Ray tracing Gaussians

Martin Šifrar, Matija Marolt, Žiga Lesar

University of Ljubljana, Faculty of Computer and Information Science
E-mail: ms64333@student.uni-lj.si, matija.marolt@fri.uni-lj.si, ziga.lesar@fri.uni-lj.si

Abstract

We review the theory of volumetric path tracing with special attention to Gaussians as a way to model volumetric media. Such a representation is simple to evaluate and permits closed-form expressions for both transmittance and distance sampling. We build on prior work by implementing a complete volumetric path tracer for rendering Gaussians, demonstrating that a scene represented as a sum of volumetric Gaussians can be efficiently path traced, while refining the underlying mathematical formulation.

1 Introduction

Path tracing is the core algorithm behind most state-of-the-art approaches to simulating light in a scene. At its core, it is a Monte Carlo algorithm, and noise is always present in any produced image. This is even more pronounced in volumetric path tracing, where estimated integrals are of a higher dimension and convergence is slower [1]. We go over basic principles of volumetric path tracing and the most important techniques for reducing the variance (i.e. noise) of Monte Carlo integration. In particular, we apply these ideas to the case of rendering Gaussians, where algorithm specialization is possible, as presented by Condor et al. [2]. We identify a problem in the original formulation and propose a correction.

1.1 Related Work

Gaussian representations for volumetric scenes were proposed by Condor et al. [2], inspired by the success of Gaussian splatting [3] as a compact and efficient scene representation. Unlike rasterization-based splatting, which requires sorting and suffers from artifacts such as popping and limited support for relighting or complex camera models, the path-traced approach offers a physically based alternative with fewer limitations.

2 Volumetric path tracing

Just like surface path tracing has its mathematical foundation in the rendering equation [4], there is a similar integral equation, which can be derived by integrating the *radiative transfer equation* [5]. The so-called *volumetric rendering equation* (VRE) expresses the incoming radiance as an integral of attenuated in-scattered radiance

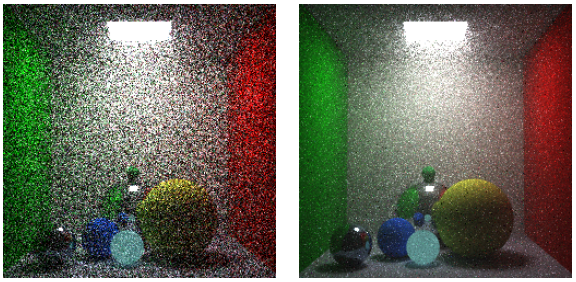

Figure 1: Integration without/with multiple importance sampling. (Left) Sampling only according to $\mathrm{PDF}_\phi(\omega')$. (Right) Multiple importance sampling.

along a ray $\mathbf{x}'(t) = \mathbf{x} + t\omega$ for a finite medium ending at $\mathbf{x}^\dagger = \mathbf{x}'(t^\dagger)$,

$$L(\mathbf{x}, \omega) = \int_0^{t^\dagger} T(\mathbf{x}, \mathbf{x}') S(\mathbf{x}', \omega)\, \mathrm{d}t + T(\mathbf{x}, \mathbf{x}^\dagger) L(\mathbf{x}^\dagger, \omega), \tag{1}$$

where $S(\mathbf{x}', \omega)$ is the in-scattered radiance. The first term sums up the attenuated in-scattering along the ray. The second term is the light transmitted through the medium from the point where the ray exits the medium. Attenuation of radiance is due to absorption and out-scattering, so we can combine their coefficients into a total attenuation coefficient $\sigma_t = \sigma_a + \sigma_s\,[m^{-1}]$. Traveling along a ray, light is attenuated by a transmittance factor

$$T(\mathbf{x}, \mathbf{x}') = \int_0^t \sigma_t(t')\, \mathrm{d}t'. \tag{2}$$

Light is added to rays by in-scattering according to $L'(t) = \sigma_s(t) S(t)$. This contribution at the point on a ray is given by the in-scattering integral

$$S(\mathbf{x}', \omega) = \sigma_s(\mathbf{x}') \int_{4\pi} \phi(\mathbf{x}', \omega, \omega') L(\mathbf{x}', \omega')\, \mathrm{d}\omega'. \tag{3}$$

Here $\phi(x', \omega, \omega')$ is the *phase function*, which encodes the directionality of the scattering process, i.e. it gives relative magnitudes of scattering from incoming direction ω' to the current ray direction ω. Scattering conserves energy, so this function is normalized $\int_{4\pi} \phi(\mathbf{x}', \omega, \omega')\, \mathrm{d}\omega' = 1$.

2.1 Monte Carlo radiance estimator

We solve the VRE integral equation by Monte Carlo (MC) integration with a recursive estimator,

$$g_L(\mathbf{x}, \omega) = \frac{T(\mathbf{x}, \mathbf{x}')}{\mathrm{PDF}(t)} \cdot g_S(\mathbf{x}', \omega) + T(\mathbf{x}, \mathbf{x}^\dagger) \cdot g_L(\mathbf{x}^\dagger, \omega). \tag{4}$$

where g_S is an estimator for the in-scattering integral,

$$g_S(\mathbf{x}', \omega) = \frac{\sigma_s(\mathbf{x}')\phi(\mathbf{x}', \omega, \omega')}{\mathrm{PDF}(\omega')} \cdot g_L(\mathbf{x}', \omega'), \tag{5}$$

which estimates Equation (3) by sampling incoming directions $\omega' \sim \mathrm{PDF}(\omega')$.

2.2 Distance sampling

To reduce variance, we use *importance sampling* [6, Ch. 2.2.2] for g_L, with a distribution $\mathrm{PDF}(t) \propto T(\mathbf{x}, \mathbf{x}')$,

$$\mathrm{PDF}(t) = \sigma_t \cdot e^{-\sigma_t t}, \quad \text{(homogeneous medium)}, \tag{6}$$

which is simple to achieve by inverse transform sampling. For the more general case of spatially varying media, distance sampling is trickier, as the desired sampling distribution is the more general

$$\mathrm{PDF}(t) = \sigma_t(t) \cdot T(\mathbf{x}, \mathbf{x}'), \quad \text{(heterogeneous medium)}, \tag{7}$$

where transmittance $T(\mathbf{x}, \mathbf{x}') = e^{-\tau(\mathbf{x}, \mathbf{x}')}$ is directly related to the *optical thickness* $\tau(\mathbf{x}, \mathbf{x}') = \int_0^t \sigma_t(\mathbf{x}')\,\mathrm{d}t$. How such samples are drawn for a prescribed $\sigma_t(\mathbf{x}')$, we will explain in Sections 2.4 and 3.2. But given samples from Equation (7), the estimator simplifies to

$$g_L = \frac{1}{\sigma_t(\mathbf{x}')} \cdot g_S(\mathbf{x}', \omega) + T(\mathbf{x}, \mathbf{x}^\dagger) \cdot g_L(\mathbf{x}^\dagger, \omega). \tag{8}$$

Such sampling produces $t \in [0, \infty)$, while the VRE integral only extends up to $t^\dagger$, so we would have to reject any $t > t^\dagger$. But since it exactly holds that $P(t > t^\dagger) = T(\mathbf{x}, \mathbf{x}^\dagger)$, we can avoid rejections by using an estimator

$$g_L = \begin{cases} \frac{1}{\sigma_t(\mathbf{x}')} \cdot g_S(\mathbf{x}', \omega) & t < t^\dagger, \text{ (ray scattered)}, \\ T(\mathbf{x}, \mathbf{x}^\dagger) \cdot g_L(\mathbf{x}^\dagger, \omega) & \text{(ray escaped the medium)}. \end{cases} \tag{9}$$

This is the estimator we use in our implementation.

2.3 Multiple importance sampling

Multiple importance sampling, introduced by Veach [1], is a way to reduce the variance of MC estimators by combining different sampling methods. In our case, we combine phase-function sampling $\omega' \sim \mathrm{PDF}_\phi(\omega') = \phi(\omega')$ and lights sampling $\omega' \sim \mathrm{PDF}_\ell(\omega')$, where $\mathrm{PDF}_\ell(\omega')$ is the uniform probability density over the directions in which there are lights present. The estimation then proceeds as follows. At each step, a sample is drawn from each distribution, while the estimators are combined into a new, improved multiple importance estimator

$$g_S^{\mathrm{MIS}} = w_\ell^{\mathrm{MIS}}(\omega'_\ell) \cdot \frac{\phi(x', \omega, \omega'_\ell)}{\mathrm{PDF}_\ell(\omega'_\ell)} \cdot L(\mathbf{x}', \omega'_\ell) + w_\phi^{\mathrm{MIS}}(\omega'_\phi) \cdot \frac{\phi(x', \omega, \omega'_\phi)}{\mathrm{PDF}_\phi(\omega'_\phi)} \cdot L(\mathbf{x}', \omega'_\phi), \tag{10}$$

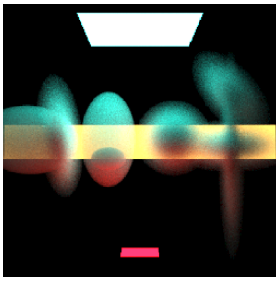

Figure 2: A test render composed of multiple overlapping Gaussians and homogeneous ellipsoids.

where w_ℓ^{MIS} and w_ϕ^{MIS} are the appropriate weights, calculated according to a balance heuristic [1]. Implementing multiple importance sampling gave a large improvement in the quality of the renders (see Figure 1 for comparison).

2.4 Delta tracking

In Section 2.2, we simplified the estimator, assuming samples are drawn from distribution in Equation (7). An obvious, but highly bias-prone way to produce such samples is inverse transform sampling via *ray-marching* [6, ch. 11.2]. An unbiased alternative is the *delta tracking* algorithm [5]. The medium is imagined to be a mixture of real and null particles (vacuum), with the dense material having a constant scattering coefficient, i.e. the majorant

$$\overline{\sigma} = \max\left\{\sigma_t(\mathbf{x}')\right\}, \quad t \in [0, t^\dagger].$$

Samples are then drawn according to $\mathrm{PDF}(t) = \overline{\sigma} \cdot e^{-\overline{\sigma}t}$, as if the medium were composed entirely of real particles. After such a sample is drawn, we draw another random number to determine if a particle is real or a null particle. If a particle is a null particle, we simulate a so-called null-scattering event, where light continues on a straight path like in a vacuum. The probability that a particle at t is real is $P_{\mathrm{real}} = \frac{\sigma_t(\mathbf{x}')}{\overline{\sigma}}$. It is best to simply write down the pseudo code (see Algorithm 1).

Algorithm 1 Delta tracking for a $t \sim \sigma_t(t)T(\mathbf{x}, \mathbf{x}')$

```
function SAMPLE_DISTANCE(x, ω, σ̄)
    t = 0
    while true do
        Draw ξ1, ξ2 ~ U(0, 1)
        t ← t − log(ξ1)/σ̄
        if ξ2 < σt(x')/σ̄ then
            break
    return t
```

3 Gaussian scene representation

We turn our attention to the special case of a medium modeled as a sum of 3D Gaussians, introduced by Kerbl et al. [3] and later adapted for path tracing by Condor et al. [2]. Here, spatially varying coefficients are represented as a sum of Gaussians,

$$\sigma_t(\mathbf{x}) = \sum_{i=1}^{N} \sigma_t^{(i)} G_{\mu,\Sigma}(\mathbf{x}), \tag{11}$$

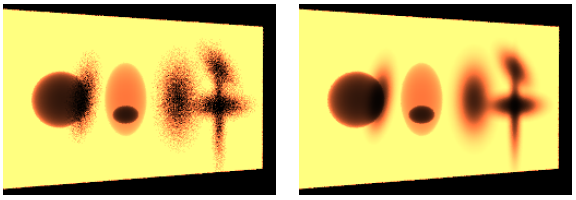

Figure 3: Transmittance of multiple overlapping Gaussians and homogeneous ellipsoids. (Left) The delta tracking transmittance estimator g_T for Gaussians. (Right) Closed-form transmittance for the Gaussians.

where $\sigma_t^{(i)}$ is the Gaussian's weight, μ is its center, and Σ is the Gaussian's covariance matrix. The basic form of the Gaussian is given as

$$G_{\mu,\Sigma}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{3}{2}}(|\Sigma|)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)}, \quad (12)$$

where $|\Sigma|^{\frac{1}{2}}$ means $\sqrt{\det(\Sigma)} = S_x S_y S_z$, with S_x, S_y and S_z being the eigenvalues of the covariance. The covariance matrix Σ is symmetric and can be decomposed as

$$\Sigma^{-1} = QM^{-2}Q^T = Q \begin{pmatrix} S_x^{-2} & 0 & 0 \\ 0 & S_y^{-2} & 0 \\ 0 & 0 & S_z^{-2} \end{pmatrix} Q^T,$$

where S_x, S_y, S_z are the $\pm 1\sigma$ radii in the Gaussian local-space. In general, the Gaussian is not axis-aligned; its rotation in world-space is given by a rotation matrix $Q \in \mathrm{SO}(3)$ or equivalently a unit quaternion $q \simeq Q$.

3.1 Closed-form transmittance

At each step of using the recursive estimator, we must propagate the radiance through media, which means multiplying it with the transmittance T. If there are multiple media boundaries, we accumulate the appropriate partial transmittances. In the simple case of homogeneous media, partial transmittances are of the form $e^{-\sigma_t^{(i)} t^{\dagger(i)}}$. This accumulation continues until we reach an opaque medium (then $T = 0$) or reach a light and add the contribution $T \cdot L_{\text{light}}$. In spatially varying media, transmittance can be estimated with a binary estimator and samples from distribution in Equation (7), e.g., with delta tracking

$$g_T = \begin{cases} 0, & t > t^\dagger \text{ (scattered)}, \\ 1, & \text{(escaped)}. \end{cases} \quad (13)$$

This approach works in general, but for the case of Gaussian primitives, we can obtain closed-form expressions of transmittance or, equivalently, the optical thickness. Condor et al. [2] derive an expression for the optical thickness along a ray through a Gaussian primitive. Based on dimensional analysis, we believe the expressions in the paper contain minor errors, and we present the corrected forms below. The optical thickness of a single Gaussian along a ray $\mathbf{x}' = \mathbf{x} + t\omega$, specified in the Gaussian's local space, is

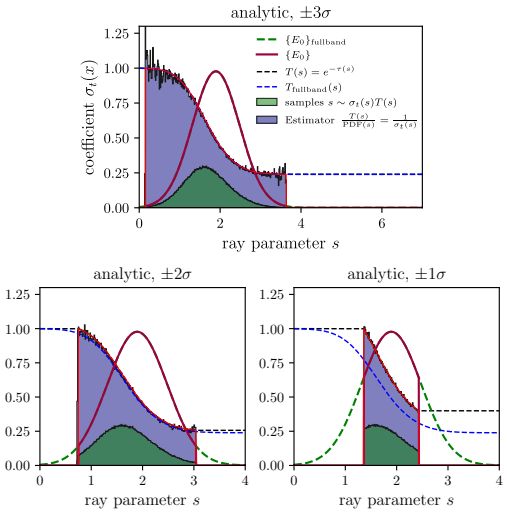

Figure 4: Analytic distance sampling for a Gaussian with different cutoffs. (Top) Gaussian limited to a $\pm 3\sigma$ ellipsoid. (Bottom) Gaussians limited to $\pm 2\sigma$ and $\pm 2\sigma$ ellipsoids.

$$\begin{aligned} \tau_i(\mathbf{x}, \mathbf{x}') &= \sigma_t^{(i)} \int_0^t G_{\mu,\Sigma}\, \mathrm{d}t \\ &= \frac{\sigma_t^{(i)} e^{-C_1}}{4\pi\sqrt{C_0}} \left(\operatorname{erf}\left(\frac{t \cdot C_0 + C_2}{(2C_0|\Sigma|)^{\frac{1}{2}}} \right) - \operatorname{erf}\left(\frac{C_2}{(2C_0|\Sigma|)^{\frac{1}{2}}} \right) \right) \end{aligned} \quad (14)$$

where C_0, C_1 and C_2 are constants depending on the starting position of the ray $\mathbf{x}$ and direction ω. Condor et al. [2] give these values as

$$\begin{aligned} C_0 &= S_x^2 S_y^2 \omega_z^2 + S_z^2 S_x^2 \omega_y^2 + S_y^2 S_z^2 \omega_x^2, \\ C_1 &= \frac{C_3 + C_4}{2C_0}, \\ C_2 &= x_z S_x^2 S_y^2 \omega_z + x_y S_z^2 S_x^2 \omega_y + x_x S_y^2 S_z^2 \omega_x, \\ C_3 &= \left(x_x^2 S_y^2 + x_y^2 S_z^2 + x_z^2 S_x^2 \right) \omega_z^2 - 2x_z\omega_z \left(x_y S_x^2 \omega_y + x_x S_y^2 \omega_x \right), \\ C_4 &= \omega_y^2 \left(x_x^2 S_z^2 + x_z^2 S_x^2 \right) - 2x_x x_z \omega_x \omega_y S_z^2 + \omega_x^2 \left(x_y^2 S_z^2 + x_z^2 S_y^2 \right). \end{aligned} \quad (15)$$

With a closed-form expression for transmittance, we avoid drawing random samples when evaluating direct lighting. In overlaps, we multiply the transmittances of the participating media $T = T^{(1)}T^{(2)} \ldots T^{(n_{\text{overlap}})}$. In Figure 3, we see that this transmittance matches the one computed with the delta tracking estimator g_T, just without the noise.

3.2 Closed-form distance sampling

Equation (14) can also be used to sample distance samples according to $\mathrm{PDF}(t) = \sigma_t(t)T(\mathbf{x}, \mathbf{x}')$. In a manner similar to homogeneous media, we pick a random $\xi \sim U(0, 1)$ and then invert the expression for optical thickness so that $\tau = -\log \xi$. We follow the paper [2] and assume that the start point of the ray $\mathbf{x}$ is well outside the Gaussian. Then the second error function term in Equation (14) becomes $\operatorname{erf}(-\infty) = -1$, and we can write

$$-1 - \frac{4\pi\sqrt{C_0}}{\sigma_t^{(i)} e^{-C_1}} (\log \xi) = \operatorname{erf}\left(\frac{t \cdot C_0 + C_2}{(2C_0|\Sigma|)^{\frac{1}{2}}} \right), \quad (16)$$

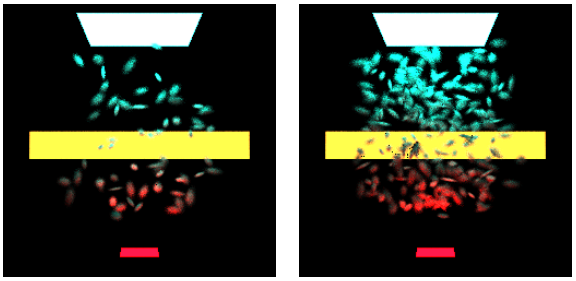

Figure 5: Rendering with closed-form distance sampling of 64 (left) and 256 (right) randomly placed Gaussians.

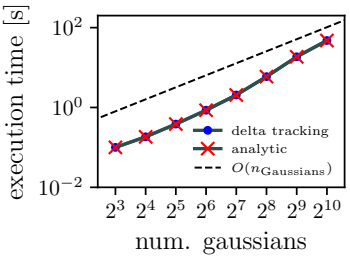

Figure 6: Execution times by the number of Gaussians. Render time is dominated by casting of rays, it scales linearly with the number of Gaussians and is almost identical for delta tracking (in blue) and closed-form distance sampling (in red).

which can be inverted by simply applying the inverse error function erf^{-1}. In Figure 4, we can see such closed-form sampling for a single Gaussian, with different cutoffs.

4 Results

We implemented all sampling strategies described in the paper in C. Scenes with randomly placed Gaussians (Figure 5) show that render time scales linearly with the number of Gaussians (Figure 6). Timing is dominated by ray casting and remains similar across both sampling methods. To achieve sub-linear scaling, Condor et al. [2] use GPU hardware-backed acceleration structures. Our implementation currently lacks such acceleration structures.

Path tracing is embarrassingly parallel, enabling efficient multithreading. Our CPU implementation splits the image across threads, but naive pixel-index partitioning causes some threads to finish much sooner than others. To prevent this imbalance, we divided the image into 64×64 tiles and randomly shuffled the tiles between the threads (Algorithm 2). This gives each thread a mix of easy and hard work and improves thread utilization, as shown in Figure 7.

Multithreaded performance suffered when using `rand` from `libc`, likely due to shared state (Figure 7). Switching to a `PCG32` generator [7] and using thread-local state fixed this issue and improved statistical properties.

Testing revealed issues with medium tracking, especially inside thin geometry. Due to floating-point errors, rays can miss exit intersections, causing unbounded media. We attempted to fix this by ignoring closely spaced intersections (closer than 10^{-5}), but this fails for oblique surfaces, leading to leaks. Rendering typical Gaussian

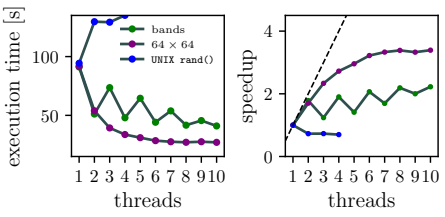

Figure 7: Timings (left) and relative speedup (right) for the Gaussian test scene with 500 samples per pixel and different numbers of threads. (In green) Naive partitioning of the image into bands over a linear pixel-index. In purple, division into 64×64 randomly shuffled tiles. In blue, division into tiles, but using `rand()` from `libc` instead of `PCG32`.

Algorithm 2 Shuffled tiles dispatch logic for one thread.

Initialize `tile_shuffle` as a random permutation of integers from 0 to $64 \times 64 - 1$
for $n = 1$ to N_{samples} **do**
  for $n = n_{\text{first tile}}$ to $n_{\text{last tile}}$ **do**
    $\tilde{n} \leftarrow$ `tile_shuffle`$[n]$
    $u, v \leftarrow \tilde{n} \bmod 64,\ \lfloor \tilde{n}/64 \rfloor$
    $x_{\text{start}},\ x_{\text{end}} \leftarrow \lfloor Wu/64 \rfloor,\ \lfloor W(u+1)/64 \rfloor$
    $y_{\text{start}},\ y_{\text{end}} \leftarrow \lfloor Hv/64 \rfloor,\ \lfloor W(v+1)/64 \rfloor$
    for $x = x_{\text{start}}$ to $x_{\text{end}} - 1$ **do**
      for $y = y_{\text{start}}$ to $y_{\text{end}} - 1$ **do**
        `render_pixel`(x, y)

splatting datasets (in the order of 10^5 Gaussians) in minutes would require a 100× speedup, ideally via GPU integration using hardware-accelerated ray casting. Future work would involve adding appropriate acceleration structures and possibly porting the implementation to the GPU.

5 Conclusion

We presented a volumetric path tracing formulation for rendering Gaussians, noted an error and suggested a corrected form of equation (14) from Condor et al. [2]. Future work includes fixing numerical issues, adding acceleration structures and improving parallelization using a GPU.

References

[1] E. Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[2] J. Condor et al. "Don't Splat your Gaussians". In: *ACM TOG* (2025).

[3] B. Kerbl et al. "3d gaussian splatting for real-time radiance field rendering." In: *ACM TOG* (2023).

[4] J. T. Kajiya. "The rendering equation". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 1986.

[5] J. Novák et al. "Monte Carlo methods for volumetric light transport simulation". In: *Computer graphics forum*. 2018.

[6] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering*. 4th ed. MIT Press, 2023.

[7] M. E. O'Neill. "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation". In: *ACM TOMS* (2014).