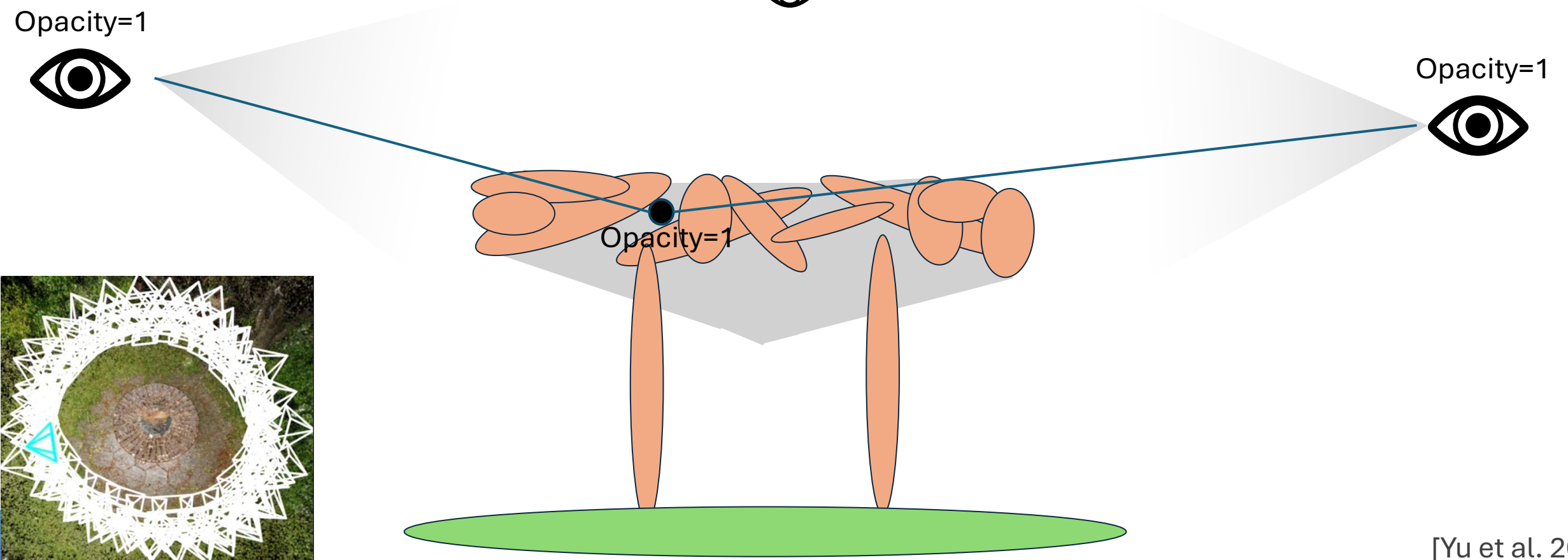
# Gaussian Opacity Fields

$$O(p) = \min_{\text{eye}}( o(p, \text{eye}) )$$



# Gaussian Opacity Fields

$$O(p) = \min_{\text{eye}}( o(p, \text{eye}) )$$

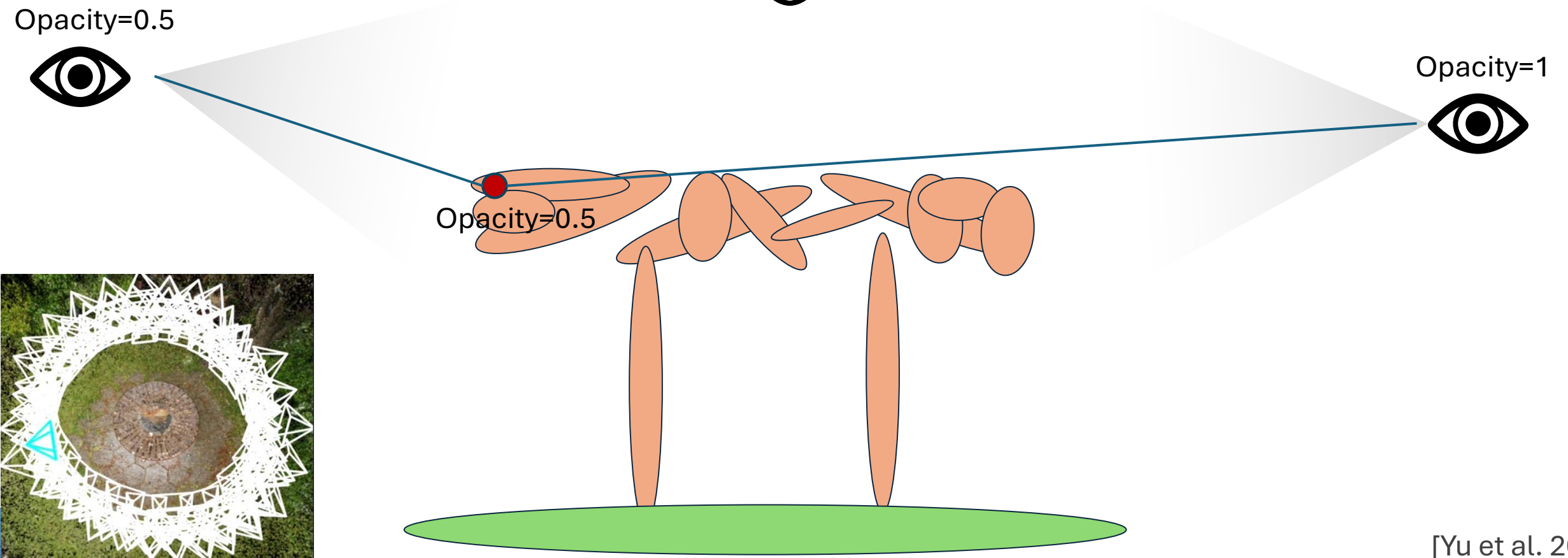





[Yu et al. 2024]

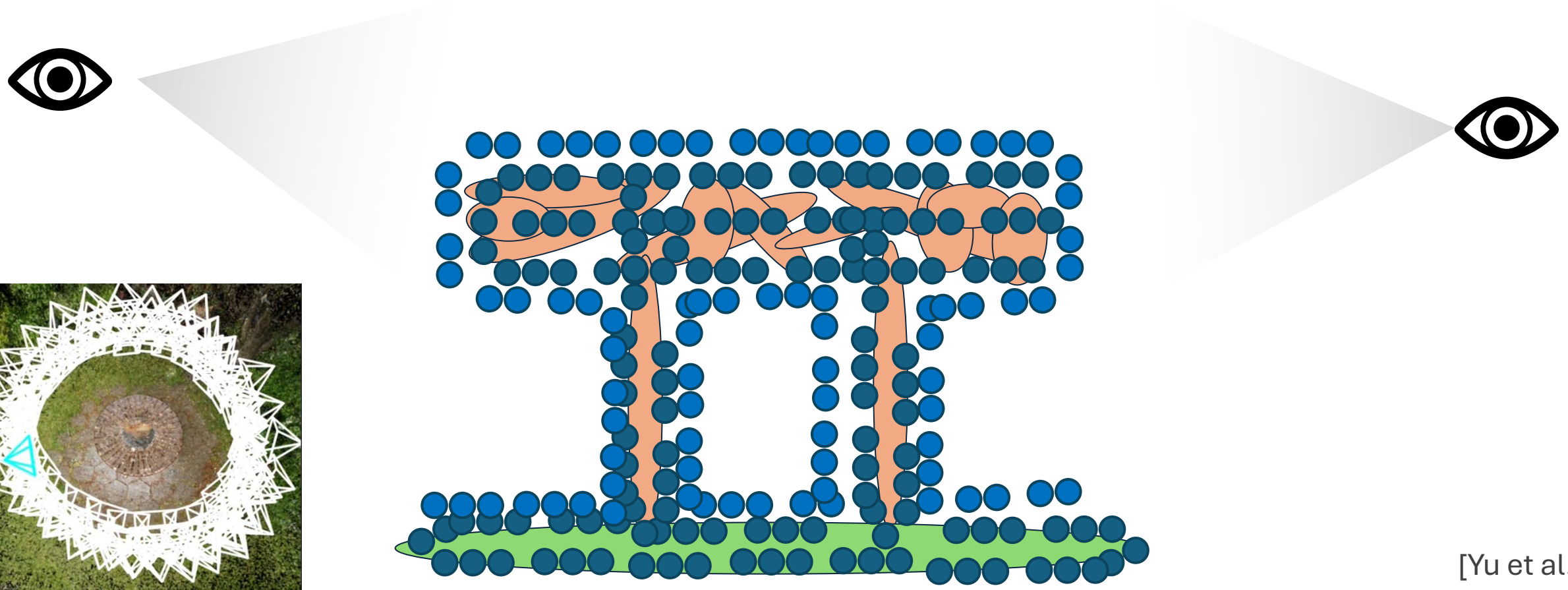
# Gaussian Opacity Fields

$$O(p) = \min_{\text{eye}}( o(p, \text{eye}) )$$



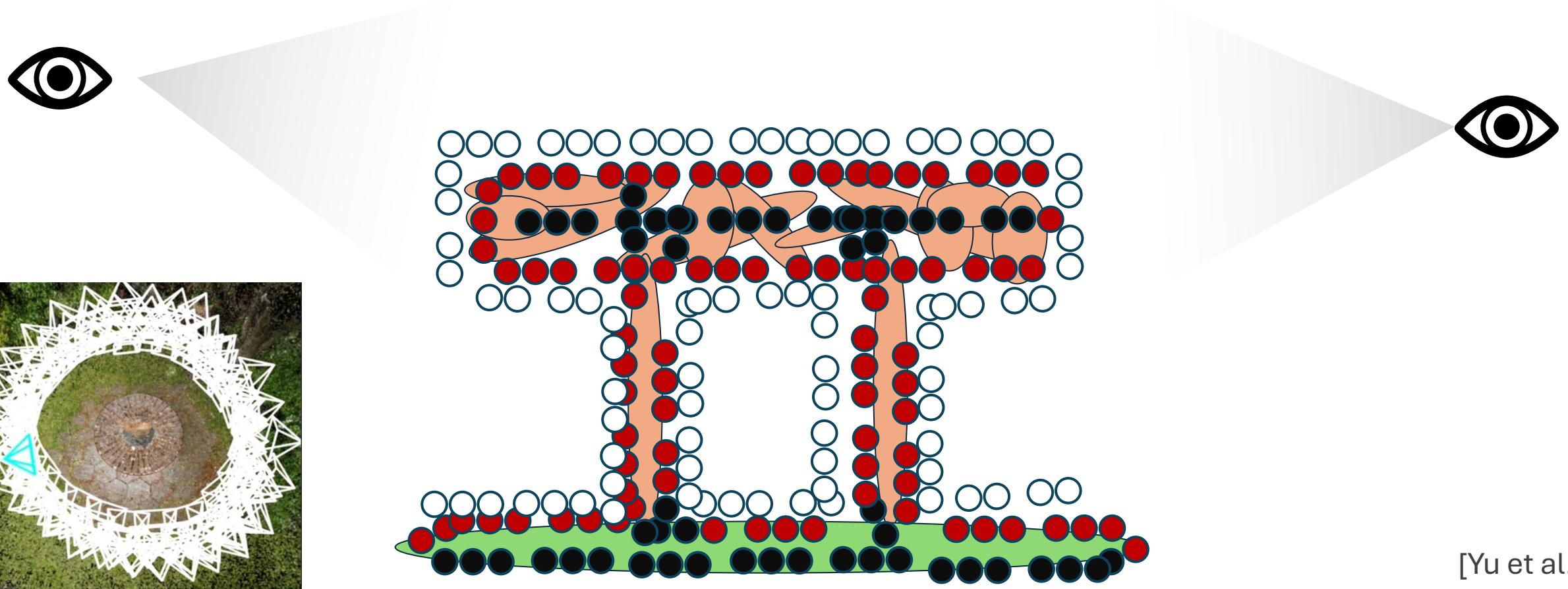

[Yu et al. 2024]

# Gaussian Opacity Fields



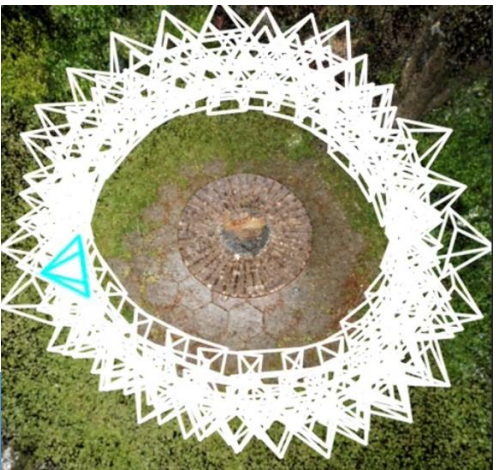
[Yu et al. 2024]

# Gaussian Opacity Fields



[Yu et al. 2024]

# Gaussian Opacity Fields



[Yu et al. 2024]

# Sorted Opacity Fields: Faster Meshing

Gaussian Opacity Fields can be slow:

$$\#samples = \#views * 9 * \#Gaussians * \#iterations$$

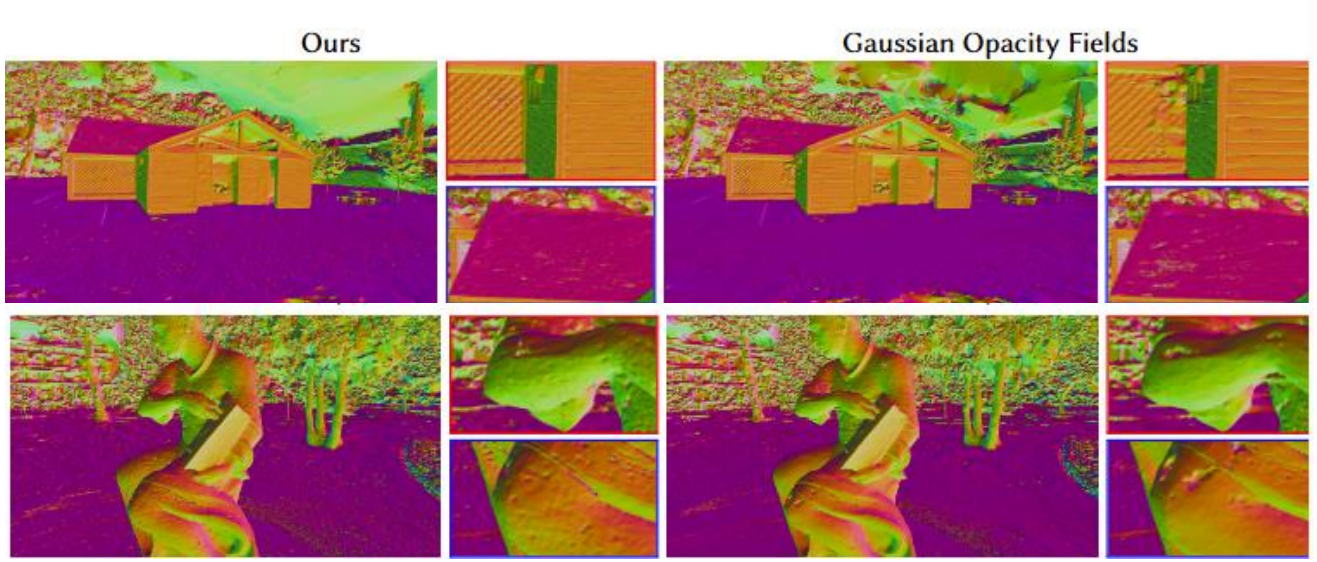
| Timings in ms | Process Points | Process Gaussians | Integrate | Total |
|---|---|---|---|---|
| GOF | 1.17 | 1.07 | 501.12 | 503.38 |
| + Tile/Ray Scheduling | 1.15 | 0.99 | 415.99 | 418.13 |
| + Min Z Bounding | 1.15 | 0.99 | 257.29 | 259.44 |
| + Early Stopping | 1.15 | 0.99 | 116.32 | 118.27 |
| + Point Pruning | 0.91 | 0.99 | 72.41 | 74.40 |

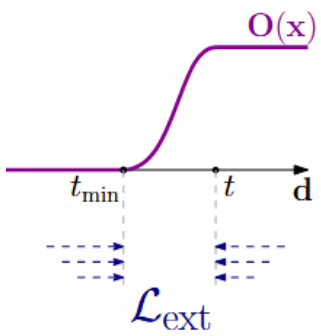
**~ 6.8x faster Meshing**



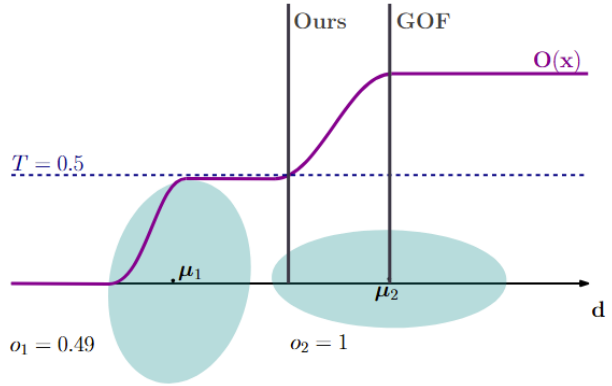
[Radl et al. 2025

# Sorted Opacity Fields: More Accuracy

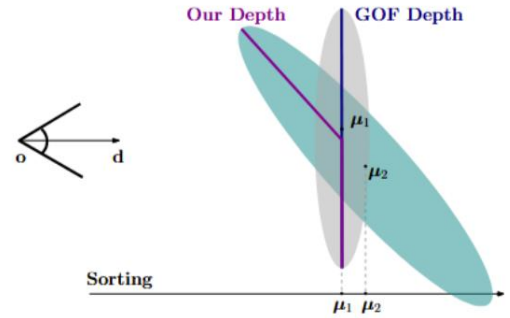



Extent Loss




Exact Median Depth




Hierarchical Sorting

[Radl et al. 2025

# Thank you


Bernhard Kerbl
kerbl@cg.tuwien.ac.at

Thomas Köhler
t.koehler@tugraz.at

Felix Windisch
felix.windisch@tugraz.at