Tutorial 12

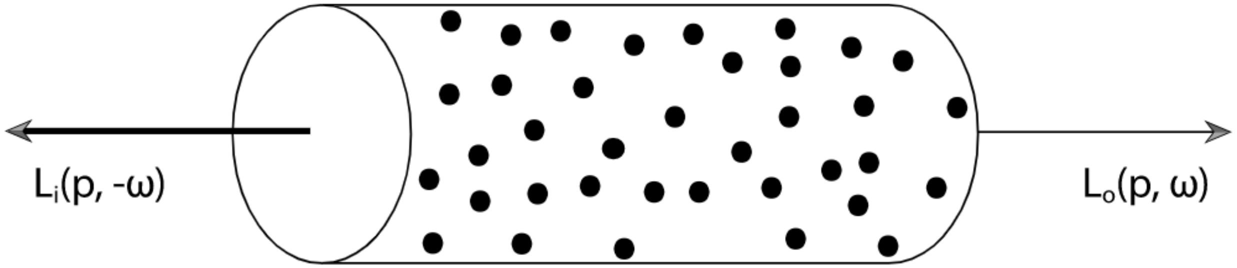
Hongtu Xu

Optical property

- Density per unit volume ρ (m^{-3})
- Cross section area σ_s and σ_a (m^2) for scattering and absorbing particles.
- Absorption coefficient $\mu_a = \rho\sigma_a$ (m^{-1})
- Scattering coefficient $\mu_s = \rho\sigma_s$ (m^{-1})
- Extinction coefficient $\mu_t = \mu_s + \mu_a$
- Represent the probability density of absorption or scattering happens per unit distance.

Volume Scattering - Absorption

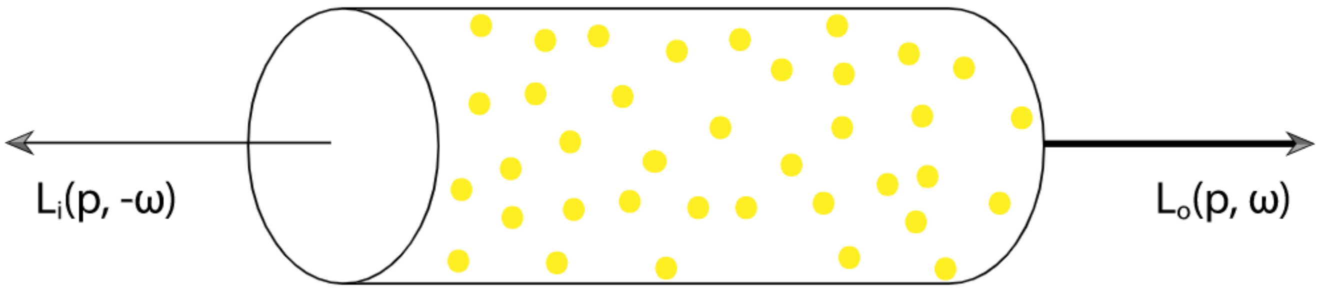
$$L_o(p, \omega) - L_i(p, -\omega) = dL_o(p, \omega) = -\mu_a(p, \omega)\, L_i(p, -\omega)\, dt$$



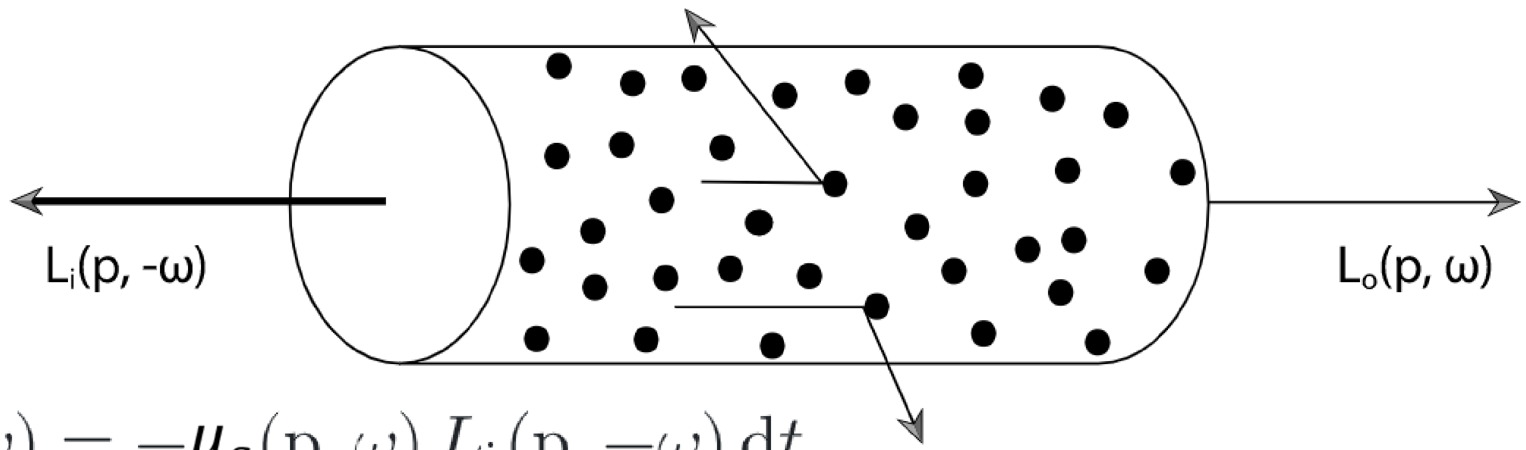
$$e^{-\int_0^d \mu_a(p+t\omega,\omega)dt}$$

Volume Scattering - Emission

$$\mathrm{d}L_{\mathrm{o}}(\mathrm{p}, \omega) = L_{\mathrm{e}}(\mathrm{p}, \omega)\,\mathrm{d}t$$



Volume Scattering - Out-Scattering and Attenuation



$$\mathrm{d}L_\mathrm{o}(\mathrm{p},\omega) = -\mu_s(\mathrm{p},\omega)\, L_\mathrm{i}(\mathrm{p},-\omega)\,\mathrm{d}t$$

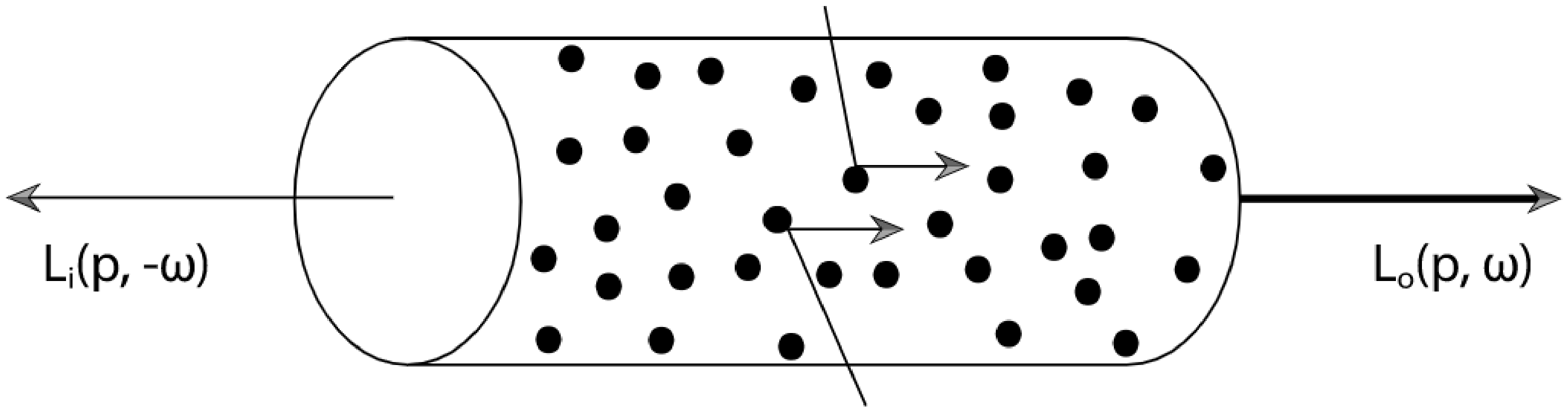
$$\mu_t(\mathrm{p},\omega) = \mu_a(\mathrm{p},\omega) + \mu_s(\mathrm{p},\omega)$$

$$\rho = \frac{\mu_s}{\mu_t}$$

quantifies the fraction of photons that undergo a scattering (vs. absorption) event at a collision, and it is commonly referred to as the single scattering albedo

$$T_r(\mathrm{p} \to \mathrm{p}') = \mathrm{e}^{-\int_0^d \mu_t(\mathrm{p}+t\omega,\omega)\,\mathrm{d}t}$$

Volume Scattering – In scattering



$$\boldsymbol{\mu_s}(\mathrm{p}, \omega) \int_{\mathrm{S}^2} p(\mathrm{p}, \omega_\mathrm{i}, \omega)\, L_\mathrm{i}(\mathrm{p}, \omega_\mathrm{i})\, \mathrm{d}\omega_\mathrm{i}$$

$$L_\mathrm{s}(\mathrm{p}, \omega) = L_\mathrm{e}(\mathrm{p}, \omega) + \boldsymbol{\mu_s}(\mathrm{p}, \omega) \int_{\mathrm{S}^2} p(\mathrm{p}, \omega_\mathrm{i}, \omega)\, L_\mathrm{i}(\mathrm{p}, \omega_\mathrm{i})\, \mathrm{d}\omega_\mathrm{i}$$

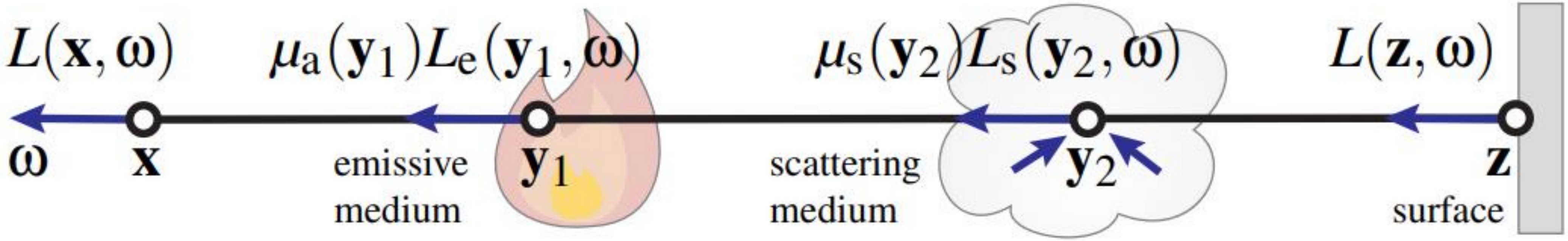
Phase Function - HenyeyGreenstein

- Similar to BRDF
- Describe scattering of volume
- A widely used phase function was developed by Henyey and Greenstein
- Easy to fit to measured scattering data
- A single parameter g (called the asymmetry parameter) controls the distribution of scattered light

$$p_{\mathrm{HG}}(\cos\theta) = \frac{1}{4\pi}\frac{1-g^2}{(1+g^2+2g(\cos\theta))^{3/2}}$$

Volume rendering equation

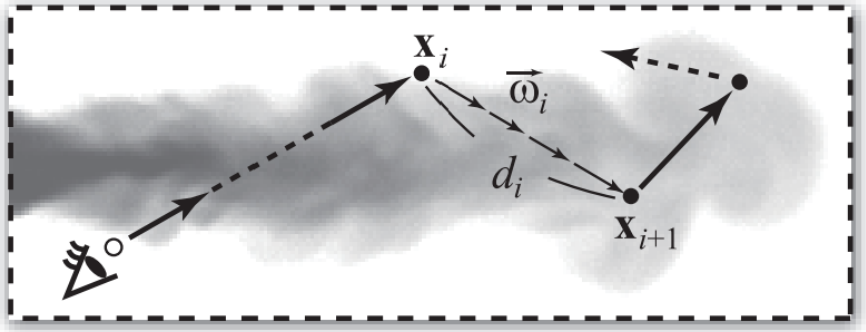
$$L(\mathbf{x},\omega) = \int_0^z T(\mathbf{x},\mathbf{y}) \left[\mu_a(\mathbf{y}) L_e(\mathbf{y},\omega) + \mu_s(\mathbf{y}) L_s(\mathbf{y},\omega)\right] \mathrm{d}y$$
$$+ T(\mathbf{x},\mathbf{z}) L(\mathbf{z},\omega). \tag{6}$$



Two key term to solve

- Transmittance $Tr(p \to p')$
- The total contribution through the light path

$$\int_0^z T(\mathbf{x}, \mathbf{y}) \left[\mu_\mathrm{a}(\mathbf{y}) L_\mathrm{e}(\mathbf{y}, \omega) + \mu_\mathrm{s}(\mathbf{y}) L_\mathrm{s}(\mathbf{y}, \omega) \right] \mathrm{d}y$$



How to sample a light path?

$$\int_0^z T(\mathbf{x},\mathbf{y})\left[\mu_\mathrm{a}(\mathbf{y})L_\mathrm{e}(\mathbf{y},\omega)+\mu_\mathrm{s}(\mathbf{y})L_\mathrm{s}(\mathbf{y},\omega)\right]\mathrm{d}y$$

$$\frac{T(x,y)[\mu_a L_e(y,\omega) + \mu_s L_s(y,\omega)]}{p(t)}$$

It is wise to sample point in media based on pdf proportion to Tr, which cancels out Tr and pdf.

How ever generate distance t proportion to Tr?

We can use a rejection based method.

Acceptance-Rejection Method

- We want to sample $f(x)$, but $f(x)$ is hard to sample
- We can easily sample a auxiliary function $g(x)$
- And $g(x)$ satisfies $cg(x) \geq f(x)$, c is a constant

- Get a sample y from g, and a sample ξ from $U(0,1)$
- If $\xi \leq \frac{f(y)}{cg(y)}$, then let $x = y$ be a valid sample
- Otherwise, we continue sampling y from g until we get a valid x

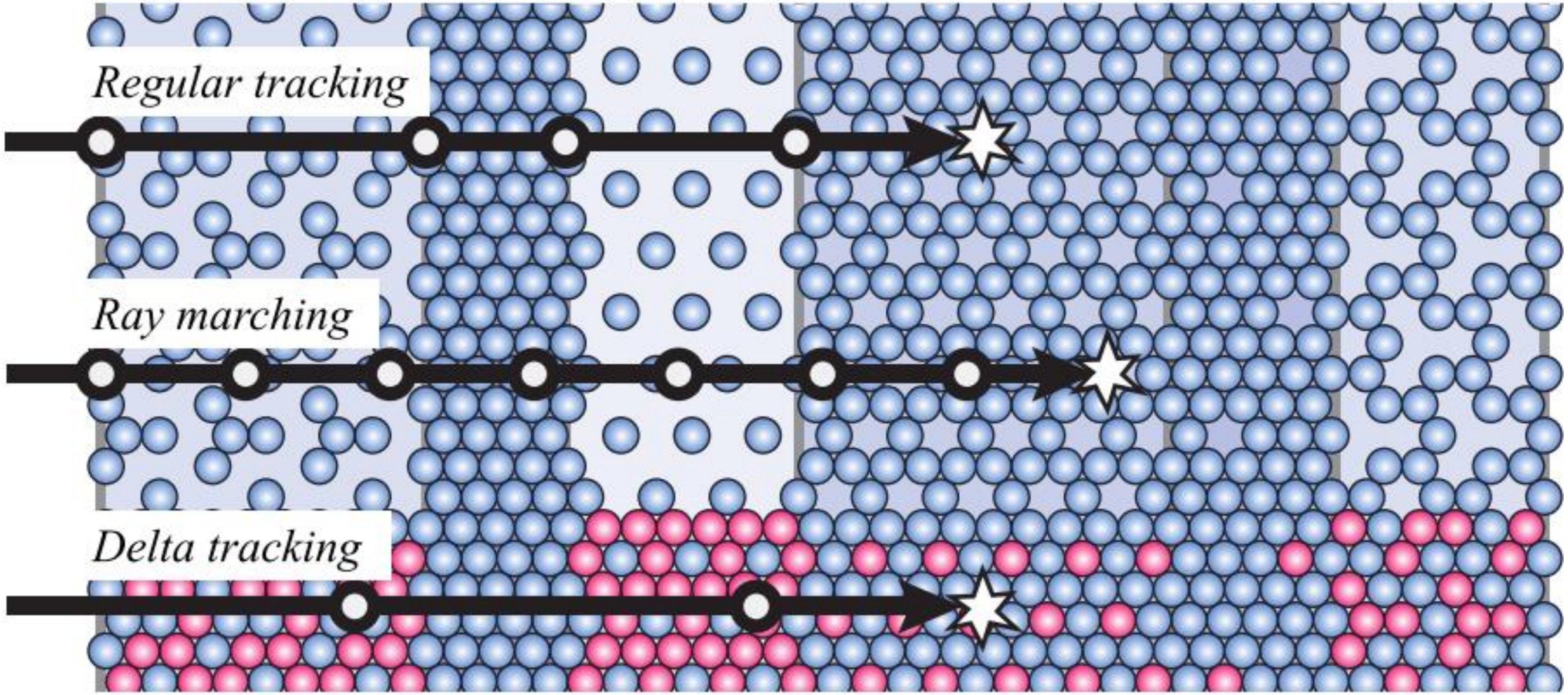
- Expected iterations: c

Homogenous media

- In homogeneous media, where the extinction μ_t is spatially invariant
- Optical thickness is linearly proportional to the distance t along the ray: $\tau(t) = t\mu_t$.

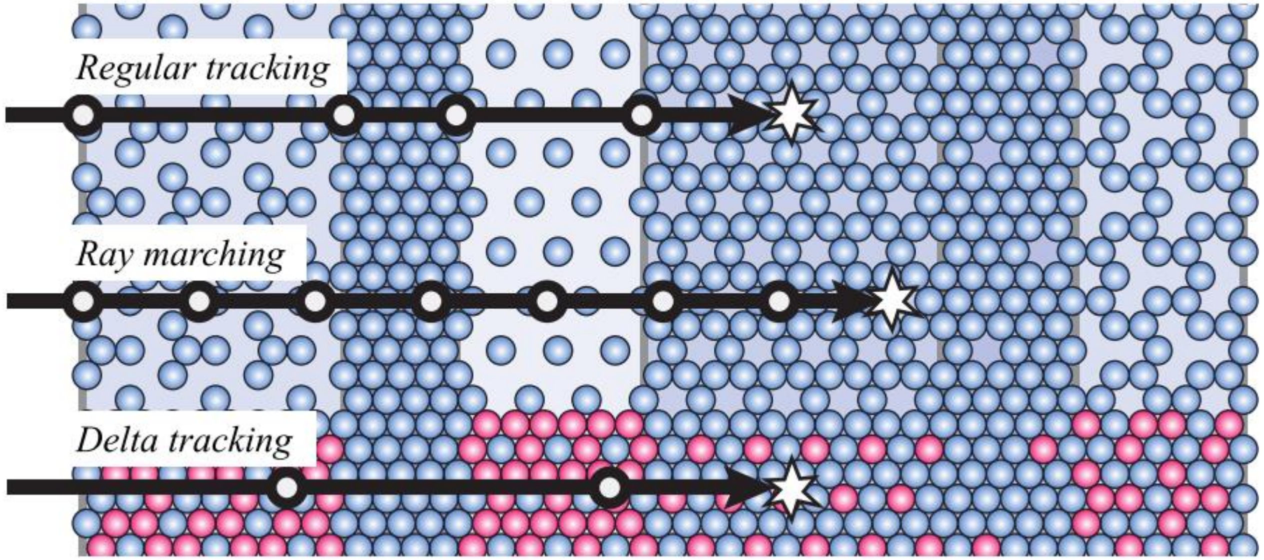
- The free-path sampling CDF thus simplifies to $F(t) = 1 - e^{-\mu_t t}$
- $t(\xi) = -\frac{\ln(1-\xi)}{\mu_t}$ Inversion method with PDF $p(t) = \mu_t e^{-\mu_t t}$

Sampling the Media: Delta Tracking



Sampling the Media: Delta Tracking

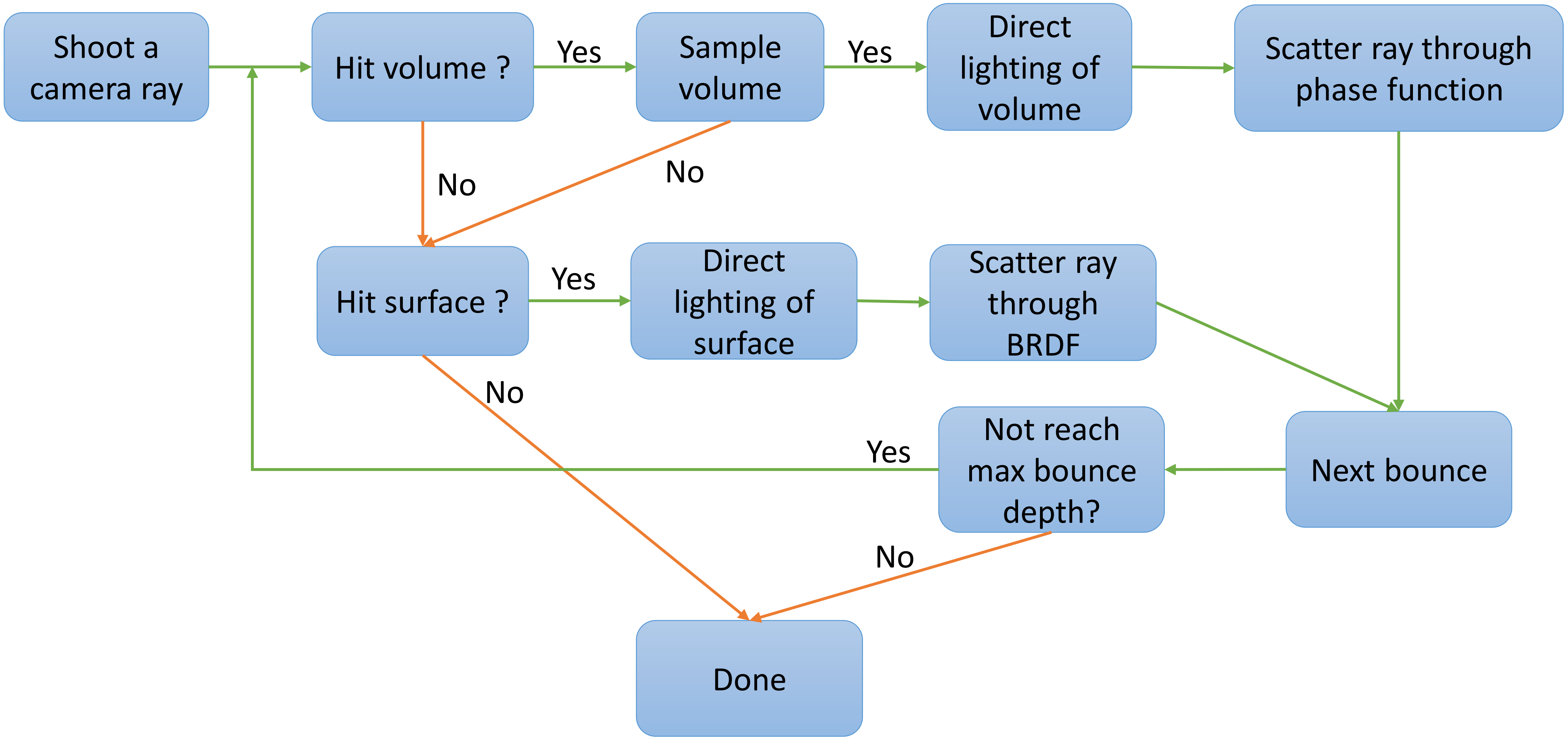
- By adding virtual particles, we homogenize the heterogeneous media.
- Delta tracking
  - Frist sample a free flight distance according to a majorant extinction to generate a tentative interaction.
  - Decide whether it is a ‘real’ collision based on local fraction of real particles.
  - Continue until we sampled a real interaction in media or go beyond the max range of ray.



Sampling the Media: Delta Tracking

Closed-form tracking	**Regular tracking**	**Ray marching**	**Delta tracking**
simple media such as homogeneous, linear, or exponential	piecewise "simple" media	any media	any media with bounded $\mu_t(\mathbf{x})$
efficient but highly limited	iterative, inefficient if free paths cross many boundaries	iterative, inefficient for media with high frequencies	iterative, inefficient if too much fictitious matter is added
unbiased	unbiased	biased	unbiased
known PDF	known PDF	known (approximate) PDF	unknown PDF

Volumetric path tracing



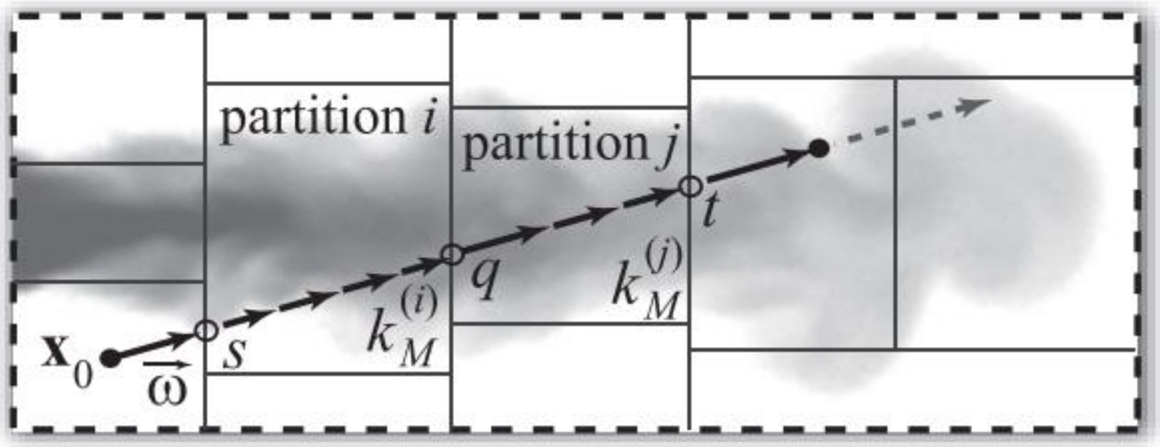
Sad story: delta tracking is super slow…

- Loose majorant cause lots of fictious collision, which does not contribute to the light path.
- Large majorant value means short sampling distance. Light was trapped in the volume and unable to travel through it.

- Solution:
- Make majorant tighter!

Adaptive sampling using spatial partition

- How to partition the volume?
- Uniform grid: feasible but inefficient.
- Kd tree. Is a better solution.

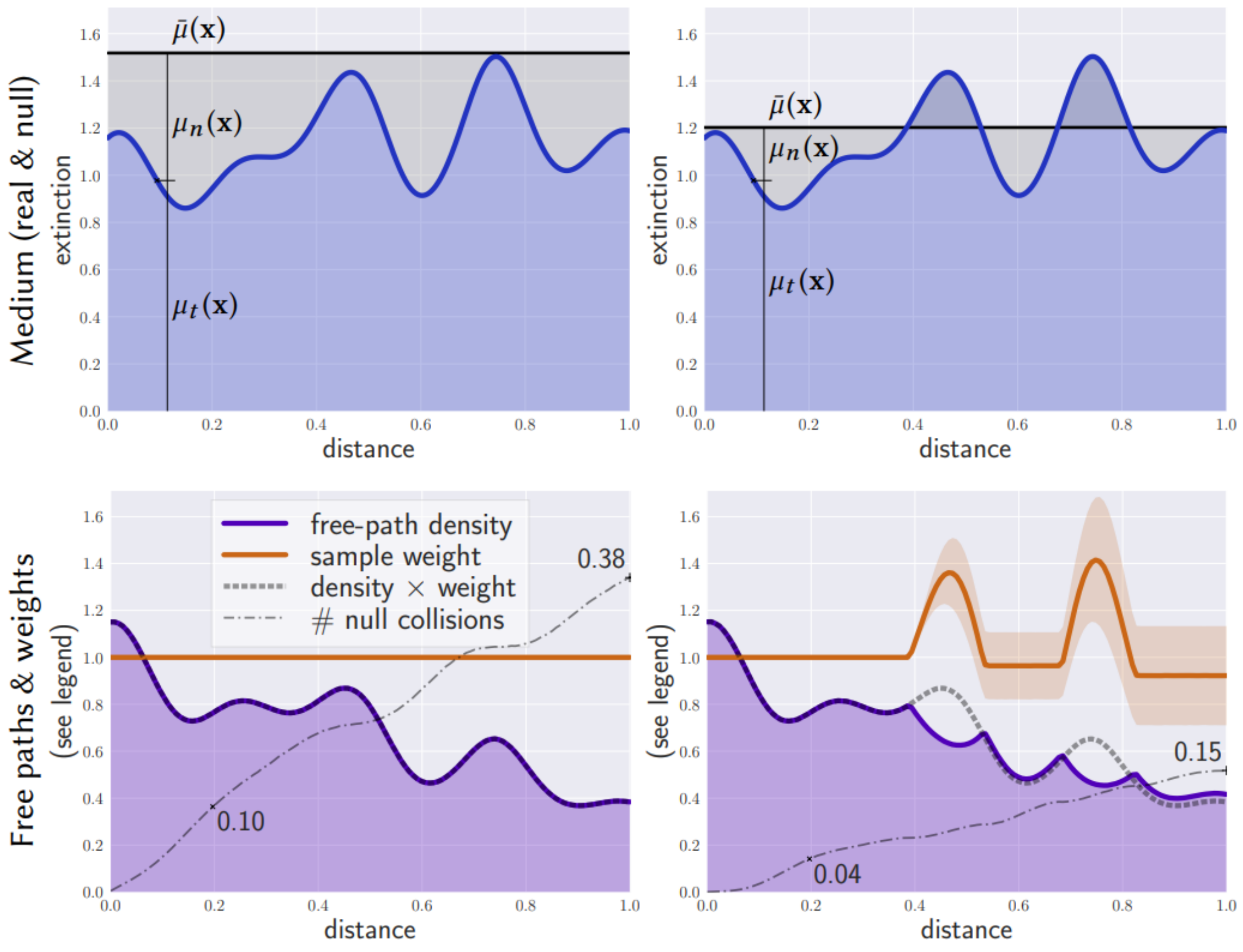


Weighted delta tracking for sampling

- Is tight majorant necessary?
- No!
- We use an additional weight to amend the density

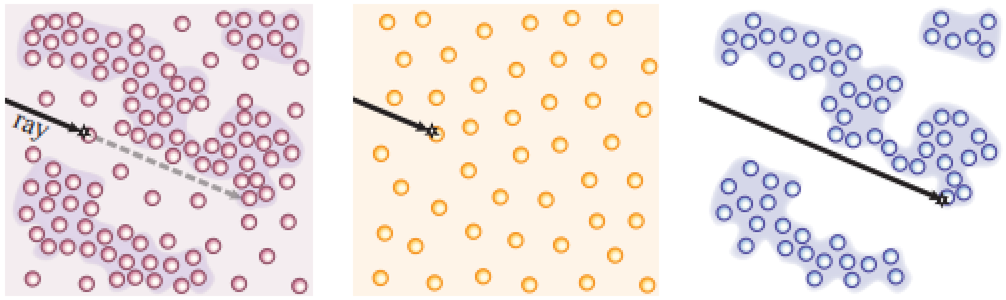
$$w_{real} = \frac{\mu_t(x) + |\mu_n(x)|}{\bar{\mu}(x)}$$

$$w_n = \frac{\mu_n(x)\,(\mu_t(x) + |\mu_n(x)|)}{\bar{\mu}(x)|\mu_n(x)|}$$

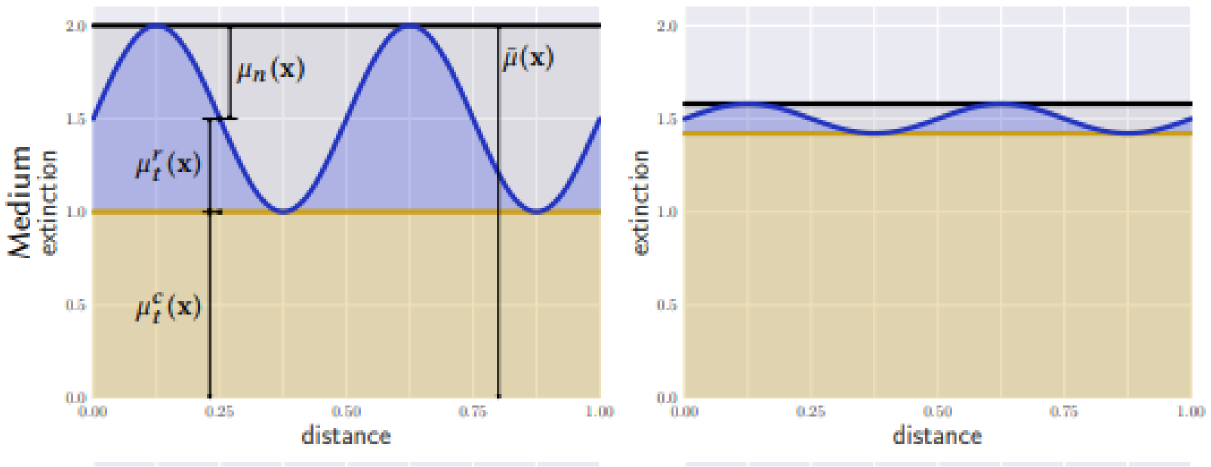


(a) Bounding $\bar{\mu}$ (b) Non-bounding $\bar{\mu}$

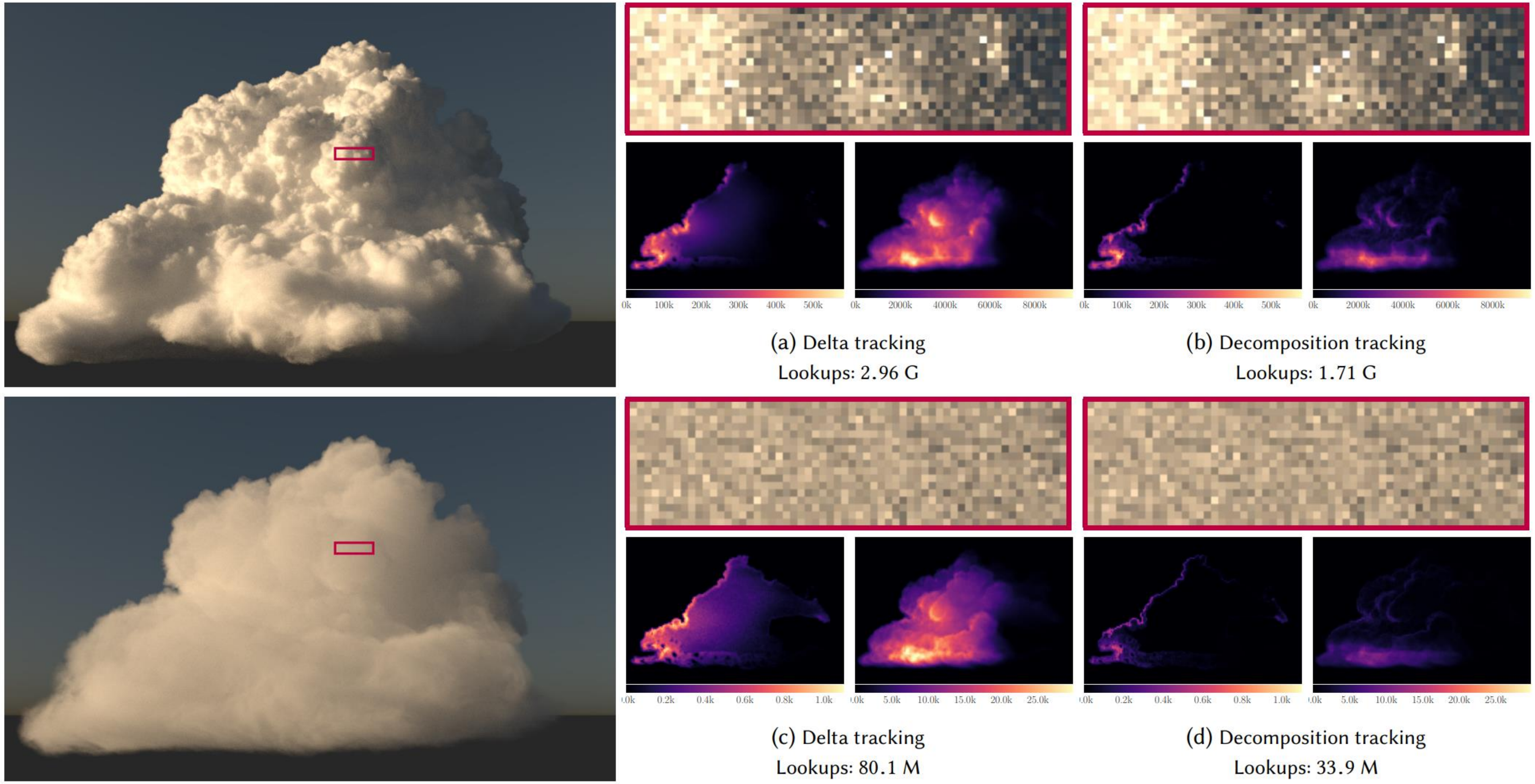
Decomposition tracking for sampling



(a) Original medium (b) Control medium (c) Residual medium



Decomposition tracking for sampling



Estimate transmittance

- Since we can generate sample based on Tr, then the transmittance is just the expectation of random indicator.
  - If ray pass through the media, return 1
  - If not, return 0
- Or we can multiply the probability of continuation at each interaction in media.

Estimate transmittance: ratio tracking

Algorithm 2: *Pseudocode of the ratio tracking estimator of transmittance along a ray with origin o, direction ω, and length d.*

RatioTrackingEstimator(o, ω, d)

$t = 0$
$T = 1$
do:
  $\zeta = \text{rand}()$
  $t = t - \frac{\log(1-\zeta)}{\bar{\mu}}$
  if $t \geq d$: **break**
  $T = T * \left(1 - \frac{\mu(o+t*\omega)}{\bar{\mu}}\right)$
while true
return T

Speed up for real time rendering

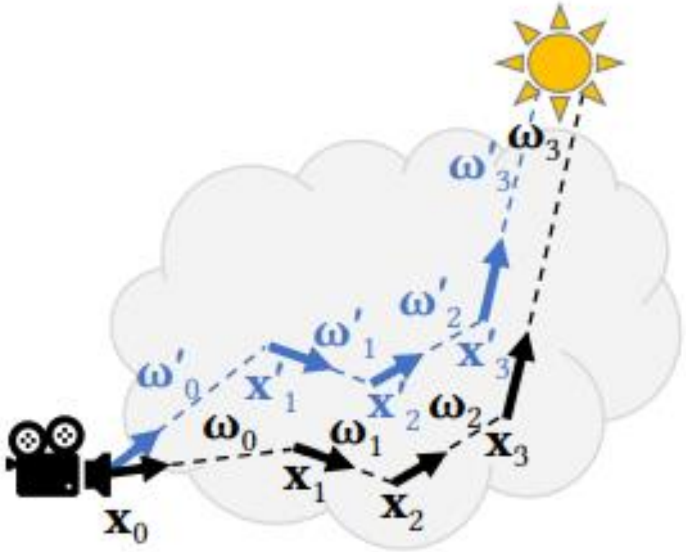
- Volume rendering is still too slow for real time rendering with all techniques above…
- Intuitive: Reuse sample path from neighboring pixels and frames.

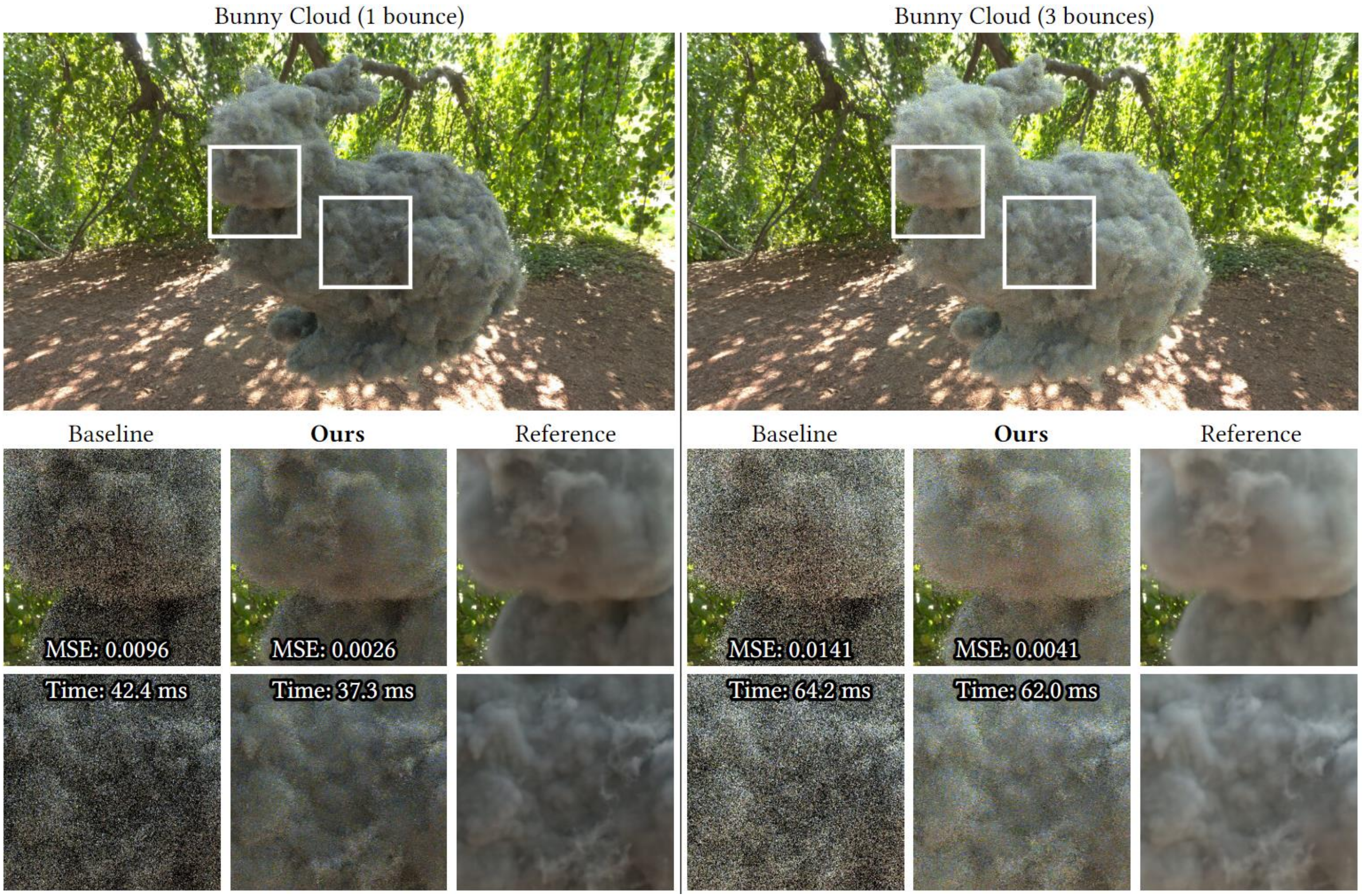
Speed up for real time rendering

Each pixel has a corresponding reservoir, recording one light path.

The light path keeps updated when we generate more sample points.

Each pixel reuse path from neighboring pixels.




Bunny Cloud (1 bounce)
Bunny Cloud (3 bounces)
Baseline
Ours
Reference
Baseline
Ours
Reference
MSE: 0.0096
MSE: 0.0026
MSE: 0.0141
MSE: 0.0041
Time: 42.4 ms
Time: 37.3 ms
Time: 64.2 ms
Time: 62.0 ms

Something about implementation

- Ray tracing framework: Intel® Embree 3 (CPU), Nvidia OptiX 7 (GPU)
- Sparse volumetric data storage: OpenVDB
- For rendering, nanovdb is a better choice
- nanovdb can be used in GPU and is further optimized for traversal
- Denoiser:
  - Intel® Open Image Denoise library
  - NVIDIA OptiX™ AI-Accelerated Denoiser

Something about implementation

Embree API (C99 and ISPC)

Ray Tracing Kernel Selection

Acceleration Structures

bvh4.triangle4
bvh8.triangle4
bvh4.quad4v
...

Builders

SAH Builder
MBlur Builder
Spatial Split Builder
Morton Builder
BVH Refitter

Traversal

Single Ray
Packet/Hybrid
Ray Stream

Intersection

Möller-Trumbore
Plücker
Flat Curve
Round Curve
Oriented Curve
Grid

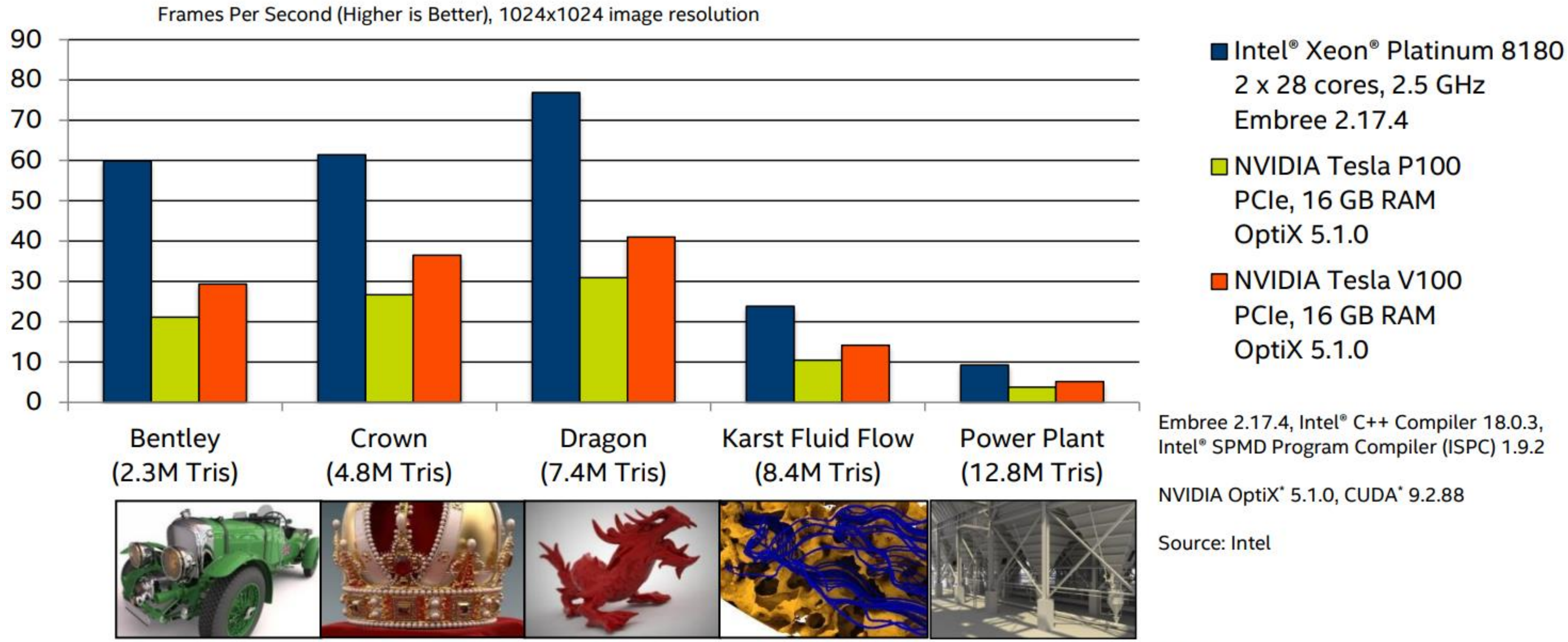
Subdiv Engine

B-Spline Patch
Gregory Patch
Tessellation Cache
Displ. Mapping

Common Vector and SIMD Library

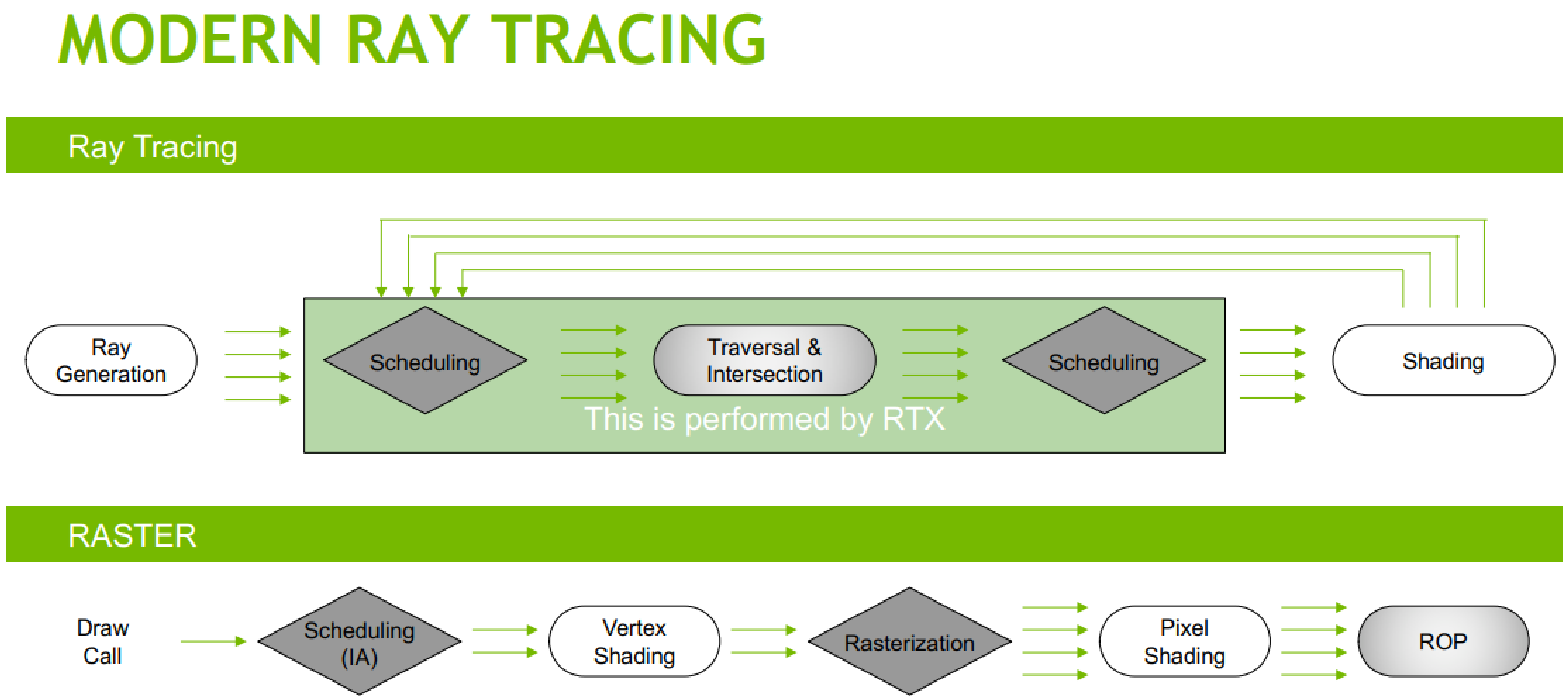
(Vec3f, Vec3fa, vfloat4, vfloat8, vfloat16, ..., Intel® SSE2, Intel® SSE4.1, Intel® AVX, Intel® AVX2, Intel® AVX-512)

Something about implementation



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.

Something about implementation



Something about implementation

TRAVERSABLES

Start of ray tracing work

General:

Geometry Acceleration structure (GAS)

- Triangles or custom primitives

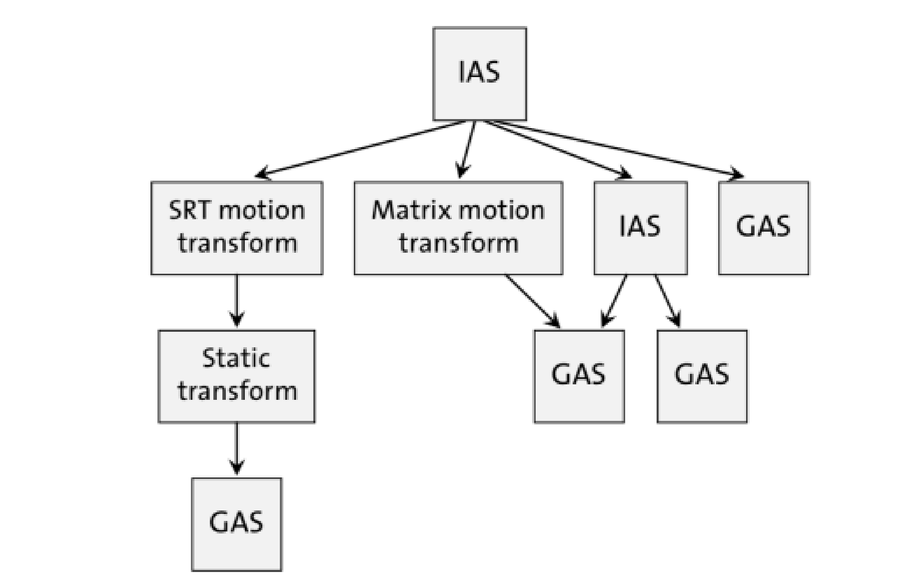
Instance Acceleration structure (IAS)

Motion blur:

Matrix motion transform

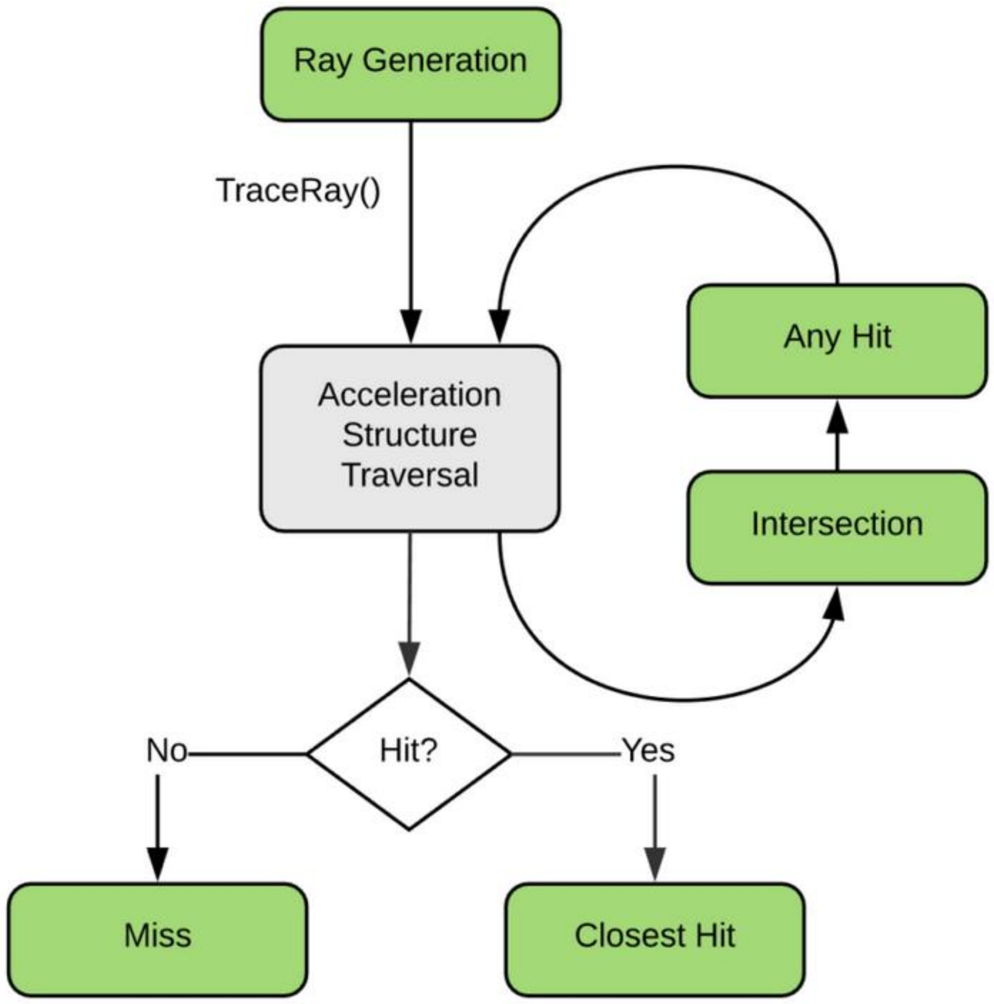
SRT motion transform

Static Transform (rarely used)

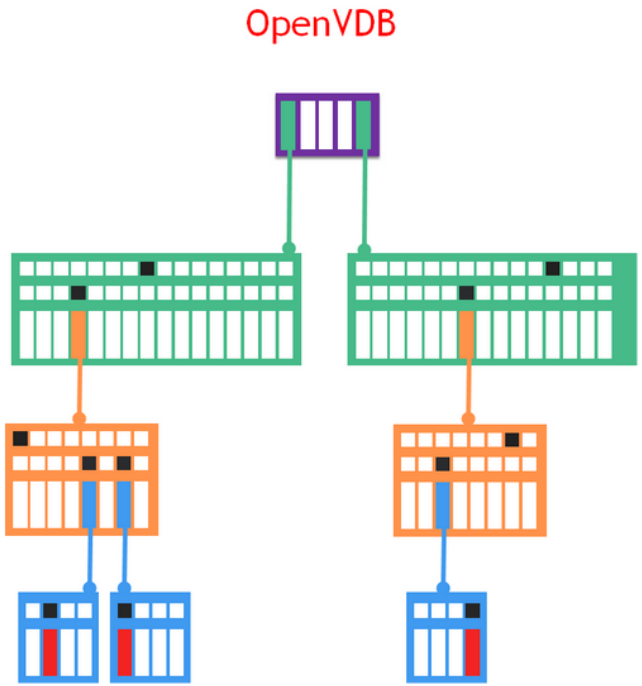


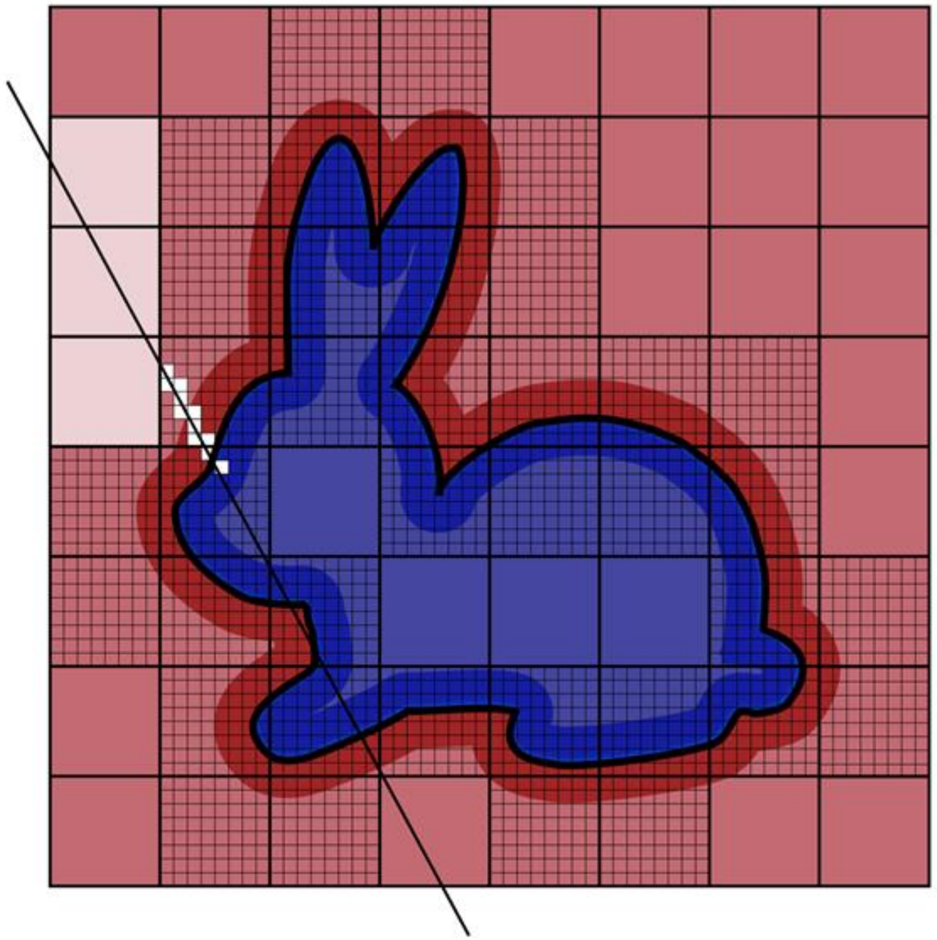
NVIDIA

Something about implementation



Something about implementation





Reference

- Jarosz W. Monte Carlo Methods for Volumetric Light Transport Simulation[J].
- Novák J, Selle A, Jarosz W. Residual ratio tracking for estimating attenuation in participating media[J]. ACM Trans. Graph., 2014, 33(6): 179:1-179:11.
- Kutz P, Habel R, Li Y K, et al. Spectral and decomposition tracking for rendering heterogeneous volumes[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1-16.
- Yue Y, Iwasaki K, Chen B Y, et al. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media[J]. ACM Transactions on Graphics (TOG), 2010, 29(6): 1-8.
- Daqi L. Fast Volume Rendering with Spatiotemporal Reservoir Resampling. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2021)
- NVIDIA OptiX™ Ray Tracing Engine | NVIDIA Developer
- Intel Embree
- OpenVDB

Thanks