3D Gaussian ~~Splatting~~ Volumetric Pathtracing

Euan Hughes

École de Technologie Supérieure

MTR871 Presentation
August 08, 2025

Table of Contents

Table of Contents

What is Rendering?

Rendering = “Light Transport Simulation”

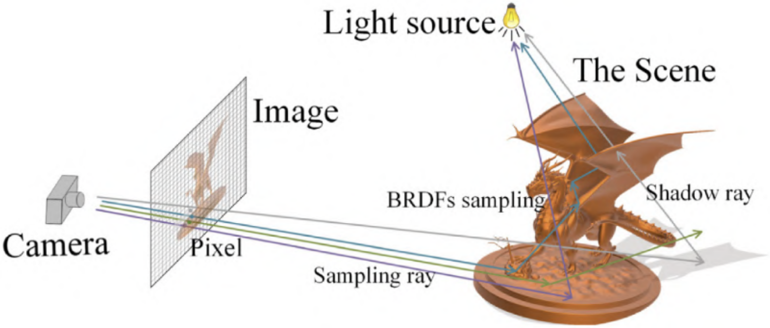


Image from Y. Li et al. 2019

Physically-Based Rendering

Simulation that adheres to the laws of physics

What is Physically-Based Volume Rendering?

Goal: accurately simulate light transport in complex volumetric media such as clouds, smoke, and fire.

Image from Britannica 2025

Image from Wikipedia 2025

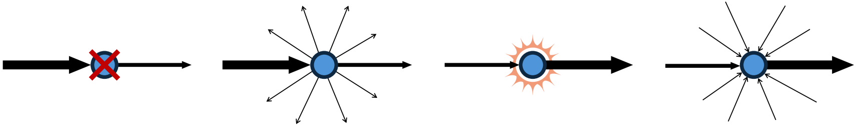
Why Physically-Based Volume Rendering?

Main Application: VFX pipelines

Image from *Up* 2009

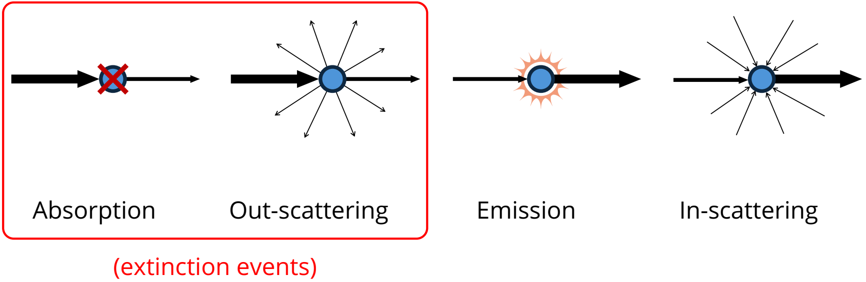
How Physically-Based Volume Rendering?

How does light change as it travels through a volume?



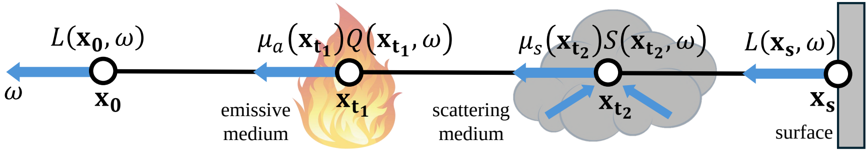
Absorption Out-scattering Emission In-scattering

How does light change as it travels through a volume?



(!!) Must integrate over all these events

Volume Rendering Equation



Transmittance

Light is continuously attenuated by extinction events. Physically described by:

$$T(\mathbf{x}_0, \mathbf{x}_t) = \exp\left(-\underbrace{\int_0^t (\mu_t(\mathbf{x}_{t'}))\,\mathrm{d}t'}_{\text{'Optical Depth'}}\right)$$

Solving the Volume Rendering Equation

Challenges

- Scattering happens in all directions
- Scattering happens constantly along a ray

(!!) Especially with multiple scattering, impossible to solve analytically

Solution: Monte Carlo Integration!!

- Randomly sample scattering directions
- Randomly sample scattering distances

Free-Flight Distance Sampling

Key Insight

Transmittance formula $T(\mathbf{x}_0, \mathbf{x}_t)$ is a *'survival function'*: describes probability of a photon not interacting up to distance t

Inverse Transform Sampling

Creates a way to sample random distances consistent with this distribution

Gets a random 'target' optical depth:

$$\int_0^t (\mu_t(\mathbf{x}_{t'}))\, \mathrm{d}t' = \tau$$

Now, solve for distance t, which depends on extinction $\mu_t(\mathbf{x})$

Free-Flight Distance Sampling

Homogeneous volumes: $\mu_t(\mathbf{x})$ is a fixed constant μ_t

$$\int_0^t \mu_t \, \mathrm{d}t' = \mu_t \, t, \quad \Longrightarrow \quad t = \frac{\tau}{\mu_t}$$

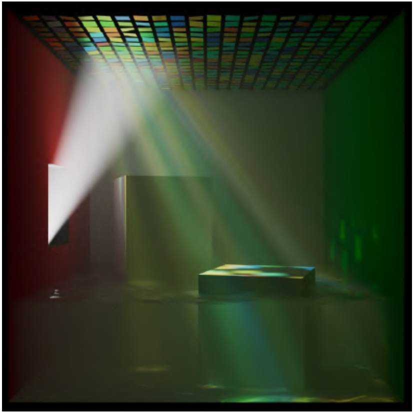


Image from @RaphaelRau 2022

Free-Flight Distance Sampling

However...

Lots of volumes are *heterogeneous*: extinction varies, cannot invert CDF

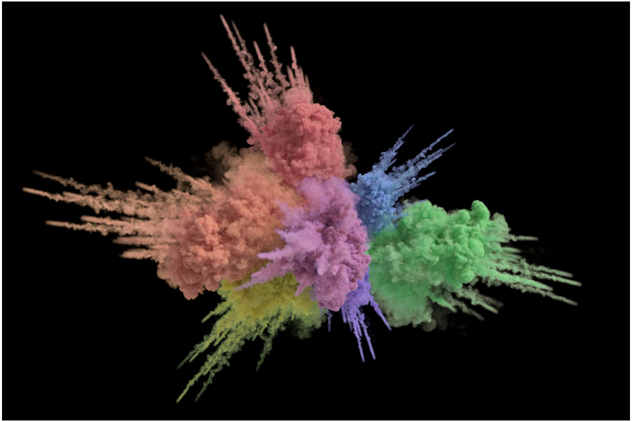
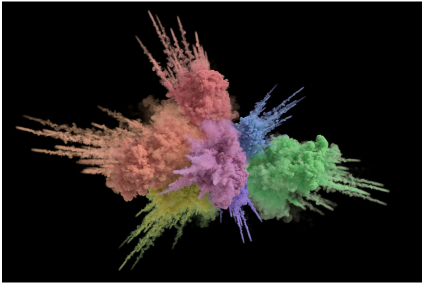


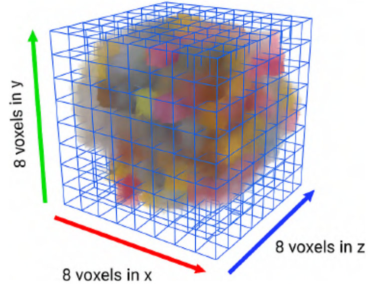
Image from Kutz et al. 2017

Regular Tracking

Solution: Voxels!

Discretize the volume into a 3D grid of piecewise homogeneous regions.





Then: trace rays through the volume, track voxel boundaries, and perform homogeneous sampling over each segment.

Regular Tracking

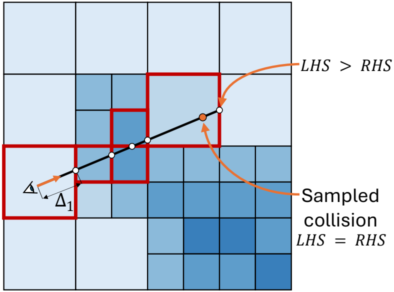
$$\boxed{\int_0^t (\mu_t(\mathbf{x}_{t'}))\, \mathrm{d}t'} = \tau$$

Equivalent to:

$$\sum_{j=1}^{k} \mu_{t,j}\, \Delta_j$$

To solve for free-flight distance t:

- Sample target optical depth: τ
- Sum segment-wise depth, in order
- Stop when running sum exceeds target
- Interpolate within the final segment



Unbiased Estimator!

Ray-Marching

Regular tracking limitation

Tracking all voxel boundaries along rays are expensive, especially in very dense grids

Alternative

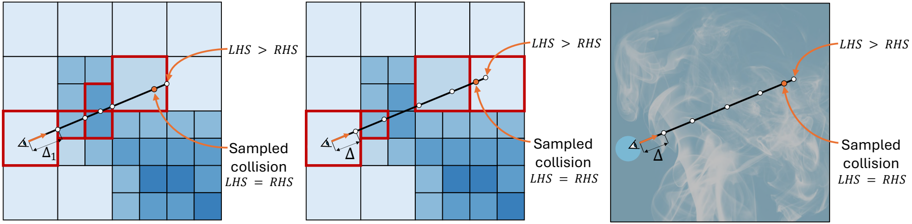
March the ray in fixed spatial increments and sample whatever voxel the step occurs in, assuming homogeneous over the step.

$$\boxed{\int_0^t \left(\mu_t(\mathbf{x}_{t'})\right) \mathrm{d}t'} = \tau$$

Approximated by:

$$\sum_{j=1}^{N} \mu_t(\mathbf{x}_{t'})\,\Delta \quad \Longrightarrow \quad \boxed{\textbf{Biased Estimator!}}$$

Ray-Marching



(a) Regular Tracking Voxels **(b)** Ray-Marching Voxels **(c)** Ray-Marching Thin Features

Representations

Voxels are state-of-the-art

Advantages

- Data can be read quickly
- Natural Level-of-Detail rendering
- Artist friendly
- Simple to integrate into pathtracers

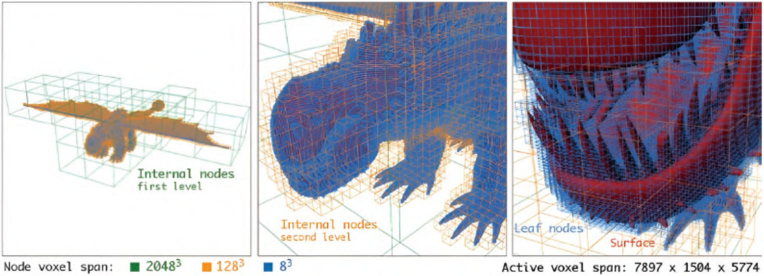
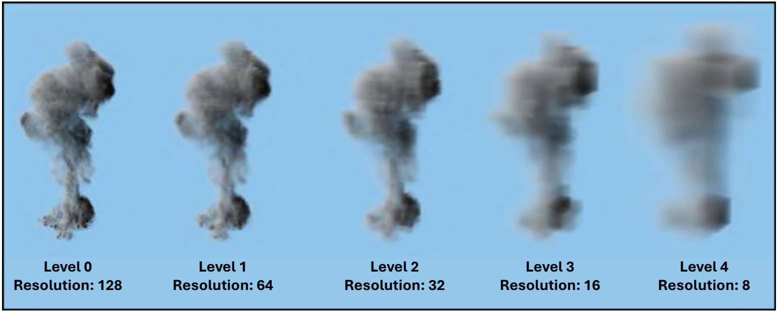


Image from Museth 2013

Representations

Voxel Disadvantages

- High memory requirements for dense grids
- Blocky artifacts at lower memory/Level-of-Detail



Alternatives to voxels?

Neural representations like NeRF have impressive compression, however:

- slow lookups
- complicated to add to pathtracers
- (!!) deviates significantly from the physical model

This motivates a new representation that:

- Adheres to the volume rendering equation
- Remains much more expressive at lower memory

Table of Contents

3D Reconstruction

Goal: infer a 3D representation of a scene from a collection of 2D images, capable of novel-view synthesis

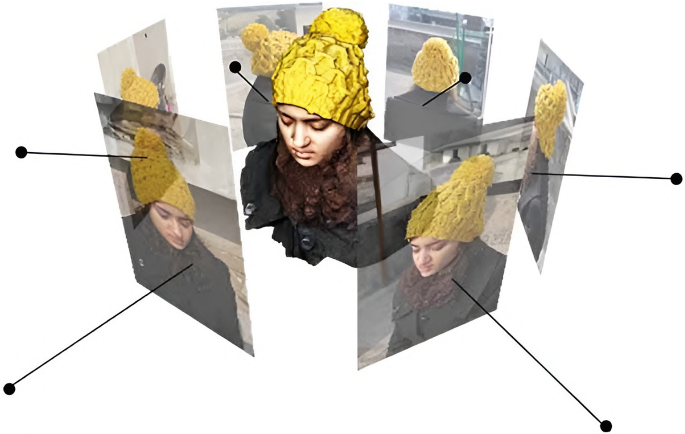
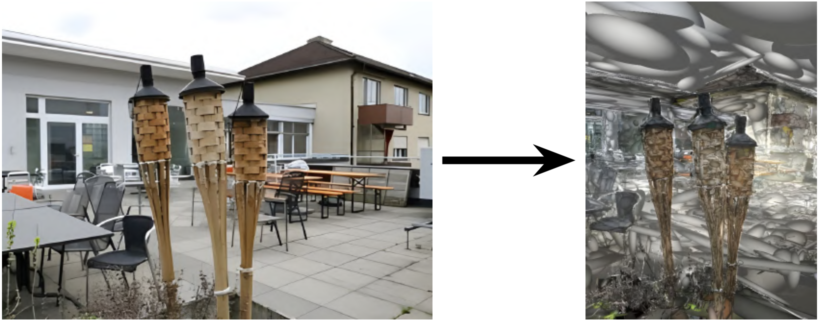


Image from Wong 2021

3D Gaussian Splatting (3DGS)

Idea:

Represent scene as collection of view-dependent Gaussians.
Optimize parameters to match images.



Images from R. Li and Cheung 2024

High fidelity and fast !

Why is 3DGS not Physically-Based?

- No light sources and scene scattering: lighting is ‘baked’ into the primitives
- “Splatting” based pipeline
  - 3D Gaussians are turned into a 2D billboard, color values are summed and alpha blended based on approximate depth ordering and a *fixed opacity*

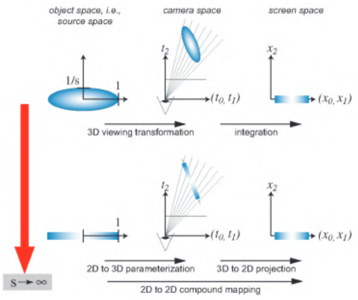


Image from Zwicker et al. 2002

Can we treat 3D Gaussians in a more physically-based manner? We will see: Yes !

The case for Physically-Based Rendering

Primarily: extremely expressive, superior to voxels at equivalent and even higher memory usage

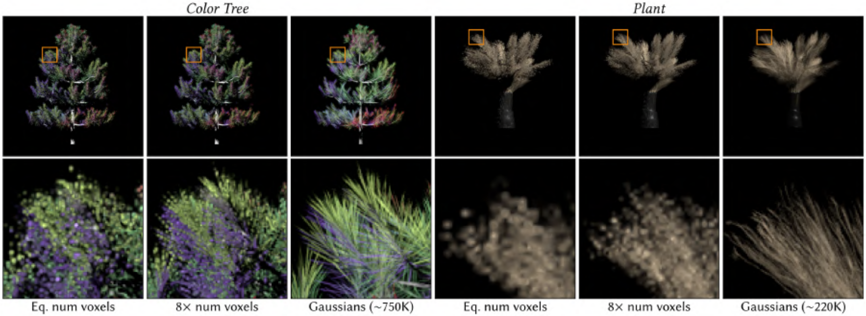


Image from Zhou, Wu, and Yan 2024

For VFX: same/higher quality renders with much lower memory footprint

The case for Physically-Based Rendering

More desirable properties:

- Gaussians have closed form integral: the Gauss error function
  $\implies$ exact solution for transmittance
- inverse of Gauss error function also has a closed form
  $\implies$ exact solution for free-flight distances

For VFX: we can get unbiased and physically accurate renders

Table of Contents

“Gaussian-Mixture Model”

Image modified from Disney 2017

Unlike 3DGS and prior methods: we have light sources, and need to consider scattering events

Gaussian-Mixture Model Formulation

Also unlike 3DGS and prior methods: perform exact integration over Gaussian densities

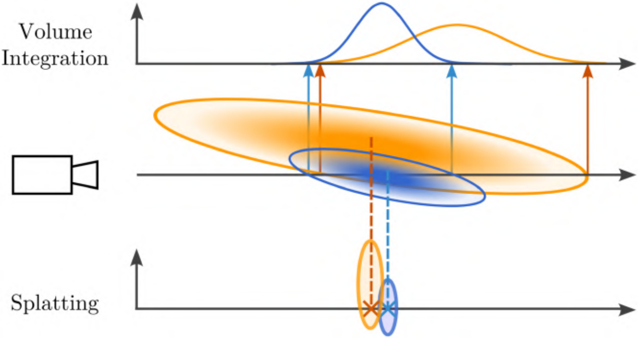


Image from Celarek et al. 2025

We need to find exact solutions to transmittance and free-flight sampling distances over these integrals

Transmittance

Extinction

$$\mu_t(\mathbf{x}) = \sum_{i=1} \sigma_i G_i(\mathbf{x})$$

In Practice: we have to find all entry/exit events $[t_{i,0}, t_{i,1}]$ to find the active range of each Gaussian along a ray (we will see how in the implementation section)

Transmittance

$$\implies \quad T(\mathbf{x}_0, \mathbf{x}_t) = \exp\Bigg(\underbrace{-\sum_{i=1} \sigma_i \int_{\max(0,t_{i,0})}^{\min(t,t_{i,1})} G_i(\mathbf{x}'_t)\,\mathrm{d}t'}_{\textit{Optical Depth}} \Bigg)$$

Transmittance

$$T(\mathbf{x}_0, \mathbf{x}_t) \;=\; \exp\left(-\sum_{i=1} \sigma_i \int_{\max(0, t_{i,0})}^{\min(t, t_{i,1})} G_i(\mathbf{x}'_t)\, \mathrm{d}t' \right)$$

Solution

Since Gaussians have a closed-form integral (erf), this has an exact solution... easy :)

Distance Sampling

... harder :(

Distance Sampling

Ray Segmentation

- Find all Gaussian entry/exit events $[t_{i,0}, t_{i,1}]$
- Sort all individual events (not pairs of events) in ascending order.
- This yields an ordered sequence of disjoint segments which the set of overlapping Gaussians is fixed.

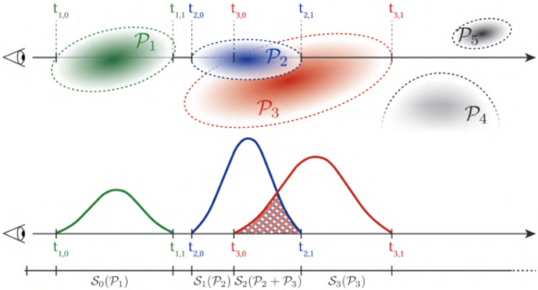


Image from Condor et al. 2025

Distance Sampling

Recall: Regular Tracking

Accumulate optical depth over piecewise homogeneous segments until exceed target depth τ)

For Gaussians: accumulate optical depth over segments with fixed numbers of Gaussians

Problem

Once we exceed the target depth, we need to solve for the exact distance within the last segment

In Regular Tracking: segments are homogeneous $\implies$ do simple linear interpolation

For Gaussians: ... how to solve?

Distance Sampling

Cases:

- **One Gaussian:** closed form solution exists with the inverse error function erf^{-1}
- **Multiple Gaussians:** impossible to invert analytically. Instead, must use numerical root finding method

With this formulation, we have exact solutions for transmittance and distance samples
$\implies$ unbiased VFX renders

Results

As desired: the implementation is capable of extremely high compression rates

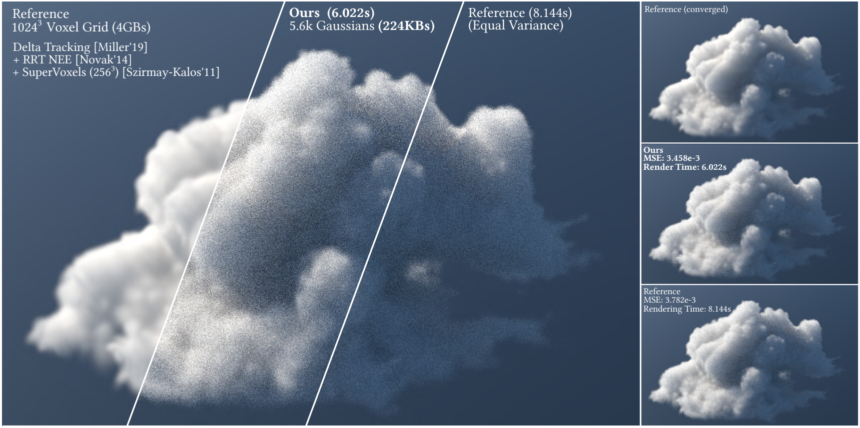


Image from Condor et al. 2025

Results

Image from Condor et al. 2025

Results

Original *cloud* (999x634x1224 effective resolution, 480MB VDB volume)

Ours, 16.2k primitives GMM (639KB)
MSE: 1.621e-4
Rendering time: 26.27mins

Ours, 5.6k primitives GMM (224KB)
MSE: 1.717e-4
Rendering time: 19.43mins

Ours, 2.6k primitives GMM (107KB)
MSE: 2.72e-4
Rendering time: 11.67mins

Image from Condor et al. 2025

Table of Contents

Building a Renderer from Scratch

Wrote full integrator from scratch

Goals

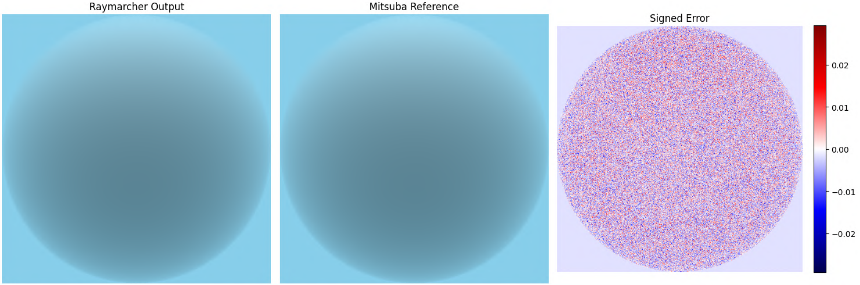
- Proper Gaussian intersection routines
- Analytical transmittance
- Free-flight sampling

Initial Version: Raymarching with Single Scattering

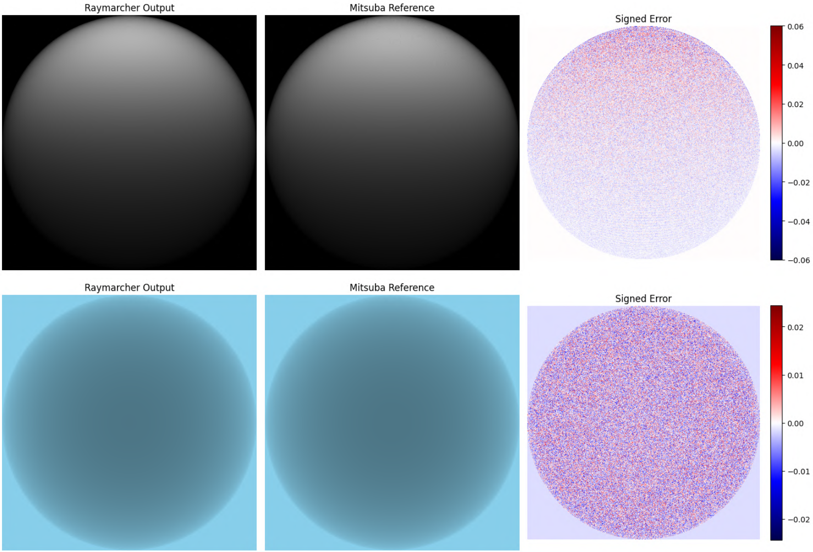
- Fixed step marching
- Compute extinction at each step $\rightarrow$ transmittance
- In-scattered radiance from lights (also marched)
- Assumed constant extinction per step

Validating the Raymarcher

- Tested on simple spheres
- Compared results with Mitsuba
- Matched reference images



Validating the Raymarcher



Representation-Agnostic Integrator

- Integrator independent of volume representation
- $\implies$ Can swap spheres $\leftrightarrow$ Gaussians
- Useful for generating reference results for Gaussians (No Mitsuba support)

Two Intersection Methods for Gaussians

Method 1: Map to unit sphere

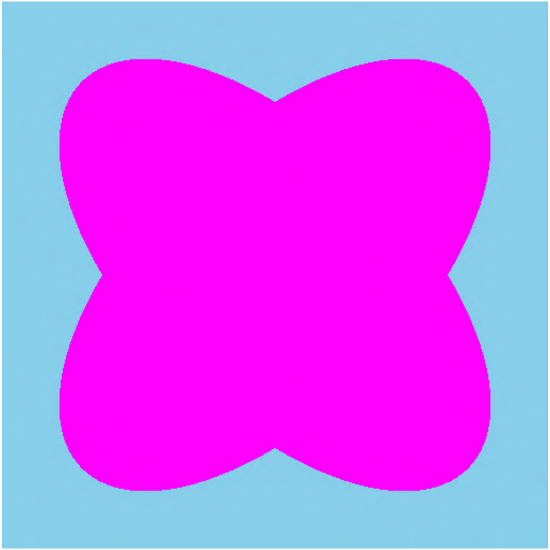
- Transform ray by whitening transform: $x' = L^{-1}(x - \mu)$
- Intersect unit sphere in transformed space
- Map intersection points back to original space

Method 2: Solve equations directly

- Treat 3D Gaussian as ellipsoid: $(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) = R^2$
- Plug ray into equation $\rightarrow$ quadratic in t
- Solve for t_0, t_1; clamp to ray bounds for entry/exit

- Direct solver is faster
- Both allow precomputation to speed up queries

Validating Intersection Routine



Extinction from Gaussians

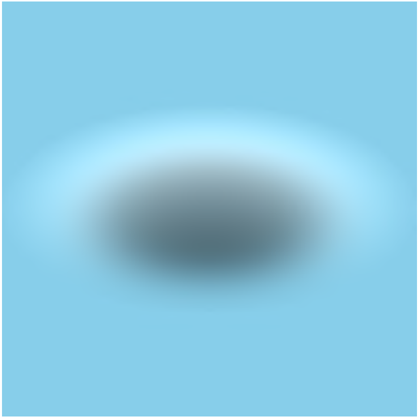
Recall

Extinction is sum over all Gaussians

$$\mu_t(\mathbf{x}) = \sum_i \sigma_i \cdot G_i(\mathbf{x})$$

Reference Renders with Gaussians

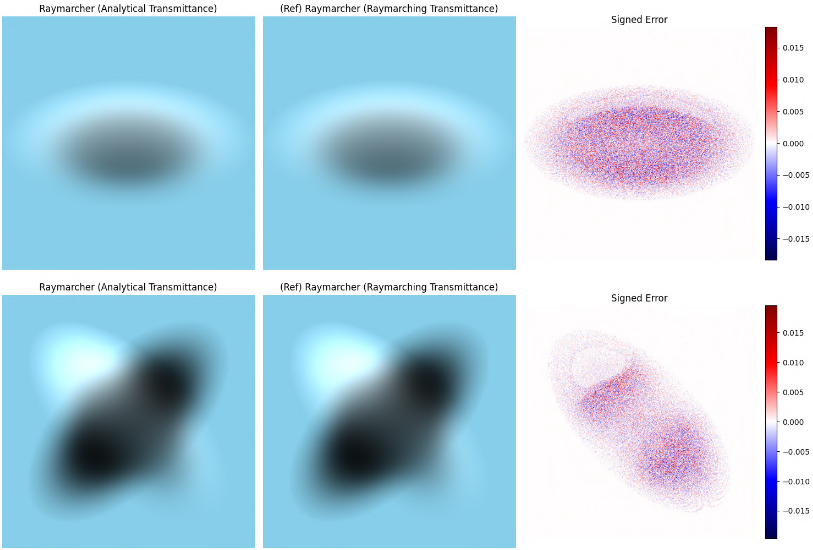
- No changes to integrator
- Only volume representation changed
- Use as reference for analytical transmittance



Adding Analytical Transmittance

- Compute exact transmittance over integral for each raymarching step
- Analytic In-Scattering
    - Find all Gaussian entry/exit points towards lights
    - Sort them
    - Compute exact transmittance to each

Validation

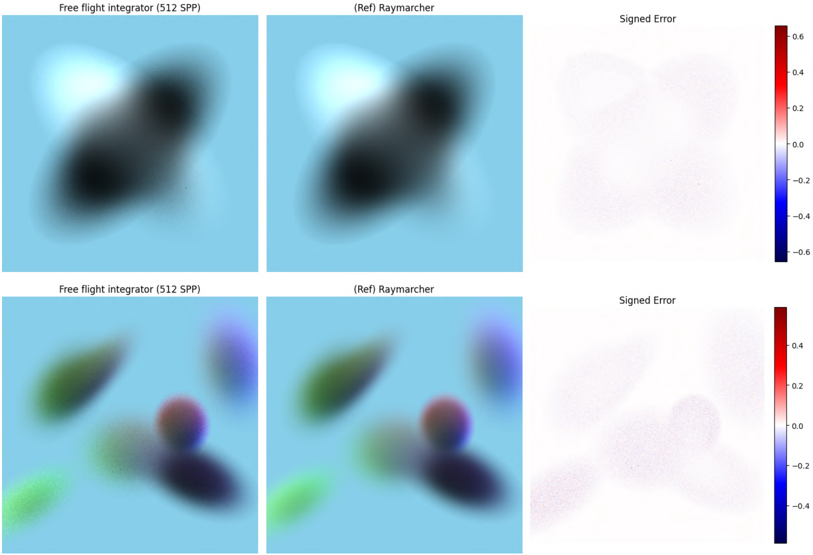

Raymarcher (Analytical Transmittance)
(Ref) Raymarcher (Raymarching Transmittance)
Signed Error
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
Raymarcher (Analytical Transmittance)
(Ref) Raymarcher (Raymarching Transmittance)
Signed Error
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015

Adding Free Flight Sampling

Distance Sampling Routine

- Segment primary ray at all intersections
- Sample target optical depth τ
- Accumulate $\int \mu_t$ across segments
- When target exceeded: solve last segment (bisection)

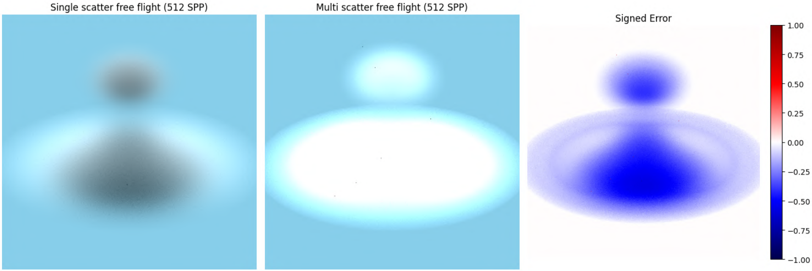
Validation:


Free flight integrator (512 SPP)
(Ref) Raymarcher
Signed Error
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Free flight integrator (512 SPP)
(Ref) Raymarcher
Signed Error
0.4
0.2
0.0
−0.2
−0.4

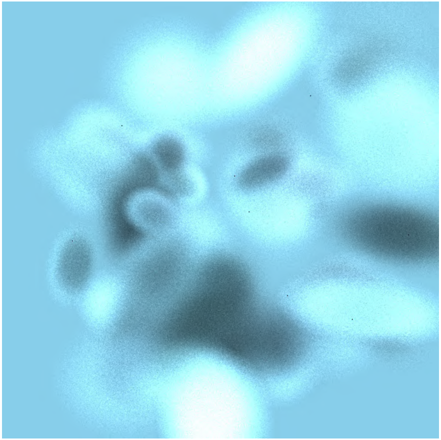
Multiple Scattering Extension

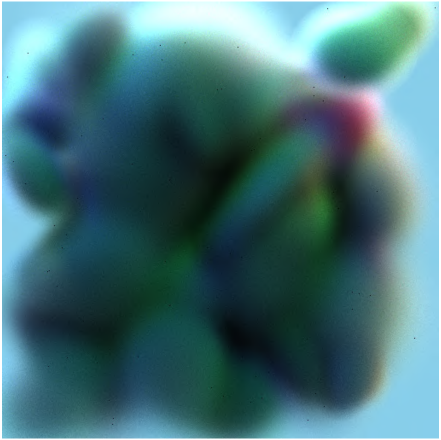
At each scattering event:

- Compute in-scattering as before
- Combine with sampling random direction and recursively walking path



Multiple Scattering Scenes





Results

Volumetric Path Tracer:

- Gaussian volume support
- Multiple scattering
- Exact transmittance
- Free-flight sampling

Next Steps:

- Acceleration structures for intersection (currently slow with many Gaussians)
- General performance optimization for large scenes

Table of Contents

Acceleration is Crucial

Key Finding

Rendering without acceleration structures becomes **prohibitively slow**.

Even with acceleration, most computation is spent on:

- Ray traversal
- Distance sampling
- Intersection tests

In a similar recent method *Unified Gaussian Primitives* (Zhou, Wu, and Yan 2024), this took up **85%** of the total rendering time.

Speeding up Distance Sampling

Distance Sampling Approximation

At the last segment, instead of solving exactly, assume homogeneity (as in raymarching).

This adds a small bias.

Delta Tracking

Adds a majorant and performs rejection sampling.

Challenge: finding a good majorant is difficult with arbitrary overlap.
Recent work on **progressive null-collision tracking** (Misso et al. 2023) allows for high quality renders with a poor majorant.

Bounding Icosphere

Wrap each Gaussian in a bounding icosphere.

This allows fast ray-triangle intersection tests.

- Significant speedup
- But increased memory usage

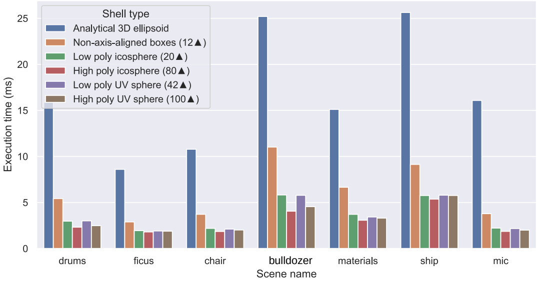


Image from Condor et al. 2025

Fighting Overlap

A major bottleneck: **high Gaussian overlap**.

Epanechnikov Kernel

Sharper falloff $\Rightarrow$ less overlap

- Closed-form transmittance + sampling still possible
- Often similar quality
- Much faster in practice

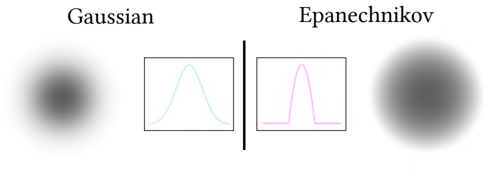


Image from Condor et al. 2025

Pruning Overlap (from LoD methods)

Idea from recent Gaussian LoD work

During optimization, prune Gaussians that overlap excessively.

- Compute average distance to *k* nearest neighbors
- Remove any Gaussian with excessive overlap

Result

Speeds up rendering & reduces memory use.

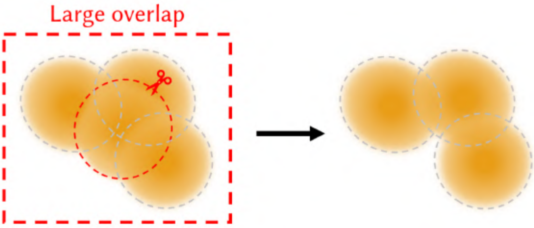


Image from Seo et al. 2025

Level-of-Detail Opportunities

Multi-Level Structures

Gaussians are very expressive $\Rightarrow$ promising for LoD!

- Can adapt to memory budgets
- Scale well across detail levels

Lots of 3DGS LoD methods are agnostic to the rendering method

But: need to modify optimization heuristics.

Scene Creation with Gaussians

Artist Pain Point

Gaussians are much harder to author than voxels.

There must be a heuristic or method to:

- Convert voxel grids $\rightarrow$ Gaussians
- Do it **accurately** or it's useless

This is a crucial pipeline stage for real production use.

Differentiable Rendering

Authors' Pipeline

Full differentiation through **stochastic multi-scattering** paths

- Much harder than 3DGS — must trace back exact random walk
- Much slower than classic 3DGS training
- But high quality and accurate

Takeaway

Differentiability is a **must-have** for practical applications.

Conclusion

Big Picture

3D Gaussians show great potential in **production volumetric pathtracing**.

- Excellent visual quality
- Orders-of-magnitude lower memory usage
- Physically-based (no approximations!)

But...

We need significant speedups in both forward and inverse rendering for widespread adoption.

Still a young area. Lots of promising ideas to explore in my own research.

References I

@RaphaelRau (May 2022). *Tweet posted on May 20, 2022*. Accessed: 2025-08-07. URL: https://x.com/OTOY/status/1527691348064669696.

Britannica (2025). *Crepuscular ray*. Accessed 7 August 2025. Encyclopaedia Britannica. URL: https://www.britannica.com/science/crepuscular-ray.

Celarek, Adam et al. (Feb. 26, 2025). *Does 3D Gaussian Splatting Need Accurate Volumetric Rendering?* DOI: 10.48550/arXiv.2502.19318. arXiv: 2502.19318[cs]. URL: http://arxiv.org/abs/2502.19318 (visited on 05/26/2025).

Condor, Jorge et al. (Feb. 28, 2025). "Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media". In: *ACM Transactions on Graphics* 44.1, pp. 1–17. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3711853. URL: https://dl.acm.org/doi/10.1145/3711853 (visited on 05/26/2025).

Disney (2017). *Walt Disney Animation Studios Cloud Data Set*. Licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. See http://creativecommons.org/licenses/by-sa/3.0/. Includes volumetric cloud model photograph by Kevin Udy from the Colorado Clouds Blog (https://coclouds.com/436/cumulus/2012-07-26/). URL: https://disneyanimation.com/resources/clouds/.

Kutz, Peter et al. (Aug. 31, 2017). "Spectral and decomposition tracking for rendering heterogeneous volumes". In: *ACM Transactions on Graphics* 36.4, pp. 1–16. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3072959.3073665. URL: https://dl.acm.org/doi/10.1145/3072959.3073665 (visited on 07/02/2025).

Li, Ruiqi and Yiu-ming Cheung (2024). "Variational multi-scale representation for estimating uncertainty in 3d gaussian splatting". In: *Advances in Neural Information Processing Systems* 37, pp. 87934–87958.

Li, Yuanhang et al. (2019). "Real-time optical 3D reconstruction based on Monte Carlo integration and recurrent CNNs denoising with the 3D light field display". In: *Optics express* 27.16, pp. 22198–22208.

References II

Misso, Zackary et al. (2023). "Progressive null-tracking for volumetric rendering". In: *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–10.

Museth, Ken (June 2013). "VDB: High-resolution sparse volumes with dynamic topology". In: *ACM Transactions on Graphics* 32.3, pp. 1–22. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/2487228.2487235. URL: https://dl.acm.org/doi/10.1145/2487228.2487235 (visited on 06/29/2025).

Seo, Yunji et al. (June 12, 2025). *FLoD: Integrating Flexible Level of Detail into 3D Gaussian Splatting for Customizable Rendering*. DOI: 10.1145/3731430. arXiv: 2408.12894[cs]. URL: http://arxiv.org/abs/2408.12894 (visited on 06/20/2025).

Up (2009). Animated feature film. United States.

Wikipedia (2025). *Deflagration*. Accessed 7 August 2025. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/w/index.php?title=Deflagration&oldid=.

Wong, Prim (Jan. 2021). *2D to 3D reconstruction*. Medium (4 min read). URL: https://medium.com/super-ai-engineer/2d-to-3d-reconstruction-2bf4f9ede53d.

Zhou, Yang, Songyin Wu, and Ling-Qi Yan (Sept. 22, 2024). *Unified Gaussian Primitives for Scene Representation and Rendering*. DOI: 10.48550/arXiv.2406.09733. arXiv: 2406.09733[cs]. URL: http://arxiv.org/abs/2406.09733 (visited on 05/26/2025).

Zwicker, Matthias et al. (2002). "EWA splatting". In: *IEEE Transactions on Visualization and Computer Graphics* 8.3, pp. 223–238.