Path tracing in Production

Part 1: Modern Path Tracing

Luca Fascione (Organizer)
Weta Digital

Johannes Hanika (Organizer)
Weta Digital

Daniel Heckenberg
Animal Logic

Christopher Kulla
Sony Imageworks

Marc Droske
Weta Digital

Jorge Schwarzhaupt
Weta Digital

SIGGRAPH 2019 Course Notes

Imagery from recent movie productions, featuring difficult light transport, massive geometric complexity, intricate material detail, as well as volumetric effects. The LEGO Movie 2: The Second Part *Image courtesy of Animal Logic, ©2019 Warner Bros. Entertainment Inc. ©The LEGO Group. All rights reserved. Spiderman: Homecoming ©2017 CTMG, Inc. All rights reserved. Alita: Battle Angel ©2018 Twentieth Century Fox Film Corporation. All rights reserved.*

SIGGRAPH '19 Courses, July 28 - August 01, 2019, Los Angeles, CA, USA
ACM 978-1-4503-6307-5/19/07.
10.1145/3305366.3328079

Abstract

In the past few years the movie industry has switched over from stochastic rasterisation approaches to using physically based light transport simulation: path tracing in production has become ubiquitous across studios. The new approach came with undisputed advantages such as consistent lighting, progressive previews, and fresh code bases. But also abandoning 30 years of experience meant some hard cuts affecting all stages such as lighting, look development, geometric modelling, scene description formats, the way we schedule for multi-threading, just to name a few. This means there is a rich set of people involved and as an expert in either one of these aspects it is easy to lose track of the big picture.

This is part I of a full-day course, and it focuses on the necessary background knowledge. In this part, we would like to provide context for everybody interested in understanding the challenges behind writing renderers intended for movie production work. In particular we will give an insight into movie production requirements for new students and academic researchers. On the other side we will lay a solid mathematical foundation to develop new ideas to solve problems in this context.

To further illustrate, part II of the course will focus on materials (acquisition and production requirements) and showcase practical efforts by prominent professionals in the field, pointing out unexpected challenges encountered in new shows and unsolved problems as well as room for future work wherever appropriate.

Contents

1 Introduction to path tracing and Monte Carlo sampling

Johannes Hanika, *Weta Digital*
Luca Fascione, *Weta Digital*

This section summarises a few basic concepts of light transport and introduces the Monte Carlo integration scheme. This forms the foundation of the path tracing family of algorithms, which is used to synthesise pictures in a unified way. While we want to introduce all commonly used equations and explain the terms therein, this will mainly form a basis for common notation rather than explain the underlying principles and the algorithms to solve the equations in exhausting detail. For a broader introduction we refer to Pharr et al. [2017]. A thorough introduction to the radiometric quantities is given by Chandrasekar [1960], and a good entry point for especially the volumetric scattering aspects are the dissertations by Jarosz [2008], Novák [2014] and the recent state of the art report by Novák et al. [2018].

1.1 Transport equations

We need to derive a measurement function which can be used to evaluate throughputs for transport paths as they are constructed during path tracing. Intuitively, this will require us to measure some physical quantity along a ray, i.e. an infinitesimally narrow direction emergent from an infinitesimally small position. This is rather abstract, so we will proceed by simplifying the problem setting somewhat, introducing the notion of photons, and then resort to simply counting photons. From there, we will derive what is the quantity the flows through a ray of light (radiance) and the equations that govern its flow (or *transport*)through the scene. This approach is inspired by Arvo [1993].

1.1.1 Assumptions

Before we dive into equations, let's make some simplifying assumptions to make our task easier. Most assumptions to be made about light transport have been prominently generalised away in computer graphics literature, so our list here comes with references to works that extended rendering beyond the discussed limitation:

- Light consists of particles (*photons*), not waves. In particular we are not interested in interference (as opposed to Werner et al. [2017]) or polarisation (unlike Jarabo and Gutierrez [2016]).
- Photons travel along straight lines, because the index of refraction does not vary continuously but only at interfaces (contrary to Ament et al. [2014]). Also photons are not affected by gravity (as they would on astronomical scales such as Thorne [2014]).
- Light is transported instantly, or, equivalently, we're only interested in some equilibrium state with constant boundary conditions (even though Jarabo et al. [2014] showed that simulating time dependence results in nice visualisations).
- We only model elastic scattering, i.e. there is no energy transfer between wavelengths, no photoluminescence (as described for instance by Glassner [1995], Hullin et al. [2010]).
- Volumes: to derive the transport equations, we will assume a uniform random distribution of light-interacting particles (unlike Bitterli et al. [2018], d'Eon [2018], Jarabo et al. [2018]) which are much larger than the wavelength (as opposed to Rayleigh scattering as introduced by Strutt [1871]), and have isotropic cross-sections (because we avoid anisotropic media described by Jakob et al. [2010]).

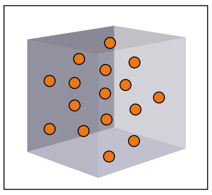
1.1.2 Counting photons

To quantify light, let's start with a straightforward concept: we'll be counting photons. A single photon corresponds to an atomic portion of *energy* E (measured in joule $[J]$):

$$E = \frac{h \cdot c_m}{\lambda} \quad [J]. \tag{1}$$

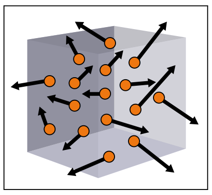
Here, $h \approx 6.62607004 \times 10^{-34}$ $[m^2 kg/s = Js]$ is Planck's constant and $c_m = c/\eta_m$ is the speed of light in a material with index of refraction η_m. This is slightly slower than the speed of light in vacuum, which is conveniently fixed by definition at exactly $c = 299,792,458\ [m/s]$. In equation (1), $\lambda\ [m]$ is the wavelength of the photon. We'll often measure it in nanometers $[nm]$ instead to help distinguish it from world space lengths.

To determine the overall *radiant energy* in a certain volume, all we need to do is count photons and add up their energies. Assuming they all have the same wavelength, they all have the same energy and we can just multiply the energy by their count $\#P$:



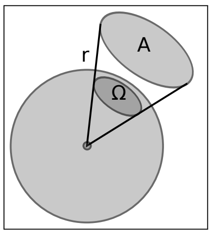
$$Q = \#P \cdot \frac{h \cdot c_m}{\lambda} \left[J = \frac{kg \cdot m^2}{s^2} \right]. \tag{2}$$

Unfortunately, photons can be moving very quickly and thus a more practically useful quantity is how many photons pass through a certain volume *per time*. This is called *radiant power*, or *flux*, and is measured in watts:



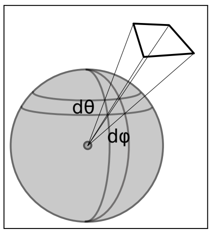
$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t} \left[\frac{J}{s} = W \right]. \tag{3}$$

To derive a quantity that considers the direction of the photon, we need a measure of directions. This two dimensional measure space is called *solid angle* and defined as the area of a piece of surface projected onto the unit sphere, measured in *steradians* $[sr]$:



$$\Omega_A = A/r^2\ [sr]. \tag{4}$$

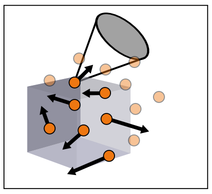
To perform actual integration in this domain, we often times need to re-parametrise the domain into polar coordinates, for instance using the latitude/longitude way. In general, spherical polar coordinates would require also a radius. Since we are only interested in directions ω, the 2D angular domain $(\theta, \varphi) \in [0, \pi] \times [0, 2\pi]$ suffices in this case, remembering the adjustment due to the change of variables[1]:



$$\mathrm{d}\omega = \mathrm{d}\theta\, |\sin\theta|\, \mathrm{d}\varphi, \tag{5}$$

$$\int_\Omega \mathrm{d}\omega = \int_0^{2\pi} \int_0^{\pi} \sin\theta\ \mathrm{d}\theta\ \mathrm{d}\varphi = 4\pi. \tag{6}$$

Since photons live in *phase space*, i.e. have a 3D position and a 2D direction, we will associate $(\mathbf{x}, \omega) \in V \times \Omega$ with them. With these tools at hand, we're finally ready to count photons per time per volume and per solid angle. Ideally we would count some place holder quantity $\cdot$ with some imaginary measurement device that only counts photons inside a certain 3D volume if their direction ω falls within the solid angle subtended by a certain imaginary funnel:



$$\Phi = \int_\Omega \int_V \cdot(\mathbf{x}, \omega)\ \mathrm{d}\mathbf{x}\, \mathrm{d}\omega\ [W]. \tag{7}$$

We will be assuming a locally uniform distribution of photons such that we can assume a piecewise smooth function $\cdot(\mathbf{x}, \omega)$ and expect to get the same answer by integrating it over a volume $V \times \Omega$ in phase space as we would

[1]The intuitive notion here is that as θ moves away from π, the length of the constant-φ lines change in length proportionally to $\sin\theta$

by counting individual photons.[2] The approach we will take in the next paragraphs, to both define such a function and to introduce the transport equations governing its propagation, is the following: First we will define the effects that change the distribution of photons in phase space, collecting all these results in a differential equation, the *radiative transfer equation* (RTE). From there we will derive the more well-known integral equation.

1.1.3 Events that change photon count

The most obvious event is *photon emission*, where particles emit light (for instance modelled by a black body emitter). Other than that, there are two things that obviously change photon count in a certain region of phase space. The first one is the most natural: *streaming*: This happens when the photon enters or leaves the volume crossing its boundary because it's continuing on its way. The other one is *collision*, when the photon interacts with matter inside the volume under consideration. In a collision event, a photon can be absorbed or scattered. While absorption is always a loss, scattering may mean both: the new direction of the photon may be inside the solid angle spanned by the current region of the phase space, or it may be not. We'll go through them in an order that makes reasoning about the quantities as easy as possible.

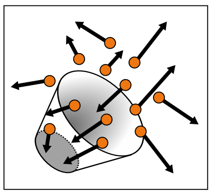
Absorption Collision with an absorbing particle is easy to model, since all that happens is that the photon disappears. All we need to model is how often such an event happens. We will be using a statistical model which can be used to determine the chance that a photon interacts with such a particle while travelling a unit distance. As mentioned earlier, we assume a locally uniform random distribution of absorbing particles, such that we can compute a locally constant density ρ in $\left[1/m^3\right]$. We will also only deal with isotropic cross-sections σ of the particles, i.e. σ $\left[m^2\right]$ does not depend on direction ω. From these two, we define the *collision coefficient* $\mu = \sigma \cdot \rho$ $[1/m]$. This is the probability of collision while travelling unit distance in the medium, or, conversely, $1/\mu$ is the *mean free path*. In neutron transport literature, this is sometimes referred to as Σ.

To model heterogeneous media, we will work with $\mu(\mathbf{x})$ to express macroscopic inhomogeneities while still assuming locally uniform random distribution of particles at microscopic scale. This is important for the derivation of the differential equations later on.

We will need a few such collision coefficients for the different types of events. For absorption, we are dealing with $\mu_a(\mathbf{x})$. If a photon intersects such a particle, it is lost and we can compute the change in flux:

$$-\int_\Omega \int_V \mu_a(\mathbf{x}) L(\mathbf{x}, \omega) \; \mathrm{d}\mathbf{x} \; \mathrm{d}\omega \; [W]. \tag{8}$$

This equation shows that the unit of $\mu_a(\mathbf{x})L(\mathbf{x},\omega)$ has to be $\left[\frac{1}{m} \cdot \frac{W}{m^2 sr} = \frac{W}{m^3 sr}\right]$ to integrate to watts $[W]$. We take this opportunity to define *radiance* as the core unit which is transported along a light transport path. It can be measured by an imaginary differential measurement device which consists of a surface counting photons (per time unit) passing through it and a funnel attached to it[3]. As both the surface and the funnel simultaneously tend to zero size (in area and solid angle, respectively), the measured quantity approaches radiance for the resulting ray $(\mathbf{x}, \omega)$:



$$L(\mathbf{x}, \omega) \qquad \left[\frac{W}{m^2 sr}\right]. \tag{9}$$

Emission Emission is very similar to absorption, in that a black body emitter model dictates that every photon intersecting an emitting particle is absorbed rather than reflected. This means that emission comes with a loss term very similar to the one for absorption. In fact, some formulations assume that μ_a includes the density of emitting particles, too. For increased clarity, sometimes the emission coefficient is explicitly modelled as $\mu_e(\mathbf{x})$.

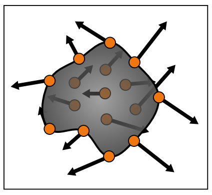
[2] This seems to be one of the longer lasting assumptions, at least we are not aware of computer graphics literature generalising the equations to count individual photons

[3] Each photon is accounted for by the energy it carries, so its contribution is scaled by its wavelength according to equation (1)

We introduce another symbol for emitted radiance $L_e(\mathbf{x}, \omega)$ to be able to quantify the amount of photons which are added per time in a certain volume $V \times \Omega$:

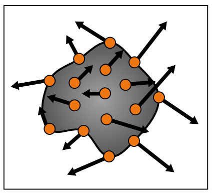
$$\int_\Omega \int_V \mu_e(\mathbf{x}) L_e(\mathbf{x}, \omega)\ \mathrm{d}\mathbf{x}\, \mathrm{d}\omega\ [W]\,. \tag{10}$$

Streaming To quantify how many photons enter and leave the current piece of phase space $V \times \Omega$, we want to count the number of photons passing through the spatial boundaries of V, which is written as ∂V. This is measured perpendicularly to the surface normal $n(\mathbf{x})$. Since the quantity $L(\mathbf{x}, \omega)$ depends on ω but itself is just a scalar and no vector field, we will measure $\omega \cdot L(\mathbf{x}, \omega)$:



$$-\int_\Omega \int_{\partial V} \langle \omega \cdot L(\mathbf{x}, \omega), n(\mathbf{x}) \rangle\ \mathrm{d}\mathbf{x}\, \mathrm{d}\omega. \tag{11}$$

The integration over the boundary ∂V will get in the way later on, so we replace it by an integration over the whole volume V by using Gauss' theorem, leaving us with a directional derivative in the integrand:



$$= -\int_\Omega \int_V \frac{\partial}{\partial \omega} L(\mathbf{x}, \omega)\ \mathrm{d}\mathbf{x}\, \mathrm{d}\omega. \tag{12}$$

Scattering Once a photon collides with a particle, it can also be scattered, i.e. experience a change of direction ω. As mentioned before, we want to limit ourselves to *elastic scattering*. This means the energy of the photon remains unchanged during the event. In particular this means it does not change wavelength, so we can limit our analysis to one wavelength at a time, and solve the resulting equations per wavelength. (we'll briefly come back to this assumption of excluding fluorescence in section 1.4).

We model scattering by a kernel $\mathrm{k}(\omega_i, \mathbf{x}, \omega)$ which determines the directional change at position $\mathbf{x}$. Depending on the new direction, this may result in a gain or a loss: the photon may be scattered into the solid angle of our piece of phase space, or out of it. We can write the in-scattering gain as

$$+\int_\Omega \int_V \int_\Omega \mathrm{k}(\omega_i, \mathbf{x}, \omega) L(\mathbf{x}, \omega_i) \mathrm{d}\omega_i \mathrm{d}\mathbf{x} \mathrm{d}\omega, \tag{13}$$

and similarly the out-scattering loss as

$$-\int_\Omega \int_V \int_\Omega \mathrm{k}(\omega, \mathbf{x}, \omega_o) L(\mathbf{x}, \omega) \mathrm{d}\omega_o \mathrm{d}\mathbf{x} \mathrm{d}\omega. \tag{14}$$

This last term can be simplified considerably by assuming the scattering kernel is separable in space and direction, i.e.

$$\mathrm{k}(\omega_i, \mathbf{x}, \omega) = \mu_s(\mathbf{x}) \cdot f_s(\omega \cdot \omega_i), \tag{15}$$

because then we can pull the scattering coefficient μ_s out of the inner integral over solid angle ω_o:

$$-\int_\Omega \mathrm{k}(\omega, \mathbf{x}, \omega_o) L(\mathbf{x}, \omega)\ \mathrm{d}\omega_o = -\mu_s(\mathbf{x}) \int_\Omega f_s(\omega_o \cdot \omega) L(\mathbf{x}, \omega)\ \mathrm{d}\omega_o \tag{16}$$

$$= -\mu_s(\mathbf{x}) L(\mathbf{x}, \omega) \int_\Omega f_s(\omega_o \cdot \omega)\ \mathrm{d}\omega_o \tag{17}$$

$$= -\mu_s(\mathbf{x}) L(\mathbf{x}, \omega), \tag{18}$$

where in the last step, we used the property of the *phase function* $f_s()$ that it integrates to one over the sphere Ω. Because it is normalised, this phase function $f_s()$ is a probability density function determining where a particle is reflected to when intersecting a scattering particle, as determined by the collision coefficient μ_s. Often we assume isotropic media, i.e. the phase function only depends on the cosine between incoming and outgoing directions, i.e. $f_s(\omega \cdot \omega_i)$. The most simple incarnation is isotropic scattering, where the outgoing direction does not even depend on the incoming direction at all, i.e. a constant distribution $f_s \equiv \frac{1}{4\pi}$. The more often used variant is the popular phase function by Henyey and Greenstein [1941]:

$$f_s(\cos\theta) = \frac{1}{4\pi}\frac{1-g^2}{(1+g^2-2g\cos\theta)^{3/2}}, \tag{19}$$

that contains a single parameter g, the mean cosine, which determines how forward ($g = 1$) or backward scattering ($g = -1$) the phase function is.

To further simplify the equations, we combine the losses due to absorption, shadowing by emissive particles, and out-scattering, and define the *extinction coefficient*:

$$\mu_t(\mathbf{x}) = \mu_a(\mathbf{x}) + \mu_e(\mathbf{x}) + \mu_s(\mathbf{x}). \tag{20}$$

1.1.4 The radiative transfer equation

Now that we discussed all events, we can proceed and combine them in a single equation. As dictated by energy conservation, all terms need to sum up to zero (because we considered all ways photons may be added or lost). Doing so gives us

$$\begin{aligned} 0 = \int_\Omega \int_V & \overbrace{-\frac{\partial}{\partial\omega}L(\mathbf{x},\omega)}^{\text{streaming}} + \overbrace{\mu_e(\mathbf{x})L_e(\mathbf{x},\omega)}^{\text{emission}} \overbrace{-\mu_t(\mathbf{x})L(\mathbf{x},\omega)}^{\text{extinction}} \\ & + \underbrace{\mu_s(\mathbf{x})\int_\Omega f_s(\omega_i\cdot\omega)L(\mathbf{x},\omega_i)\mathrm{d}\omega_i}_{\text{in-scattering}} \,\mathrm{d}\mathbf{x}\,\mathrm{d}\omega. \end{aligned} \tag{21}$$

Since this equation holds for integration over any arbitrary part of phase space, they have to hold for every individual point, too. This means we can leave away the integration over phase space $\Omega \times V$, and arrive at a simpler differential equation, the *radiative transfer equation* (RTE) as discussed by Chandrasekar [1960], which defines the change of radiance L at a point $\mathbf{x}$ in direction ω is due to emission (μ_e), extinction (μ_t), and scattering (μ_s):

$$\begin{aligned} \frac{\partial}{\partial\omega}L(\mathbf{x},\omega) &= \mu_e(\mathbf{x})L_e(\mathbf{x}) - \mu_t(\mathbf{x})L(\mathbf{x},\omega) \\ &\quad + \mu_s(\mathbf{x})\int_\Omega f_s(\omega\cdot\omega_i)L(\mathbf{x},\omega_i)\,\mathrm{d}\omega_i. \end{aligned} \tag{22}$$

The directional derivative $\frac{\partial}{\partial\omega}$ is sometimes written as $\omega \cdot \nabla$. This formulation presents an alternate intuitive way of looking at the terms: $\frac{\partial}{\partial\omega}L(\mathbf{x},\omega)$, the left hand side, is the change of radiance in direction ω. To find out how radiance changes along the ray direction ω, we need to add emission at this point, subtract extinction, and add the in-scattered radiance over all incoming directions ω_i.

The mathematical structure of equation (22) is an integro-differential equation, since it contains differentials and integrals. This makes it hard to solve, especially with path tracing: in rendering we know how to solve difficult high-dimensional integrals with the Monte Carlo method, thus we'll have to work on the differential.

In an effort to integrate both sides of equation (22), let's look at a simple one dimensional example of a differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}x}L(x) = -\mu_t L(x). \tag{23}$$

where of course potential similarity in naming of the variables are entirely coincidental. To solve it, we'll need boundary conditions, in our example $x > 0$, $L(0) = 1$. Given this, we know a closed-form solution to this problem, using and integrating factor:

$$L(x) = \exp(-\mu_t \cdot x) \cdot L(0). \tag{24}$$

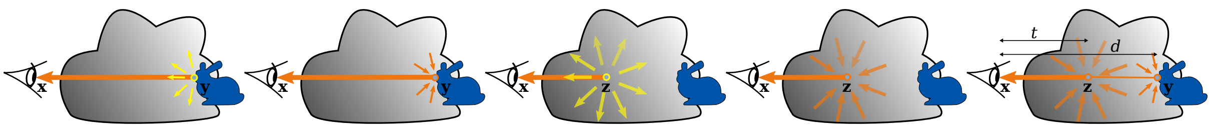
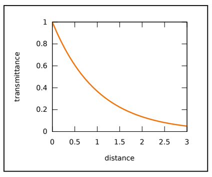


Figure 1: *Illustration of all events taking part in the change of radiance along a ray: surface emission (a), surface scattering (b), volume emission (c), volume scattering (d), and the distances as they appear in equation (28).*

Transmittance

Applying a very similar reasoning as in this last example in equation (23) to the RTE in equation (22), we arrive at a very similar exponential factor:



$$T(\mathbf{x},\mathbf{y}) = e^{-\tau(\mathbf{x},\mathbf{y})}, \tag{25}$$

$$\tau(\mathbf{x},\mathbf{y}) = \int_0^d \mu_t(\mathbf{x} + t\cdot\omega)\mathrm{d}t, \tag{26}$$

where $d = \|\mathbf{x}-\mathbf{y}\|$ and $\omega = (\mathbf{x}-\mathbf{y})/d$. $T(\mathbf{x},\mathbf{y})$ is called the *transmittance* and computes the fraction $\in [0,1]$ of light which will make it from $\mathbf{x}$ to $\mathbf{y}$. This attenuation is called the Beer-Lambert law, and $\tau(\mathbf{x},\mathbf{y})$ is called the *optical thickness*. Note how the transmittance for homogeneous materials (i.e. $\mu_t \equiv$ const.) degenerates to a factor very much like the one in equation (24), illustrated in the graph next to equation (25). On the other hand, this classic kind of exponential attenuation is challenged by recent works by Bitterli et al. [2018], d'Eon [2018] and Jarabo et al. [2018], who do not assume a uniform random distribution of particles.

Boundary conditions

To solve the RTE, we still need to define boundary conditions. These can be found in surface transport, formalised as a recursive integral equation by Kajiya [1986]:

$$L(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_\Omega f_r(\mathbf{x},\omega,\omega_i)L(\mathbf{x},\omega_i)\,\mathrm{d}\omega_i^\perp, \tag{27}$$

where the integration domain Ω is the (hemi-)sphere of incoming directions ω_i, and the measure $\mathrm{d}\omega^\perp = \cos\theta\ \mathrm{d}\omega$ is the projected solid angle measure, which includes a foreshortening factor to account for Lambert's law. The term $f_r(.)$ is the *bidirectional scattering distribution function* (BSDF) and characterises how incoming irradiance is converted to outgoing radiance. This is responsible for the look of surface materials, such as texture or glossiness.

The integral form of the RTE

With an integrating factor (transmittance) and boundary conditions (surface transport) we are now ready to convert the RTE to integral form. Putting everything together, the result is

$$\begin{aligned} L(\mathbf{x},\omega) = {} & T(\mathbf{x},\mathbf{y})\overbrace{\left(L_e(\mathbf{y},\omega) + \int_\Omega f_r(\omega_i,\mathbf{y},\omega)L(\mathbf{y},\omega_i)\mathrm{d}\omega_i^\perp\right)}^{\text{contribution from the point } \mathbf{y} \text{ on surface}} \\ & + \int_0^d T(\mathbf{x},\mathbf{z})\underbrace{\left(\mu_e(\mathbf{z})L_e(\mathbf{z},\omega) + \mu_s(\mathbf{z})\int_\Omega f_s(\omega\cdot\omega_i)L(\mathbf{z},\omega_i)\mathrm{d}\omega_i\right)}_{\text{contribution from any point } \mathbf{z} \text{ at distance } t \text{ in volume}}\mathrm{d}t. \end{aligned} \tag{28}$$

Figure 1 visualises all terms in the integral form of the RTE. Since this is somewhat bulky, we can simplify the notation by dropping a few parameters and introducing a linear transport operator $\mathbf{T}$, which represents the scattering at either the point on the surface $\mathbf{y}$ or a point $\mathbf{z}$ in the volume, as introduced by Veach [1998]:

$$L = L_e + \mathbf{T}L. \tag{29}$$

1.2 The path space

With the physical foundation and mathematical tools at hand, we can now determine how much light flows along a single transport path. Explicitly unrolling the recursion in equation (29) for an example path with five vertices

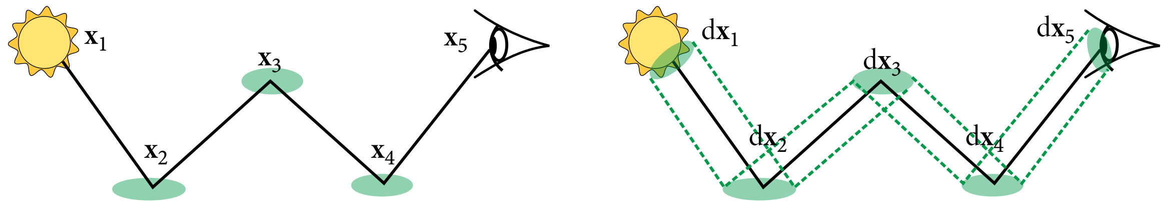


Figure 2: *A light transport path is represented as a list of vertices (left). Each vertex comes with an integration domain in vertex area measure, as indicated by* $\mathrm{d}\mathbf{x}_i$ *in the right image. The full integration domain consists of the product of all these infinitesimal surface patches, the product vertex area measure as indicated by the dashed tube around the path.*

(cf. figure 2) shows that we need to transport the emitted radiance four times. In general

$$L = \mathbf{T}^{k-1} L_e \tag{30}$$

for k path vertices. Substituting the original terms from equation (28) instead of the transport operator shorthand, this results in the "flat view" of the rendering equation, the measurement contribution function as defined by Veach [1998]:

$$f(\mathbf{X}) = L_e(\mathbf{x}_1)T(\mathbf{x}_1, \mathbf{x}_2)G(\mathbf{x}_1, \mathbf{x}_2) \cdot \left(\prod_{i=2}^{k-1} f_x(\mathbf{x}_i)T(\mathbf{x}_i, \mathbf{x}_{i+1})G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) \cdot W(\mathbf{x}_k). \tag{31}$$

This formulation contains two new terms: W is the sensor responsivity function. It models how the sensor reacts to light. While this is mostly used to un-do the vignetting caused by most camera models, it may also model a spectral response corresponding to the colour filter array of the sensor.

The other term, G, is called the geometry term and appears here as the Jacobian determinant from projected solid angle $\mathrm{d}\omega^\perp$ to vertex area measure $\mathrm{d}\mathbf{x}$, such that we can integrate $f(\mathbf{X})$ over the path space in product vertex area measure. This measure space is useful because it is agnostic of the transport direction and allows for easy inclusion of *next event estimation* samples which directly sample transport points on surfaces instead of outgoing directions. The geometry term G also contains the mutual visibility $V(\mathbf{x}, \mathbf{y})$ between the points. More precisely

$$G(\mathbf{x}, \mathbf{y}) = V(\mathbf{x}, \mathbf{y}) \frac{D(\mathbf{x}, \mathbf{y})D(\mathbf{y}, \mathbf{x})}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{32}$$

$$D(\mathbf{x}, \mathbf{y}) = \begin{cases} |n(\mathbf{x}) \cdot \omega_{\mathbf{x}\to\mathbf{y}}| & \text{if } \mathbf{x} \text{ is on a surface,} \\ 1 & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \tag{33}$$

$$L_e(\mathbf{x}, \mathbf{y}) = \begin{cases} L_e(\mathbf{x}, \omega_{\mathbf{x}\to\mathbf{y}}) & \text{if } \mathbf{x} \text{ is on a surface,} \\ \mu_e(\mathbf{x})L_e(\mathbf{x}, \omega_{\mathbf{x}\to\mathbf{y}}) & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \tag{34}$$

$$f_x(\mathbf{x}) = \begin{cases} f_r(\mathbf{x}) & \text{if } \mathbf{x} \text{ is on a surface,} \\ \mu_s(\mathbf{x})f_s(\mathbf{x}) & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \tag{35}$$

The scattering function f_x is generalised to express the BSDF f_r on surfaces as well as the scattering collision coefficient and phase function $\mu_s(\mathbf{x}) \cdot f_s(\omega, \omega_i)$ inside media. Note that for brevity we dropped the dependency of the scattering functions on the incoming and outgoing directions in equation (31) and equation (35).

To model a camera as a measurement apparatus, collecting photons per time incident on the pixel area from a certain solid angle, i.e. to compute radiant power, all we need to do is integrate the measurement contribution function equation (31) over all vertex areas $\mathrm{d}\mathbf{x}$ associated with a path. For instance for the path configuration in figure 2, we get

$$\int \int \int \int \int f(\mathbf{X}) \, \mathrm{d}\mathbf{x}_1 \, \mathrm{d}\mathbf{x}_2 \, \mathrm{d}\mathbf{x}_3 \, \mathrm{d}\mathbf{x}_4 \, \mathrm{d}\mathbf{x}_5, \tag{36}$$

which means we need to integrate over five individual surface patches in square meters (this would be cubic meters for vertices in volumes). As a shorthand, we define $\mathrm{d}\mathbf{X}_5$ as a product measure for paths with five vertices. Now, light is not restricted to be transported only by paths of length five, but in general we need to sum up contributions of all path lengths, for instance by explicitly expanding the *Neumann series*

$$L = L_e + \mathbf{T}L_e + \mathbf{T}^2 L_e + \cdots . \tag{37}$$

To be able to write one single integral for transport paths of any length, we also define $\mathrm{d}\mathbf{X}$ to signify the union of all $\mathrm{d}\mathbf{X}_k = \prod_{i=1}^{k} \mathrm{d}\mathbf{x}$ with $k \geq 2$. This finally enables us to unify lighting computations in one single mathematical framework (be it surfaces, sub-surface scattering, or volume contributions). Intuitively, we track all possible paths that photons could take from all light sources via multiple interactions with objects and their materials, into the camera lens. These photons are then "counted" on the sensor.

The sampling space we just defined is called the *path space* $\mathcal{P}$ and contains all possible transport paths $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\} \in \mathcal{P}$, which are lists of k path vertices $\mathbf{x}$ (see figure 2). To compute the color I_p of a pixel p, we simply integrate over this space, weighted by a pixel filter $h_p(\mathbf{X})$, such as for instance the one derived by Harris [1978]. Note that sometimes the pixel filter $h_p(\mathbf{X})$ is folded into the camera responsivity $W(\mathbf{x}_k)$. In summary, we finally arrive at the path integral for global illumination:

$$\boxed{I_p = \int_{\mathcal{P}} h_p(\mathbf{X}) \cdot f(\mathbf{X})\, \mathrm{d}\mathbf{X}}. \tag{38}$$

Now the only remaining challenge is to efficiently sample good transport paths $\mathbf{X}$ to numerically evaluate the integral.

1.3 The Monte Carlo method

Equation (38), i.e. integrating the measurement contribution function over path space, presents a high dimensional integration problem. Depending on path length, the integral can easily contain hundreds of dimensions. This makes many popular integration schemes perform poorly (for instance quadrature rules would yield exponential complexity in the number of dimensions).

The method of choice due to its behaviour for high dimensionality is the Monte Carlo method (see for instance Ermakow [1975] or Sobol' [1994] for an introduction).

The main idea is to make use of the definition of the expected value of a continuous random variable x distributed with a *probability distribution function* (PDF) $p(x)$ to solve the integral

$$\mathbb{E}(x) = \int x \cdot p(x)\, \mathrm{d}x \tag{39}$$

by drawing a few random trials from x instead of solving the integral analytically. This results in a noisy Monte Carlo estimator

$$\hat{x} = \frac{1}{N} \sum_{i=1}^{N} x \approx \mathbb{E}(x). \tag{40}$$

Applying this same principle to the path space integral, we need to divide out the probability distribution function $p(.)$ from the integrand $f(\mathbf{X})$ to match the definition of the expected value in equation (39):

$$I_p = \int_{\mathcal{P}} h_p(\mathbf{X}) \cdot f(\mathbf{X})\, \mathrm{d}\mathbf{X} \approx \hat{I}_p = \frac{1}{N} \sum_{i=1}^{N} \frac{h_p(\mathbf{X}) \cdot f(\mathbf{X})}{p(\mathbf{X})}. \tag{41}$$

The crucial difficulty in designing good estimators is now to find an appropriate PDF $p(\mathbf{X})$ which minimises the integration error of this approximation. Such error manifests itself mostly as variance

$$Var(\hat{I}_p) = \frac{1}{N} \int \left(\frac{h_p(\mathbf{X}) f(\mathbf{X})}{p(\mathbf{X})} - I_p \right)^2 p(\mathbf{X})\, \mathrm{d}\mathbf{X}. \tag{42}$$

Mostly here means that we assume all employed algorithms will be unbiased, such that the error is spread around the correct mean and the deviation will be only random noise, decreasing with higher sample count N. Equation (42) shows that the primary estimator $h \cdot f / p$ needs to be close to I_p to reduce variance. In practice, this can be achieved by variance reduction techniques, such as importance sampling. This tries to choose $p(\mathbf{X})$ to follow $f(\mathbf{X})$ as closely as possible (of course the PDF will be normalised while f is not). This goal is all but trivial to achieve in general for the high dimensional path space.

1.4 Colour formation in a renderer

While the first section introduced the path tracing framework and the transport equations for a full path, these were written without dependency on wavelength λ. Actually, most of the terms such as BSDF, transmittance, emission and sensor responsivity have spectral equivalents and depend on wavelength. Modelling this wavelength dependency as closely as possible to physical reality results in much improved fidelity, as well as better importance sampling.

Colour can come into a rendering by simulating three discrete wavelengths, as typically done in standard RGB transport. Even when simulating the wavelength domain directly, we usually assume only *elastic* scattering happens. This means that the energy of a photon remains the same after an event and only depends on the wavelength. In particular a photon can only be absorbed or scattered into a different direction, but no inter-wavelength transport is considered (such as it would be required to accurately model fluorescence or phosphorescence).

In the most simple case, colour is treated completely detached from the path sampling. This means the path is constructed and colour is added "after the fact", if you will, multiplying colour dependent components such as BSDF, transmittance, etc. Ignoring cases where path creation has a strong dependency on wavelength (we will get to that in a moment), this boils down to multiplying chromatic factors, for instance for a simple path:

$$f(\mathbf{X}) = L_e(\lambda) \cdot G_1 \cdot f_r(\lambda) \cdot G_2 \cdot W(\lambda). \tag{43}$$

This will be integrated in the frame buffer, to yield tristimulus colour:

$$X = \int_{380..830nm} f(\lambda) \cdot \bar{x}(\lambda)\, \mathrm{d}\lambda, \tag{44}$$

where X is the first channel of the CIE XYZ tristimulus colour space and $\bar{x}(\lambda)$ is the normalised colour matching function for this channel. The Y and Z channels are computed analogously. See for instance Fairman et al. [1998] for more information on the colour matching functions.

What happens in RGB transport, when using the CIE RGB primaries, this equation will only be evaluated for three distinct wavelengths of 700 nm (red), 546.1 nm (green) and 435.8 nm (blue). It is clear that a lot of information between these wavelengths is lost because it is never evaluated. On the other hand, the transport for these wavelengths is evaluated physically correctly. The tristimulus values which end up in the frame buffer are then directly

$$R = f(\lambda = 700nm), \quad G = f(\lambda = 546.1nm), \quad B = f(\lambda = 435.8nm). \tag{45}$$

In general, using any other RGB space to perform the multiplications in equation (43) and replacing the integral in equation (44) like in equation (45) is wrong and will yield non-physical transport. This is especially apparent for indirect illumination. Agland [2014] performed extensive comparisons on the impact of the rendering colour space.

1.4.1 Colour reproduction

With spectral rendering, precise colour reproduction is simple. All relevant formulas are collected in the fundamental book by Wyszecki and Stiles [2000]. Just model the light source emission, the surface reflection, and the camera responsivity with measured spectral data and the result will be correct. Modeling the spectral camera response will also give a render directly in camera RGB space rather than XYZ. This means that the render will even show the same metamerism as the live footage. There are, however, a few subtleties to keep in mind.

As often, physical plausibility has advantages and downsides. A possible downside, especially when rendering cartoons, may be that energy conservation poses a limit on colour saturation and brightness of a surface. This has been recognised early on by Schrödinger [1919] (and who are we to argue with that).

The issue is that energy conservation dictates that, in the absence of fluorescence, no wavelength λ may result in more reflected than incoming energy:

$$\int_{\Omega} f_r(\mathbf{x}, \omega, \omega_i, \lambda)\, \mathrm{d}\omega_i^{\perp} \leq 1 \quad \forall \lambda. \tag{46}$$

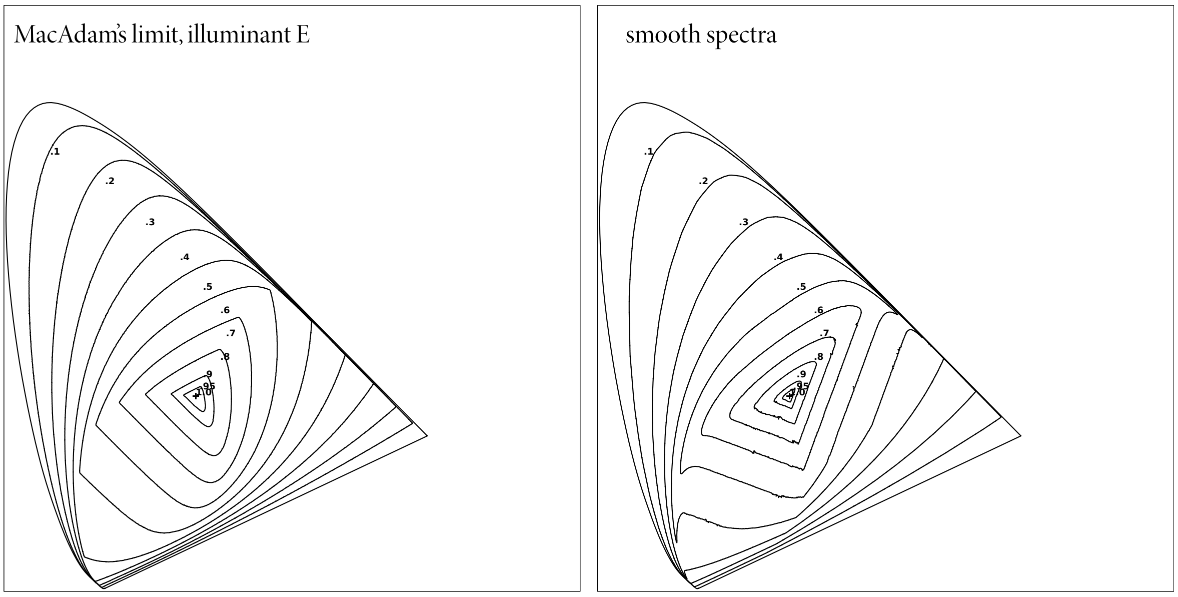


Figure 3: *The maximal gamuts of surface reflection, reproduced after Meng et al. [2015]. The graphs show the maximum brightness $(X + Y + Z)/3$ of a surface colour (for instance a diffuse albedo such that $X = Y = Z = 1$ would be the peak at 1 in the graph), as an iso line graph. The coordinate system as seen from the top is the standard space of chromaticities, i.e. $x = X/(X + Y + Z)$ and $y = Y/(X + Y + Z)$. Left: the theoretical maximum which can possibly be achieved using step functions as spectra. Right: a slightly smaller gamut that can be achieved using more natural, smooth spectra.*

For a diffuse BSDF, $f_r = \rho(\lambda)/\pi$, where $\rho(\lambda)$ is the albedo, this means that $\rho(\lambda) \leq 1$ for all wavelengths λ. Now, the total brightness of the surface as seen in an RGB image has something to do with the XYZ brightness, i.e. $X + Y + Z$, which is essentially the integral of $\rho(\lambda)$. Naturally, a more saturated colour means a more peaky spectral shape, which forces the integral to diminish since the maximum cannot be increased.

Figure 3, reproduced after Meng et al. [2015], shows the limits on brightness of a surface colour $(X+Y+Z)/3$, depending on colour saturation. As the chromaticity of the colour moves towards the edge of the spectral locus, the maximum achievable brightness becomes dimmer. The gamut shown on the left is the one derived by MacAdam [1935], who gave a constructive proof as to which spectra will yield the highest possible brightness for a given chromaticity. The one on the right is derived by Meng et al. [2015] and uses more natural smooth spectra. These lead to more believable indirect lighting, since the shape is usually closer to most reflectances encountered in the wild.

In the future, to add even more realism, renderers may require fluorescence to exceed the MacAdam gamut, similar to Couzin [2007]. Note that this is only an issue when such bright and saturated colours are required. For the more regular case that realistic surface reflection needs to be reproduced, this limit of energy conservation poses a natural restriction on the look of the materials. This automatically avoids unrealistically bright and glowing surfaces. Together with smooth spectra, this results in much more life-like indirect lighting than using naive RGB transport. Note that this is mostly a problem for wide gamut input colour spaces: when staying within Rec709 primaries all colours can be represented by valid reflectances, albeit some of them will be boxy (cf. Jakob and Hanika [2019]).

1.4.2 Where to get input spectra from

We can get spectral definitions for some light sources or cameras from the manufacturers. Also some special materials, such as for instance spectral absorption of melanin (for hair) and hemoglobin (for skin), can be readily found in text books. Even for these it is sometimes useful to be able to overrule or modulate them by artist-drawn textures. Since these are usually working in RGB, there is a need to convert tristimulus data to full continuous spectra.

Early work by MacAdam [1935] facilitates this, but with the limitation that the resulting colours will always be as bright as possible and thus box functions in shape. Since natural reflection spectra are usually smooth, this results in unnatural looking indirect lighting.

Smits [1999] devised a method to upsample RGB values to spectra, taking into account smoothness and optimising the process to try and achieve energy conservation too. Depending on the input tristimulus coordinate, it may not be possible to meet both goals: chromaticity and energy conservation. Also, this method only works within a certain RGB working space, not for the whole gamut of visible colours. Meng et al. [2015] recognise this and separate the process into two steps: first, the colour from tristimulus values is upsampled, disregarding energy conservation. Secondly, a gamut mapping step is performed that enforces energy conservation in case the input colour was too saturated and bright for a physically plausible reflectance value. This is not needed in case a light source emission is upsampled from RGB values. This approach ensures a surface lit by illuminant E will look the same when using the RGB reflectances and the upsampled spectrum, when observed with the CIE XYZ colour matching functions $\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)$.

There is a refreshed interest in accurate and fast spectral upsampling: Otsu et al. [2018] employ a clustered PCA to create spectra that are similar to a certain input set and clamp to $[0, 1]$ if needed. Jakob and Hanika [2019] present a particularly fast method to upsample spectra, and Peters et al. [2019] provide a sophisticated method with scalable accuracy. The last two approaches get away without look up tables during render time, which makes them particularly interesting for memory bound scenarios.

1.4.3 Colour noise

As mentioned above, introducing a randomly sampled wavelength λ into the path tracing process introduces noise. Fortunately, natural reflectance spectra are smooth, and also forced to be this during a potential upsampling step from tristimulus texture input. It is thus an effective strategy to use stratified samples in the wavelength domain to resolve colour.

Wilkie et al. [2014] do this in combination with efficient path reuse: the path construction is still performed with one main wavelength, and a set of 3 stratified wavelengths are evaluated alongside with it. The final contribution is weighted using multiple importance sampling (MIS), resulting in a much lower variance picture.

The evaluation of the PDF and wavelength-dependent BSDF can be performed in SSE, evaluating four wavelengths in four lanes in one instruction. Note that this method requires precise computation of PDFS (that is, a stochastically evaluated or approximate PDF may lead to problems). Due to its usefulness for noise reduction it has found adoption in production (in WETA DIGITAL's *Manuka*), but may be limiting in some contexts that do not already depend on multiple importance sampling.

1.4.4 Importance sampling

While at first sight it may seem path construction can be performed independently of wavelength, there are a few important special cases.

The first is obviously chromatic dispersion in dielectrics, causing the prominent rainbow-like colours in caustics, for instance under a glass of water on a table in the sun. This is one obvious effect that is hard to model in an RGB-based rendering system. On the other hand, shots with such effects are relatively rare. This is even more so because usually the visually rich materials in VFX have fine details such as scratches, grease stains on glass, or dirt particles scattering the light under water. All this blurs or masks away such subtle dispersion effects most of the time.

There are some scattering models which include a spectral shape of the lobe. This includes diffraction at surface points as well as Rayleigh scattering in the atmosphere. Using spectral sampling, it is easy to incorporate such advanced models into a render.

The most important case, however, is chromatic extinction in participating media. That is, the extinction coefficient $\mu_t(\mathbf{x}, \lambda)$ depends on the wavelength. This governs the transmittance term equation (25) which is simply $\exp(-\mu_t(\lambda) \cdot d)$ for homogeneous media. The mean free path in the medium $1/\mu_t$ depends on the wavelength in chromatic media, resulting in very different importance sampling strategies for red vs. blue photons.

This is important for instance when using fully ray traced sub-surface scattering in skin: skin has a particular look that scatters red light farther than blue light. This is the reason why black and white portrait photography looks smoother with a red filter.

Radiometric spectral				Photometric			
name		unit	symbol	name		unit	symbol
Radiance		$W/(m^2 \cdot sr \cdot m)$	L_λ	Luminance	*nit*	$nt = lm/(m^2 \cdot sr)$	L_v
Irradiance		$W/(m^2 \cdot m)$	E_λ	Illuminance	*lux*	$lx = lm/m^2$	E_v
Radiosity		$W/(m^2 \cdot m)$	J_λ	Luminosity	*lux*	$lx = lm/m^2$	J_v
Radiant emittance		$W/(m^2 \cdot m)$	M_λ	Luminous emittance	*lux*	$lx = lm/m^2$	M_v
Radiant intensity		$W/(sr \cdot m)$	I_λ	Luminous intensity	*candela*	$cd = lm/sr$	I_v
Radiant power	*watt*	W/m	Φ_λ	Luminous power	*lumen*	lm	Φ_v
Radiant energy	*joule*	$J/m = W \cdot s/m$	Q_λ	Luminous energy	*talbot*	$Tb = lm \cdot s$	Q_v

Table 1: *Correspondence between radiometric and photometric units. We abbreviate the unit for luminous energy talbot as Tb instead of the also common T to avoid confusion with the unit for magnetic flux tesla. We also use the convention of subscripting photometric quantities with v (for visual), radiometric quantities with e (for energetic) and spectral radiometric quantities with λ. this follows the recommendations in documents such as USAS and ASME [1967].*

The domain of distance sampling is fairly extreme: $[0, \infty)$. This means that scattering vertices will be sampled far apart when importance sampling the transmittance for different wavelengths. In some cases, when one wavelength does not interact with the medium at all, this leads to infinite variance, as recognised by [Raab et al., 2008, Sec. 3.2].

This application of spectral importance sampling is often the most important one, since it is very hard to perform principled importance sampling which can be combined in a flexible way with generic sampling strategies in RGB transport (such as combination with equi-angular sampling or sampling distances by scattering coefficient instead of extinction).

1.4.5 Radiometry vs. Photometry

Radiometric quantities (such as watts for flux or watts/square meter/steradian for radiance) are great to work with during light transport, since they allow a 1:1 mapping to the equations we find in physics books.

For a lighter, however, it may be more intuitive to work with photometric quantities. These account for the fact that different colours appear to be of different brightness to a human observer. To be precise, a spectral power distribution can be converted from radiometric quantities to photometric ones by weighting by a luminosity function. Usually the photopic, daytime brightness function of the CIE is used. This allows us to express radiant power not in watts but as *lumen*, which is then called luminous power, for instance. For all radiometric quantities, there are equivalent photometric ones (cf. Tab. 1). Designing user interfaces for lighters around this notion allows them to change the colour of a light source while maintaining the perceived brightness in a principled way.

As said earlier, when dealing with spectral light sources, the photopic luminosity function $\bar{y}(\lambda)$ is used: this is the result of a series of experiments and tabulations first published by the CIE in 1924 (the function was called $V(\lambda)$ at the time) and then included in the color matching functions for the standard 2 degree colorimetric observer, published in 1931.

At this point we have enough information to write equations correlating radiometric quantities to their corresponding photometric ones: given a radiometric spectral quantity $X_\lambda(\dots, \lambda)$ the corresponding photometric quantity $X_v(\dots)$ is simply obtained integrating X_λ against $K_m \cdot \bar{y}(\lambda)$ where K_m is a scaling constant about equal [4] to 683:

$$X_v(\dots) = K_m \int X_\lambda(\dots, \lambda)\bar{y}(\lambda)d\lambda.$$

For example, given spectral radiant power $\Phi_\lambda(\lambda)$, the corresponding luminous power Φ_v is

$$\Phi_v = K_m \int \Phi_\lambda(\lambda)\bar{y}(\lambda)d\lambda.$$

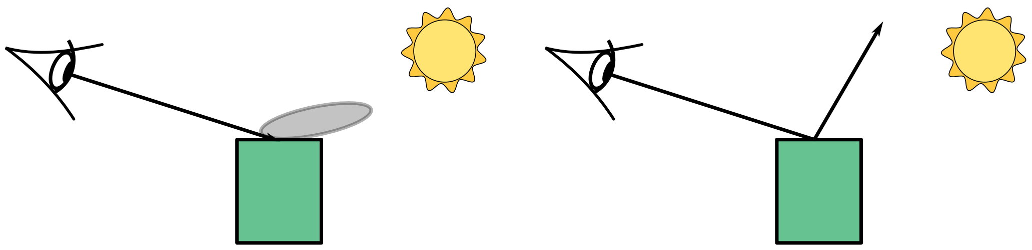
[4] The scaling constant K_m is actually closer to 683.002, because the value of $\bar{y}$ is about 0.999 998 at 555.016 nm, but the value of 683 can safely be used for all practical applications, see CIE [1996], Wyszecki and Stiles [2000]

1.5 Path construction techniques

This section will give a quick run down on a few basic path construction techniques and the related problems. The examples here are generic and simplified, in practical production scenarios they will be aggravated by difficult geometry, complex materials, and combinations of many lighting effects.

1.5.1 Path tracing

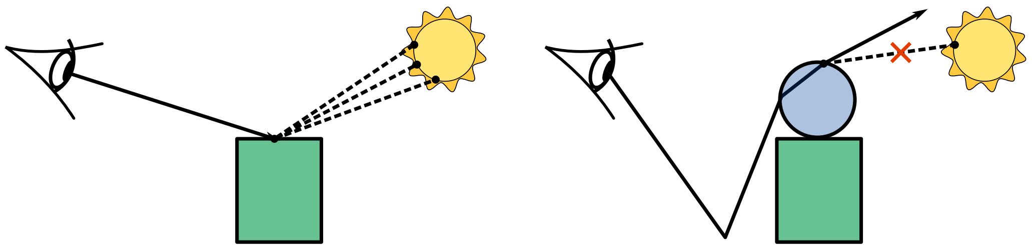
In the most simple incarnation as described by Kajiya [1986], path construction proceeds by sampling path vertices iteratively by successively extending the path starting at the eye or camera sensor. This involves sampling a pixel, a point on the camera aperture, and tracing a ray to the first intersection with the scene. At this point, the BSDF is queried to generate a good outgoing direction given only the incoming ray (left):



This local decision is not necessarily a good idea globally, however. In the example in the right image above, the generated outgoing direction misses the light source. This is especially problematic for small and far away light sources, subtending a small solid angle as seen from the shading point.

1.5.2 Next event estimation

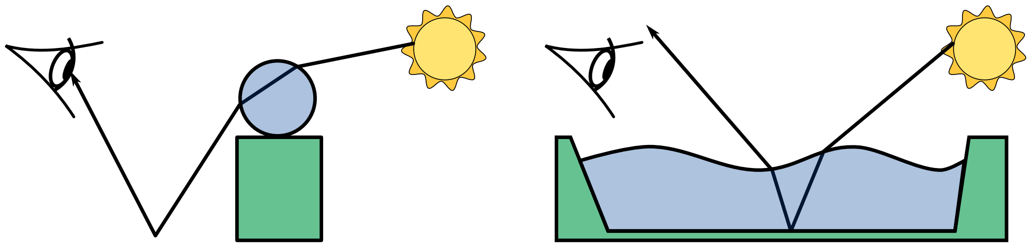
The problem outlined in the previous section is addressed by a technique called *next event estimation* (NEE). In this context, a path is directly connected to the light sources by explicitly sampling the surface area of the emitter in vertex area measure instead of sampling an outgoing direction in projected solid angle measure. This has been known in neutron transport for a long time and coined *next event estimation* by Coveyou et al. [1967]. Connections can be performed multiple times from the same vertex to increase sampling efficiency for long paths with difficult illumination (left):



This technique, however, disregards the BSDF. In extreme cases this can result in no throughput at all, if the BSDF contains a Dirac delta, such as smooth specular materials in the example on the right. The problem persists with glossy materials, too, and gradually becomes less of an issue for diffuse materials. Unfortunately, even those will have sub optimal importance sampling when using straight uniform area sampling as opposed to more advanced mappings such as for instance the one proposed by Ureña et al. [2013].

1.5.3 Light tracing

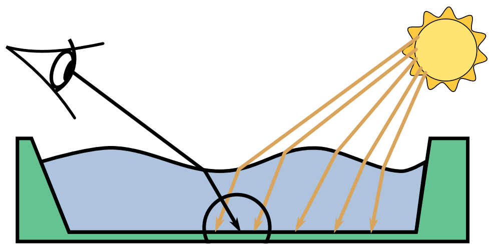
Another technique devised by Arvo [1986] to solve the problem in the last section is called *light tracing* as opposed to path tracing, because the random walk starts at the emitters and performs next event estimation to the eye point (left):



Unfortunately, it is very easy to construct symmetric fail cases for this type of next event estimation by putting a smooth specular material between the eye point and the diffuse surface (see right image). Light tracing can be combined via multiple importance sampling with all previously mentioned techniques to generate a more robust composite estimator. The example in the image above is however one that cannot be rendered efficiently by bidirectional path tracing. This is because the diffuse event on the bottom of the pool is encapsulated by specular events on both sides, a so called SDS situation (from specular-diffuse-specular, this system for classifying path structures was introduced in [Heckbert, 1990]). Again, this is a problem even if the BSDF are not perfectly smooth specular, sampling efficiency will smoothly degrade as the BSDF tends to a Dirac delta.

1.5.4 Photon mapping-based approaches

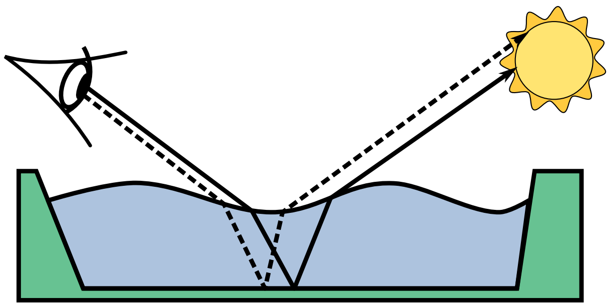
To render such SDS scenarios, one can employ path caches by means of photon mapping (see the works of Georgiev et al. [2012], Hachisuka et al. [2012], Jensen [1996]). Techniques employing this are usually two-pass methods, first tracing fractional light packets (colloquially called photons in computer graphics) or gather points from the light or the eye, respectively. In this example image, "photons" are traced from the light, as indicated by the yellow paths:



It is then possible in a second pass, here performed from the eye (indicated by the black rays), to collect these photons via kernel estimation (indicated by the black circle). Since many "photons" are traced in the first pass, it is likely to intersect them in the second pass, facilitating efficient reuse of computation. Drawbacks of this method include that photons aren't distributed according to importance from the eye and can be wasteful without further refinement of the method, such as the one proposed by Gruson et al. [2016]. The kernel estimation is not always simple and can lead to visible bias near complex geometry, which is unfortunately quite important for production scenes. This is especially apparent on hair and fur.

1.5.5 Markov chain-based methods

An unbiased alternative is the family of Markov chain-based methods. These include Metropolis light transport as introduced by Veach and Guibas [1997], primary sample space Metropolis light transport by Kelemen et al. [2002], and Hamiltonian Monte Carlo as introduced to graphics by Li et al. [2015]. On a high level, these methods are based on a Markov chain that holds a current state (indicated as the dashed path in the figure):



This path is then mutated by iteratively proposing tentative new paths which are drawn from some transition probability distribution. These mutations can be modelled to explicitly handle a certain class of important transport, such as exploring specular manifolds (see Jakob and Marschner [2012]). Markov chains have traditionally suffered from temporal flickering. This is because we need to strike a balance between global discovery of important lighting effects (disconnected islands in path space, potentially with different dimensionality than the current path) and local exploration (small mutations). Plain Monte Carlo is good at the first task, especially when combined with stratified sampling such as when using the randomly scrambled Halton sequence. Markov chain-based methods are usually very good at the second task, but fail to stratify the samples globally, resulting in temporal instability.

1.6 Conclusion

This introductory section summarised the most essential and basic concepts of light transport theory, the related equations, a few thoughts on colour reproduction which is essential to match live footage in production, as well as the most common path space sampling techniques.

References

Steve Agland. 2014. CG Rendering and ACES. http://nbviewer.ipython.org/gist/sagland/3c791e79353673fd24fa.

Marco Ament, Christoph Bergmann, and Daniel Weiskopf. 2014. Refractive Radiative Transfer Equation. *ACM Trans. on Graphics* 33, 2 (April 2014), 17:1–17:22. https://doi.org/10.1145/2557605

James Arvo. 1986. Backward Ray Tracing. In *SIGGRAPH Course Notes*. 259–263.

James Arvo. 1993. Transfer equations in global illumination. In *SIGGRAPH Course Notes*.

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. *A radiative transfer framework for non-exponential media*. Technical Report.

Subrahmanyan Chandrasekar. 1960. *Radiative Transfer*. Dover Publications Inc. ISBN 0-486-60590-6.

CIE. 1996. *The Basis of Physical Photometry*. Commission Internationale de l'Éclairage, CIE Central Bureau.

Dennis Couzin. 2007. Optimal fluorescent colors. *Color Research & Application* 32, 2 (2007), 85–91.

R. R. Coveyou, V. R. Cain, and K. J. Yost. 1967. Adjoint and Importance in Monte Carlo Application. *Nuclear Science and Engineering* 27, 2 (1967), 219–234. https://doi.org/10.13182/NSE67-A18262

Eugene d'Eon. 2018. A reciprocal formulation of non-exponential radiative transfer. 1: Sketch and motivation. *ArXiv e-prints* (March 2018). arXiv:physics.comp-ph/1803.03259

Sergej Mikhailovich Ermakow. 1975. *Die Monte Carlo Methode und verwandte Fragen*. VEB Deutscher Verlag der Wissenschaften.

Hugh Fairman, Michael Brill, and Henry Hemmendinger. 1998. How the CIE 1931 color-matching functions were derived from Wright-Guild data. *Color Research and Application* 22, 1 (1998), 11–23.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 31, 6 (2012), 192:1–192:10.

Andrew S. Glassner. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann.

Adrien Gruson, Mickaël Ribardière, Martin Šik, Jiří Vorba, Rémi Cozot, Kadi Bouatouch, and Jaroslav Křivánek. 2016. A Spatial Target Function for Metropolis Photon Tracing. *ACM Trans. on Graphics* 36, 1 (Nov. 2016), 4:1–4:13. https://doi.org/10.1145/2963097

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 31, 6 (2012), 191:1–191:10.

Frederic J. Harris. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE* 66, 1 (1978), 51–83.

Paul S. Heckbert. 1990. Adaptive Radiosity Textures for Bidirectional Ray Tracing. *SIGGRAPH Comput. Graph.* 24, 4 (Sept. 1990), 145–154. https://doi.org/10.1145/97880.97895

L. Henyey and J. Greenstein. 1941. Diffuse radiation in the Galaxy. *Astrophysical Journal* 93 (1941), 70–83.

Matthias Hullin, Johannes Hanika, Boris Ajdin, Jan Kautz, Hans-Peter Seidel, and Hendrik Lensch. 2010. Acquisition and Analysis of Bispectral Bidirectional Reflectance and Reradiation Distribution Functions. *Transactions on Graphics (Proceedings of SIGGRAPH)* 29, 4 (2010), 1–7.

Wenzel Jakob and Johannes Hanika. 2019. A Low-Dimensional Function Space for Efficient Spectral Upsampling. *Computer Graphics Forum (Proceedings of Eurographics)* 38, 2 (March 2019).

Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 58:1–58:13.

Wenzel Jakob, Jonathan T. Moon, Adam Arbree, Kavita Bala, and Steve Marschner. 2010. A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 29, 10 (July 2010), 53:1–53:13. https://doi.org/10.1145/1778765.1778790

Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A Radiative Transfer Framework for Spatially-Correlated Materials. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 37, 4 (2018).

Adrian Jarabo and Diego Gutierrez. 2016. Bidirectional Rendering of Polarized Light Transport. In *Proceedings of CEIG '16*.

Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. 2014. A Framework for Transient Rendering. *ACM Trans. on Graphics* 33, 6 (Nov. 2014), 177:1–177:10. https://doi.org/10.1145/2661229.2661251

Wojciech Jarosz. 2008. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. Ph.D. Dissertation. UC San Diego, La Jolla, CA, USA. Advisor(s) Henrik Wann Jensen and Matthias Zwicker.

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Proc. Eurographics Workshop on Rendering*. 21–30.

James T. Kajiya. 1986. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)* (1986), 143–150.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum* 21, 3 (2002), 531–540.

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 34, 6 (Nov. 2015), 209:1–209:13.

David L. MacAdam. 1935. Maximum Visual Efficiency of Colored Materials. *Journal of the Optical Society of America* 25, 11 (1935), 361–367.

Johannes Meng, Florian Simon, Johannes Hanika, and Carsten Dachsbacher. 2015. Physically Meaningful Rendering using Tristimulus Colours. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34, 4 (June 2015), 31–40.

Jan Novák. 2014. *Efficient Many-Light Rendering of Scenes with Participating Media*. Ph.D. Dissertation. Karlsruhe Institute of Technology.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum (Eurographics State of the Art Reports)* 37, 2 (2018), 1–26.

H. Otsu, M. Yamamoto, and T. Hachisuka. 2018. Reproducing Spectral Reflectances From Tristimulus Colours. *Computer Graphics Forum* 37, 6 (2018), 370–381. https://doi.org/10.1111/cgf.13332

Christoph Peters, Sebastian Merzbach, Johannes Hanika, and Carsten Dachsbacher. 2019. Using Moments to Represent Bounded Signals for Spectral Rendering. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 38, 4, Article 136 (2019), 14 pages. https://doi.org/10.1145/3306346.3322964

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2017. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc.

Matthias Raab, Daniel Seibert, and Alexander Keller. 2008. Unbiased Global Illumination with Participating Media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. 591–606.

Erwin Schrödinger. 1919. Theorie der Pigmente größter Leuchtkraft. *Annalen der Physik* 367, 15 (1919), 603–622.

Brian Smits. 1999. An RGB-to-spectrum conversion for reflectances. *Journal of Graphics Tools* 4, 4 (1999), 11–22.

Ilya Sobol'. 1994. *A Primer for the Monte Carlo Method*. CRC Press.

John Strutt. 1871. On the light from the sky, its polarization and colour. *Philos. Mag.* 41, 4 (1871), 107–120,274–279.

Kip Thorne. 2014. *The Science of Interstellar*. W. W. Norton & Company.

Carlos Ureña, Marcos Fajardo, and Alan King. 2013. An Area-preserving Parametrization for Spherical Rectangles. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2013), 59–66. https://doi.org/10.1111/cgf.12151

USAS and ASME. 1967. *USA Standard Letter Symbols for Illuminating Engineering*. United States of America Standards Institute.

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. *Proc. SIGGRAPH* (1997), 65–76.

Sebastian Werner, Zdravko Velinov, Wenzel Jakob, and Matthias B. Hullin. 2017. Scratch Iridescence: Wave-optical Rendering of Diffractive Surface Structure. *ACM Trans. on Graphics* 36, 6 (Nov. 2017), 207:1–207:14. https://doi.org/10.1145/3130800.3130840

Alexander Wilkie, Sehera Nawaz, Marc Droske, Andrea Weidlich, and Johannes Hanika. 2014. Hero Wavelength Spectral Sampling. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 33, 4 (July 2014), 123–131.

G. Wyszecki and W. S. Stiles. 2000. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons.

2 Monte Carlo integration

Luca Fascione, *Weta Digital*
Johannes Hanika, *Weta Digital*

2.1 Introduction: a holistic look at integration theory

Path tracing algorithms can be seen as specialized algorithms carefully designed to compute a very high dimensional integral of a function with a relatively specific structure. Generations of researchers, following Jim Kajiya's seminal article introducing the rendering equation to graphics [Kajiya, 1986], have used various kinds of Monte Carlo methods to estimate approximate solutions to the rendering equation, sometimes without asking too many questions as to why exactly a method or another was actually chosen.

In what follows we showcase how there are a few possibilities for numerically estimating the integral of a given function f : if a closed formula can be used, using arguments based from an analytical application of Riemann or Lebesgue integration, the error will be only a function of the precision with which the closed form expression of the analytical integral can be evaluated.

However in the common case analytical means are not applicable, for example because a closed form for the integral is not known, or maybe because it is f to not have an analytic form in the first place, and other numerical approaches must be employed. One kind of avenue is based on approximating sections of f with functions for which closed form integration is available and well understood and controlled, such as the Newton-Cotes formulas, whereas the other avenue is to use Monte Carlo integration, which apparently starts with a lesser degree of control on the estimation error ε_n.

Newton-Cotes becomes hard if the integration domain Ω is complicated and/or high dimensional: first off it might require sampling f an enormous amount of times (this is the so-called *curse of dimensionality*), and secondarily the control on error depends on how well the Lagrange interpolation approximates f.

On the other hand Monte Carlo has no specific issues dealing with integration domains of awkward shapes and/or high dimensionality, but the convergence is only available in a probabilistic sense (and in particular the error depends on the specific instance of the stochastic process underlying the integration procedure) and the tightening of the convergence bounds is only achieved by increasing the number of sampling locations, at a rate of $O(1/\sqrt{n})$: as halving the error requires quadrupling the sample count, the feasibility of the brute-force approach fades away quickly, although admittedly nowhere near as quickly as it is the case for Newton-Cotes.

In fact, the approach of increasing the number of samples n to reduce integration error is well-known from practical experience in graphics to be difficult or near impossible to sustain, and the vast majority of the research in Monte Carlo integration is around increasing the quality of the estimator θ_n, or in other words, keeping n about constant (mostly as a consequence of a relatively fixed compute budget) and making sure that the standard deviation σ of the estimator is as small as possible.

Moving onto quasi-Monte Carlo, we will see how the lack of applicability of the central limit theorem to quasi-random processes makes empirical estimation of convergence rate difficult to do with tight, informative bounds. As the integration error $\varepsilon = O((\log n)^d/n)$, we see that if the number of dimensions d is large we have $(\log n)^d > n$ until $n > 2^d$, this can be a large number if d is large. In rendering d can be in the three digits, so for the hard bounds of quasi-Monte Carlo to come into play, the sample counts n need to be enormous, far beyond what is achievable with practical means. Lastly, as the Koksma-Hlawka inequality is bounded by the total variation of f, if f or its derivates jump around a lot, the bound becomes very loose making the bound on error of limited practical use. All this being said, the empirical observation is that in practice, for functions f that are differentiable (maybe except for a set of very small measure) integrals seem to converge much faster using quasi-Monte Carlo.

Background material

The references used in this section are often directing the reader to very early discussions of the various subjects. It is well understood that this has mostly historical value, and the reader is referred to classic texts in mathematical analysis such as [Apostol, 1991, Rudin, 1953, 1987, Spivak, 2006] for a more pedagogically oriented, contemporary treatment. Further classic and more modern references are provided throughout the text as appropriate.

2.2 Problem statement

The problem we are discussing is how to compute the integral θ of a real-valued function $f : U \to \mathbb{R}$ over some domain $\Omega \subseteq U$:

$$\theta = \int_\Omega f(x)\,\mathrm{d}x \qquad f\colon U \to \mathbb{R} \tag{47}$$

Focusing for a moment on the case in which the domain of f is the set of the real numbers $U = \mathbb{R}$, and the integration domain Ω is a compact subset of it, such as an interval $[a, b]$, this turns into the more familiar symbolic form

$$\theta = \int_a^b f(x)\,\mathrm{d}x \qquad f\colon \mathbb{R} \to \mathbb{R} \tag{48}$$

which corresponds to the familiar problem statement of *computing the area under the curve*: the area enclosed by the curve at the top, two vertical segments on the sides, and the x axis at the bottom.

In order to avoid getting distracted by the specific shape of the integration domain, in what follows we will focus our attention on domains $\Omega = [0, 1]^d$ which are unit cubes of dimension d. In one dimension this would be the expression

$$\theta = \int_0^1 f(x)\,\mathrm{d}x \qquad f\colon [0,1] \to \mathbb{R} \tag{49}$$

There is of course no need that the domain U of our function f be a unidimensional space, in fact it's quite clear from the introductory examples and the other sections of these notes that our main focus is indeed on functions that have a domain of dimension higher than one (often much higher).

To be more precise, in what follows we will assume that for all the functions $f\colon U \to \mathbb{R}$ under consideration, there exist a positive integer d and a differentiable isomorphism $\varphi\colon [0,1]^d \to U$ that maps the function domain U into a hypercube of some appropriate dimension. Then our integral over a given domain $\Omega \subseteq U$ can be expressed as

$$\int_\Omega f(x)\,\mathrm{d}x = \int_{\varphi^{-1}(\Omega)} f(\varphi(u)) \left|\det \left.\frac{\partial\varphi}{\partial x}\right|_u\right| \mathrm{d}u \tag{50}$$

where we have used $\left.\frac{\partial\varphi}{\partial x}\right|_u$ for the Jacobian of our map φ evaluated at u:

$$\left.\frac{\partial\varphi}{\partial x}\right|_u = \begin{bmatrix} \left.\frac{\partial\varphi_1}{\partial x_1}\right|_u & \cdots & \left.\frac{\partial\varphi_1}{\partial x_d}\right|_u \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial\varphi_d}{\partial x_1}\right|_u & \cdots & \left.\frac{\partial\varphi_d}{\partial x_d}\right|_u \end{bmatrix} \tag{51}$$

and the correction factor for the integral, $\left|\det \left.\frac{\partial\varphi}{\partial x}\right|_u\right|$ is the absolute value of the determinant of the Jacobian of φ at u. This quantity represents the local change of density of coordinates at u imposed by our map φ with respect to the coordinates at the corresponding location $\varphi^{-1}(u)$.

2.2.1 The Riemann integral

Of course, integration is a fundamental construct in mathematics, and the first formal approach to the specific meaning of this symbolism that students normally encounter was proposed by Bernhard Riemann in [Riemann, 1868][5]: start by approximating the area we're looking for as the sum of areas of rectangles under the curve, then refine this approximation using a larger and larger number of rectangles. In the limit the sum of the areas of these rectangles converges to the area under the curve.

[5]Riemann's definition of the definite integral was proposed in 1854 as part of his qualification to become professor in the university of Göttingen. The paper, titled "On the representability of a function by a trigonometric series", was published fourteen years later in the proceedings of the Royal Philosophical Society at Göttingen. Riemann's definition and discussion of the integral appears in section 4, pages 101–103

More specifically: in the case we need to integrate the function f over the interval $[a, b]$, let's subdivide the interval $[a, b]$ into n subintervals by picking $n - 1$ locations[6] $x_0 = a, \dots, x_i, \dots, x_n = b$. We will call this a *partition* of the interval $[a, b]$, and given it has n subintervals $[x_i, x_{i+1}]$, we will give it a corresponding subscript: P_n.

The most important trait of a partition P_n is its *norm* $\|P_n\|$, being the size of the largest subinterval in the partition

$$\|P_n\| = \max_i \{x_{i+1} - x_i\} \tag{52}$$

Given a partition, we can pick n *sampling locations* $t_i^n \in [x_i, x_{i+1}]$, one sampling location per subinterval. This gives us a quantity $\theta_n(P, t)$ called the *Riemann sum*, defined as:

$$\theta_n(P, t) = \sum_{i=0}^{n-1} f(t_i^n)(x_{i+1} - x_i) \tag{53}$$

This is an approximation of the integral of f over $[a, b]$ obtained by first replacing f with a piecewise constant function $\tilde{f}$, and then integrating this new function over $[a, b]$. Note that the integral of $\tilde{f}$ is easily computed exactly, due to how $\tilde{f}$ was constructed and is in fact the expression in equation (53).

With regards to sequences of partitions of a given interval, we say that a sequence of partitions is *contracting* if the sequence of its norms is decreasing and converges to 0. Given two partitions P^1 and P^2 we say that P^2 is a *refinement* of P^1 if P^2 contains all the partition locations of P^1 plus potentially more, in other words if P^2 can be obtained from P^1 by splitting some subintervals. In this case we write $P^2 \geq P^1$. A sequence of partitions is *refining* if all later elements are refinement of earlier elements: $P_i \geq P_j \Leftrightarrow i > j$. Note that refining sequences of partitions do not have to be contracting (imagine for example a refining sequence that never splits its first subinterval: if all the other intervals do get split, eventually the size of the first subinterval becomes the norm of the partition and all of its refinements constructed this way, preventing convergence to 0). In the same way, contracting sequences are in general not refining.

2.2.2 Integrable functions

The eagle-eyed reader will have noticed that there might be a possibility that if a function f is sufficiently contorted, one could find strategies for picking partition sequences and sampling locations in a few different manners, say t_i^n and s_i^n, to produce two sequences $\theta_n(P, t), \theta_n(P, s)$ that are not only different element by element, but also converge to different values. This gives rise to the concept of *Riemann-integrable function*: these are all functions for which the limit of the $\theta_n(P, t)$ sequence exists, it is finite and takes the same value independent of the choice used for the partitions P_n and sampling locations t_i^n, with the important hypothesis that the partitions make a contracting sequence. Note there is no requirement the sequences for the associated Riemann sums to have the same values, it is just the limits that need to agree.

The actual definition of Riemann-integrable function is slightly different, though. It states that $\theta = \int_a^b f(x)\,\mathrm{d}x$ if and only if for every real value $\varepsilon > 0$, however small, there exists a value $\delta > 0$ so that for all partitions P_n for which $\|P_n\| < \delta$ and all choices of sampling locations t_i^n in them one has $|\theta - \theta_n(P_n, t^n)| < \varepsilon$. Note that there is no requirement that the partition sequence be refining, it only needs to be contracting. However, if the limit of the Riemann sums exists and is the same for all contracting refining sequences, then it exists and is the same for all contracting (not refining) sequences as well.

Incidentally, it is easy to show that if f is integrable over $[a, c]$, the integral can be computed proceeding in pieces:

$$\int_a^c f(x)\,\mathrm{d}x = \int_a^b f(x)\,\mathrm{d}x + \int_b^c f(x)\,\mathrm{d}x \qquad \forall a < b < c \tag{54}$$

The left hand side of the equation tells us that when computing the Riemann sums for f over $[a, c]$, their limit values are the same independent of what specific partition sequence is used (this is the definition that f be integrable). Now looking at the right hand side: merging the partitions over $[a, b]$ and $[b, c]$, one obtains partitions of $[a, c]$

[6] Without loss of generality, these locations will be assumed to be in increasing order: $i < j \Rightarrow x_i \leq x_j$

that just so happen to have a partition location at b, so these merged partitions are just a subset of all possible partitions of $[a, c]$ and our statement is proved.

One example of function that is not Riemann-integrable is constructed as follows: we start with the *characteristic function* of a set A, being the function $\chi_A(x)$ that has value 1 on A and 0 otherwise:

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

the characteristic function can be applied to any set. Let's look at a few examples: the characteristic function of an interval $[b, c]$ can easily be integrated over any interval $[a, d]$. Let's look at the case $a < b < c < d$:

$$\int_a^d \chi_{[b,c]}(x)\,\mathrm{d}x = c - b \tag{56}$$

the result is easily proven because the function is piecewise constant. We can restate our original integral as a sum of three terms

$$\int_a^d \chi_{[b,c]}(x)\,\mathrm{d}x = \int_a^b \chi_{[b,c]}(x)\,\mathrm{d}x + \int_b^c \chi_{[b,c]}(x)\,\mathrm{d}x + \int_c^d \chi_{[b,c]}(x)\,\mathrm{d}x \tag{57}$$

where for the first and third term on the right hand side the integral is 0 and for the second term the integral is $c - b$, in all three cases independent of how the partition locations or sampling locations are chosen.

A second simple example is starting from a set of one value $X = \{\bar{x}\}$ and computing the integral of χ_X, which is a function that is everywhere 0 except at one location $\bar{x}$, where it is 1:

$$\int_a^b \chi_X(x)\,\mathrm{d}x = 0 \tag{58}$$

assuming $\bar{x} \in [a, b]$ (otherwise the integral is 0 because χ_X is everywhere 0) there are two possibilities for partitions of $[a, b]$: either $\bar{x}$ is a partition location, or it's not. If it is not, there is one subinterval in our partition that contains $\bar{x}$: if the sampling location on that subinterval is at $\bar{x}$, we have θ_n be the size of this subinterval, otherwise θ_n is 0. However, in the case in which all partitions in a sequence do have a sampling location at $\bar{x}$, the convergence of the norm of the partitions in the sequence to 0 brings the limit of the associated Riemann sums θ_n to 0. The last case to analyse is the case in which $\bar{x}$ is a partition location: in that case there are two subintervals instead of one that have a chance of making θ_n non-zero in case the corresponding sampling locations happen to be at $\bar{x}$. Once more, as the norms of the partitions converge to 0, so do the associated θ_n, proving our statement. Of course the same results applies for all finite sets $X = \{\bar{x}_0, \dots, \bar{x}_n\}$, which is easily proven by first subdividing the integration domain in regions around the various $\bar{x}_i$ and then applying the previous result.

Now the question is what happens if we have infinitely many points in X, for example what if $X = \mathbb{Q}$, the set of rational numbers. Turns out that $\chi_{\mathbb{Q}}(x)$ is not Riemann-integrable over the interval $[0, 1]$: in fact we can choose a sequence of partitions

$$P_n = \left\{ \left[\frac{i}{n}, \frac{i+1}{n} \right] \right\}_{i \in \{0,\dots,n-1\}} \tag{59}$$

and two sequences of sampling locations t_i^n, s_i^n:

$$t_i^n = \frac{i + \frac{1}{2}}{n} \qquad s_i^n = \frac{i + (\sqrt{2} - 1)}{n} \tag{60}$$

where it can easily be seen that

$$\chi_{\mathbb{Q}}(t_i^n) = 1 \qquad \chi_{\mathbb{Q}}(s_i^n) = 0 \qquad \forall n, i \in \mathbb{N} \tag{61}$$

and these make two Riemann sums $\theta_n(P, t) = 1$ and $\theta_n(P, s) = 0$ for all values of n.

In fact this example of bounded function that is not Riemann integrable captures the essence of the problem: the trouble for integrability in the Riemann sense comes from jump discontinuities, namely when there are too many and too finely spaced. If our function f has a finite set of jumps and is elsewhere continuous, the refinement of the partitions will eventually isolate the discontinuities, eventually minimizing the influence from the jumps away to the point where they don't make any contribution to the final value of the integral. The alternative view is that the way to deal with jump discontinuities is simply splitting the function in branches just at the discontinuity locations, so that the integral over the whole domain is simply a sum of separate integrals of continuous functions over a few smaller, contiguous subdomains. If instead the set of jumps has infinite cardinality (countable or more) it becomes impossible to cut the function into branches as before, because each integration domain is smaller than the current means can appropriately capture (in fact it's 0 sized).

2.2.3 Interlude: the Fundamental Theorem of Calculus

Analysis takes back with one hand what it gives with the other. I recoil in fear and loathing from that deplorable evil: continuous functions with no derivatives

Charles Hermite

The *Fundamental theorem of calculus* can be loosely stated as saying that the operation of derivative and indefinite integration are inverses of one another. In fact it is the case for all f that are Riemann integrable that the indefinite integral is a continuous differentiable function, and that the derivative of this function equals f:

$$\frac{d}{\mathrm{d}x}\int f\,\mathrm{d}x = f \tag{62}$$

and it can be shown that the opposite relation is also true, up to a constant: the indefinite integral of the derivative of f is equal to $f + C$ for some constant $C \in \mathbb{R}$. Indeed this result is so powerful that symbolic integration of a given function f is in practice performed by finding the *antiderivative* of f (being a function g so that its derivative equals f) and then evaluating it at the boundary of the domain of definition, while direct proofs of integrability in the Riemann sense are generally framed as academic curiosities. We invite the reader to carefully reconsider such a position in order to maximize the value of what follows in these pages.

2.2.4 The Lebesgue integral

There is also another way of looking at the integration process, proposed by Henri Lebesgue in [Lebesgue, 1904]: this is a far more powerful approach and was used by Andrey Kolmogorov in the 1930's as the foundation of his probability theory. In Lebesgue's approach, instead of starting by slicing the domain of the function into vertical regions, and then summing the area of rectangles spanning these regions, one starts with slices of the set of function values, and then proceeds to ask the question of what is the set $\omega_i \subseteq \Omega$ over which the function f actually does take values in the selected ranges. Formally these are the sets defined as $\omega_i = \{x \in U \,:\, f(x) \in [y_i, y_{i+1}]\}$, or with some rather abusive overload of notation $\omega_i = f^{-1}([y_i, y_{i+1}])$. Do note that while the value intervals are closed and compact, the topology of the sampling intervals ω_i can be more complex, especially in cases in which f is not continuous.

The key new ingredient in this approach to the integral is a function μ called *the measure*, which given a set $\omega \subseteq U$ returns its measurement $\mu(\omega) \in \mathbb{R}$, a non-negative real number. Once a measure over sets in U is available, the process of integration can be described in a manner similar to the Riemann integral: first the range of $f\colon U \to \mathbb{R}$ is divided into n segments picking value locations $y_0, \ldots, y_n$ as well as one sample value per segment $s_i \in [y_i, y_{i+1}]$, then the corresponding sets $\omega_i = f^{-1}([y_i, y_{i+1}])$ are determined, and the sequence of θ_n is defined as

$$\theta_n = \sum_{i=0}^{n-1} s_i \mu(\omega_i) \tag{63}$$

lastly the integral is defined, much like before, as:

$$\theta = \lim_{n\to\infty} \theta_n \tag{64}$$

There can be many functions defined from subsets of U and mapping to non-negative real numbers, but in most cases just having a function that given any subset of U returns a positive real value is not good enough. For such a function to be a measure on U it has to behave like our intuition tells us measuring things in the real world would behave (for example, larger sets should have a measure higher than smaller sets, or the measure of unions of sets should be the sums of their individual measures if the are no overlaps). It can be proven that it's not possible to exhibit a function μ that is both a good measure and capable to deal with any subset of U, and so it arises the notion of *measurable subset*, being is a subset of U that μ is actually able to operate on.

Once we have μ and its associate family of measurable subsets, the last requirement is for μ to be *σ-additive* that is, it must have the property that given a countable or finite collection $\{\Omega_i\}$ of disjoint measurable subsets of U the measure of their union equals the sum of their individual measures:

$$S \text{ countable} \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \in S \implies \mu\left(\bigcup_{i\in S} \Omega_i\right) = \sum_{i\in S} \mu(\Omega_i) \tag{65}$$

One interesting and very valuable property of σ-additive measure is that they turn out to be *monotonic*: if a set ω_1 is a subset of a set ω_2, the measure of the first needs to be smaller than the measure of the second:

$$\omega_1 \subseteq \omega_2 \qquad \implies \qquad \mu(\omega_1) \leq \mu(\omega_2) \tag{66}$$

this also means that the measure of a set $\omega \subseteq U$ must be the measure of the smallest open set that contains it, or more precisely:

$$\mu(\omega) = \inf_{\substack{\omega\subseteq A \\ A \text{ open}}} \mu(A) \tag{67}$$

Reasoning about the properties of measures it becomes clear that given a few measurable sets it's possible to construct other ones that must be measurable, such as their union as well as their difference. In fact a little bit more is possible: the space of measurable set for a σ-additive measure is called a *σ-algebra*: this is a collection of subsets of U that includes U itself, is closed under complement, and is closed under countable unions. The definition immediately implies that any σ-algebra also includes the empty subset and that it is closed under countable intersections, as complements of U and countable unions respectively.

In mathematics one meaning of the word *algebra* is to indicate a structure over a set that provides it with two operations, which are often thought of as addition ($+$) and multiplication ($\cdot$). These operations each have a neutral element, indicated respectively as 0 and 1, so that $a + 0 = a$ and $a \cdot 1 = a$, and have a distribution rule with the same structure encountered in elementary mathematics: $a \cdot (b + c) = a \cdot b + a \cdot c$. More often than not when two operations are available on a set, the first (addition) is commutative ($a + b = b + a$) and invertible ($\forall a \ \exists b \ : \ a + b = 0$), whereas the second may or may not be (non commutative is the space of matrices over a field, non invertible is the set of integers $\mathbb{Z}$).

In set theory one can almost have algebras of sets using union as the first operation and intersection as the second, with the empty set being the neutral element for the first operation and the full set being the neutral for the second. Distribution rules work out as expected. Unfortunately it so happens that with this definition the first operation is commutative but not invertible. This issue can be solved introducing the symmetric difference operation $A \triangle B = (A \cup B) \backslash (A \cap B)$ (the full story is quite intriguing and is explored in great detail in [Kolmogorov and Fomin, 1960]). Our last note will be that the symmetric difference can be used to define a metric d on a measurable space $d(A, B) = \mu(A \triangle B)$ if the sets are considered equivalent when their symmetric difference has zero measure $A \sim B \Leftrightarrow \mu(A \triangle B) = 0$.

Note that the definition of the algebra structure involve finite quantities of operands: for example in $\mathbb{Q}$ any finite addition or rational is still a rational, but it's quite easy to find a series of rationals that converges to irrational numbers, for a classic example consider Leibniz's formula for π:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \tag{68}$$

by way of contrast a σ-algebra is instead a structure that is closed under countable applications of either operation. This can be somewhat delicate to achieve, the example in equation (68) shows how $\mathbb{Q}$ won't admit a σ-algebra structure with the ordinary definition of operations, because closure under countable sums and products would turn it into $\mathbb{R} \cup \{-\infty, +\infty\}$.

Like in the case of the Riemann integral, a function f is *Lebesgue integrable* if the sequence of the approximations θ_n from equation (63) exists, it is finite and is the same value independent of the choice used for the various value locations y_i and sample values s_i. Like before, there is a requirement that the value intervals $[y_i, y_{i+1}]$ are constructed so that they all get smaller and smaller, in particular the largest needs in the limit to shrink to 0 height. There is also one extra requirement: namely that the measure μ is in fact able to measure the preimage of any value interval: that is for any a, b in the image of f it is necessary that $\mu(f^{-1}([a, b]))$ exists and be finite.

Let's go over the sequence of events with an example: let's say we want integrate $\chi_{[a,b]}(x)$ over $[0, 1]$:

$$\theta = \int_0^1 \chi_{[a,b]}(x) \tag{69}$$

under the condition $0 < a < b < 1$. Given the function χ only takes on values 0 and 1, we pick for our values of y_i the sequence $0, \frac{1}{2}, 1$ thereby having the intervals $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$ and the domain sets $\omega_1 = [a, b]$ and $\omega_0 = [0, a] \cup [b, 1]$. Then our integral is

$$\theta = 0\mu(\omega_0) + 1\mu(\omega_1) = \mu(\omega_1) = b - a \tag{70}$$

As a second example, we can show how the Lebesgue integral has a lot less trouble integrating functions that are not integrable in the Riemann sense. The classic example is $\int_0^1 \chi_{\mathbb{Q}}(x)\,\mathrm{d}x$: the Lebesgue integral is of course $1 \cdot \mu(\chi_{\mathbb{Q}}(x) \cap [0, 1])$ which leaves us with the question of what's the measure of $\chi_{\mathbb{Q}}(x) \cap [0, 1]$. This is where σ-additivity comes in to help: first off, singleton sets $X = \{x\}$ must have measure $\mu(X) = 0$:

$$X = \bigcap_{n \in \mathbb{N}} \left(x - \frac{1}{n}, x + \frac{1}{n}\right) \implies \mu(X) = \mu\left(\bigcap_{n \in \mathbb{N}} \left(x - \frac{1}{n}, x + \frac{1}{n}\right)\right) = \lim_{n \to \infty} \frac{2}{n} = 0 \tag{71}$$

having this piece of information, we know from set theory that $\mathbb{Q}$ is countable, which means $\mathbb{Q} \cap [0, 1]$ is also countable. The we can use equation (65) with $S = \mathbb{Q} \cap [0, 1]$ to conclude that $\mu(\mathbb{Q} \cap [0, 1]) = 0$. Incidentally this is also the reason why we have a limitation in the definition of σ-algebra that unions and intersections should be finite or countable: if uncountable unions were admitted, we could for example express the interval $[0, 1] \subseteq \mathbb{R}$ as a union of singletons X as above, obtaining the contradiction $1 = \mu([0, 1]) \neq \mu(\cup X_i) = \sum \mu(X_i) = 0$ which violates our hypotheses on the σ-additivity of μ.

Now that we know that Lebesgue integration can deal with functions that Riemann integration cannot, the question is whether examples of the reverse exist: functions that are Riemann integrable but not Lebesgue integrable. It turns out that this never happens if the integration domain has finite measure, but can happen for infinite integration domains (these are called *improper Riemann integrals*) due to how the integration over infinite domains is defined differently in the two cases. However, the focus of these notes is on integration of functions over bounded domains only, so we leave if to the interested reader to follow up on this thread on a book on real analysis such as [Kolmogorov and Fomin, 1960]. We conclude confirming that when a function f is integrable both in the Riemann as well as in the Lebesgue sense, the two integrals have the same value.

2.2.5 A topological interlude

If geometry is dressed in a suit coat, topology dons jeans and a T-shirt

David S. Richeson

One question naturally comes up: more often than not, even if the domain U of our function f is not a simple subset of $\mathbb{R}^d$ (say a ball or a hypercube), the space U has subsets that are *open* (and of course also has subsets that are *closed*). The reader will recall a *topological space* is a space U and a family $\mathcal{T}$ of it subsets so that

$$\begin{array}{ll} U, \emptyset \in \mathcal{T} & \text{the full space and the empty set are in the topology} \\ \{\omega_i\}_{i \in I} \subseteq \mathcal{T} \implies \bigcup_{i \in I} \omega_i \in \mathcal{T} & \text{closed under } \textit{arbitrary} \text{ union} \\ \{\omega_i\}_{i=0}^n \subseteq \mathcal{T} \implies \bigcap_{i=0}^n \omega_i \in \mathcal{T} & \text{closed under } \textit{finite} \text{ intersection} \end{array} \tag{72}$$

under this condition $\mathcal{T}$ is the family of the *open sets* contained in U. The set U will also contain *closed sets* defined as complements of open sets. Note that the two notions of open and closed are not mutually exclusive: there are sets that are both open and closed at the same time, for example U and $\emptyset$. Such sets are sometimes called *clopen*. To give an example of some interest, consider the space $\mathbb{Q}$ with the ordinary topology induced by the euclidean metric (in other words, start from ordinary open intervals on $\mathbb{Q}$ to build a topological space, and then complete it by adding all the sets resulting from finite intersections and arbitrary unions of those). Now take the set $A = \{q \in \mathbb{Q} \;:\; q > 0, q^2 > 2\}$ of all positive rational numbers whose square is bigger than 2. Given that $\sqrt{2}$ is not in $\mathbb{Q}$, one can show quite easily that A is a clopen subset of $\mathbb{Q}$ (on the one side it's open, because it's the union of all open intervals with the low end larger than $\sqrt{2}$; however it's also closed because it's the complement of the union of all open intervals with the high end smaller than $\sqrt{2}$). Interestingly A is not a clopen subset in the usual topology over $\mathbb{R}$: it is neither open nor closed in $\mathbb{R}$, because infinite sets in $\mathbb{R}$ that only contain elements of $\mathbb{Q}$ (A is made of rationals) are not open nor closed (sets containing a finite number of elements of $\mathbb{Q}$ are closed because they are finite union of closed sets).

Topology is the method to reason in an abstract way about the general notion of *proximity* (as intended in natural language) between elements in U: the origin of the idea is that two objects are in proximity of one another if they belong to the same open set. This is also a starting point for the intuition as to why it is that the open sets (as opposed to the closed ones) are the fundamental concept in topology: it can be shown that the difference between an open set A and its *closure* $\bar{A}$ (defined the intersection of all the closed sets that contain A) is its boundary ∂A. The dual notion of closure is the *interior* A°, being the union of all open sets contained in A. These definitions imply that

$$A^\circ = A \Leftrightarrow A \text{ open} \qquad \bar{B} = B \Leftrightarrow B \text{ closed} \qquad \forall A \implies \bar{A} = A^\circ \cup \partial A \tag{73}$$

and of course classic mental image for these concepts is that given a set A the interior of A is its "flesh" and the exterior is its "skin", which takes on extra significance when a geometric interpretation is added: it can be shown that the boundary of a set has dimension strictly lower that the set itself. In fact if M is a sufficiently well behaved geometric object of dimension d (by which we mean a *smooth manifold*, a generalization to any dimension of the 3-space concept of a smooth surface), its boundary is either a manifold of dimension $d - 1$ or it is the empty set. For example, a ball in dimension 3 is a manifold, its boundary is a sphere (an object of dimension 2). In turn the sphere has no boundary. Readers interested in this subject might like to follow the classic treatment in [do Carmo, 1976] or possibly the more modern approach of [Lee, 2003].

Topology helps us generalize everyday geometric notions to structures of increasing generality. Think for example of the concept of distance between two elements of a set: in a somewhat roundabout way, one could define the distance between two elements x and y as the result of the process of taking the infimum diameter over the family of all the open balls that contain both x and y. This is based on the definition of the *open ball* at x of radius r as

$$B(r, x) = \{y \in U \;:\; \text{dist}(x, y) < r\} \tag{74}$$

for some distance function $\text{dist}(\cdot, \cdot)\colon U \times U \rightarrow \mathbb{R}_{\geq 0}$. Of course if our space has such a distance function, we can easily prove that the distance between x and y is indeed as defined. However our new definition being based only on a family of sets and the notion of their diameter (which is a construct similar to the set measure), it does not require a distance being defined on the space at all.

The cardinality of the family of balls above is obviously very high, and its structure impedes imposing a total order on it, although a natural partial order can be induced starting from inclusion: $B_1 \leq B_2 \Leftrightarrow B_1 \subseteq B_2$. Even with a natural partial order it's still beneficial to work with families of balls with lower cardinality, because although imposing a total order on the sets in the family is too restrictive a limitation, an order imposed "from the outside" is still a useful machinery to have. In fact, building ordered countable collections of open sets in such a manner that later elements are included in earlier ones is all that is needed to support the abstract notion of limit of a sequence[7] of which our generalized definition of distance is one specific application. The treatment of the subject is certainly fascinating, but it gets quite technical very quickly, due to the highly abstract nature of the material, classic references on the subject are the still excellent [Bourbaki, 1966] as well as the more recent [Munkres, 2000].

[7] In the field of topology, constructs of this kind are called *filters* or *nets* depending on the specific requirements for inclusion

Given their nature of indicators for proximity, it is quite natural to wonder if a measure can be constructed so that it would be able to deal with all open sets of a given topology on our function domain U. This is certainly quite possible: Émile Borel introduced the notion of *Borel sets*, being the closure under countable union and countable intersection of the open sets in a topological space. The Borel sets form a σ-algebra which is the smallest σ-algebra containing all the open sets of a given topological space. A measure able to deal will all the opens is necessarily able to deal with all Borel sets in a given space, and it's called the *Borel measure* on the space. The eagle-eyed reader will have noticed how the Borel algebra of sets has two implications: one is that the corresponding measure is able to deal with arbitrary unions of open sets (because all the opens are measurable), which includes finite unions, as well as countable and uncountable (albeit uncountable unions of open sets can be difficult to wrap one's head around). The other is that although they are arguably not open nor necessarily closed, the measure can deal with countable (as opposed to finite) intersections of open sets.

The requirement of a measure to be monotonic creates a link between measures and topology, especially when a measure is considered under the perspective coming from equation (67), which seems to indicate that for measures open sets have a special role. In fact we can now see why it is desirable for topology and measure to interact in constructive ways. Further it might seem that proximity of one set to another, or elements from one set being densely distributed into another might carry implications into the relative measures of the two.

However, it turns out this is not necessarily the case. To understand why let's consider a few examples so we can observe how the specifics work out. There are sets made of points that are arbitrarily near to other points, yet have 0 measure: we have discussed before the case of $\mathbb{Q}$, for example. From a topological perspective $\mathbb{Q}$ is everywhere dense in $\mathbb{R}$: for any value $r \in \mathbb{R}$ and for any $\varepsilon > 0$ there exists a $q \in \mathbb{Q}$ so that $|r - q| < \varepsilon$, or in other words, the open sets of $\mathbb{R}$ are unable to separate $\mathbb{Q}$ from $\mathbb{R}$: all points in $\mathbb{Q}$ are in proximity of points of $\mathbb{R}$, and viceversa. However, given an interval $[a, b] \in \mathbb{R}$, we've seen before how the Lebesgue measure on $\mathbb{R}$ will give us $\mu([a, b]) = (b - a)$, as well as $\mu([a, b] \cap \mathbb{Q}) = 0$. This may at first appear as some kind of contradiction, but when approached from a probabilistic perspective it makes sense: having 0 measure does not mean that a set is empty, it means that the probability of picking a given point by chance from it is 0. Indeed, as much as $\mathbb{Q}$ is dense in $\mathbb{R}$, it still is very "skinny", and in fact we know contains far fewer elements than $\mathbb{R}$: $[a, b] \cap \mathbb{Q}$ has $\aleph_0$ points whereas $[a, b]$ has $\aleph_1 = 2^{\aleph_0}$ elements in it, which is a higher order infinity. Readers interested in general set theory might find useful material in the classic [Kuratowski and Mostowski, 1967].

2.2.6 Integrals in higher dimensions

If the domain of function f has dimension $d > 1$ very few changes are done formally to the descriptions above. The only potentially delicate discussion is around how the integration domain Ω is partitioned into subdomains when considering Riemann integration: the crucial property here is that points in Ω are either in the interior of exactly one subdomain or on the border of at least one subdomain. The first part of the hypothesis is easy to understand: for $x \in \Omega$ to be in the interior of subdomain Ω_i means that there exists a positive value $\varepsilon > 0$, likely small, for which the d-dimensional open ball $B^d(\varepsilon, x)$ of radius ε centered at x is entirely contained in Ω_i. Observe how such a ball has $\mu(B^d(\varepsilon, x)) > 0$.

On the other hand, for a different point $y \in \Omega$ to be on the boundary $\partial\Omega_i$ of the subdomain Ω_i, it means that there exist a dimension $c < d$ and a positive value $\varepsilon > 0$, likely small, for which the c-dimensional ball $B^c(\varepsilon, y)$ of radius ε centered at y is entirely contained in $\partial\Omega_i$. Observe how in this case for all $c < d$, and for any value of ε, we have $\mu(B^c(\varepsilon, y)) = 0$. The euclidean intuition of this is a consequence of $B^c(\varepsilon, y)$ having zero extent in the $c - d$ dimensions orthogonal to $\partial\Omega_i$ at y: when the measure is computed the volume of the ball will indeed be non-zero as a c-dimensional object, but it will then be multiplied by 0 for the cospace covering the remaining dimensions[8].

So given that the measure of all the subdomain boundaries is $\mu(\partial\Omega_i) = 0$, and that the subdomains are a finite count and countably many in the limit, their collective measure $\mu(\bigcup_i \partial\Omega_i)$ is also 0, and the behaviour of f at these locations is of no consequence. The observing reader will see how making a purely Riemann argument along these lines would be somewhat more delicate, while reaching the same conclusion.

[8]To visualize why the value of c is not always $d - 1$ as one might expect, consider for example $d = 3$ and Ω_i being a cube. If $y \in \partial\Omega_i$ a few cases are possible: either y is resting on a face of the cube, in which case we have indeed $c = 2$. However y could also lie on an edge ($c = 1$) or on a corner ($c = 0$). This has to do with the fact that our chosen test uses open balls, not with the dimension of the boundary manifold $\partial\Omega_i$ which is indeed $d - 1$.

2.3 Numerical integration

Now that we have an understanding of the continuous setting for integration we move onto the discrete case, which is necessary for numerical treatment of the integration problem. The first step is to understand what it means to have a function in the discrete case, and one possible way of looking at this question is to think of it as the situation in which values for our function are known for a set of given locations only, and an expression is wanted to provide values at other locations.

For example, say that a few functions $e_0(x), \dots, e_n(x) : \Omega \to \mathbb{R}$ were available to us, built so that we could calculate

$$\theta_k = \int_\Omega e_k(x)\,\mathrm{d}x \qquad \forall k \tag{75}$$

exactly making use of analytical expressions. If we also had a method so that for all the functions $f : \Omega \to \mathbb{R}$ of interest to us we could determine a bunch of real numbers $w_0, \dots, w_n \in \mathbb{R}$ so that we'd have

$$f(x) = \sum_{k=0}^{n} w_k e_k(x) \qquad \forall x \in \Omega \tag{76}$$

then we could immediately conclude that our integral θ is simply

$$\theta = \int_\Omega f(x)\,\mathrm{d}x = \int_\Omega \sum_{k=0}^{n} w_k e_k(x) = \sum_{k=0}^{n} w_k \theta_k \tag{77}$$

In this case we would say that the function set $\{e_k(x)\}_{k\in\{0,\dots,n\}}$ is a *basis* for the space $\mathcal{F}$ of functions we're interested in working with. The usual concepts from linear algebra and vector spaces apply to the function space we want to work with as well: we'd want that if two functions $f, g \in \mathcal{F}$ are given from our space, their sum is also in the space $f + g \in \mathcal{F}$ as well as their scaled version by any real $\alpha \in \mathbb{R} \Rightarrow \alpha f \in \mathcal{F}$.

As much as sometimes it is indeed possible to work with bases $\{e_k(x)\}_{k\in\{0,\dots,n\}}$ that actually do span the functions spaces $\mathcal{F}$ we're interested in, a far more common occurrence is that the span of our basis is actually only able to produce functions $\tilde{f}$ that are close to (that is, they *approximate*) our desired function f. There are many ways for two functions to approximate one another, so let's establish that our notion of distance between two functions is *uniform*. More precisely we'll say that the distance between two functions is defined as follows:

$$\mathrm{dist}_\infty(f, g) = \operatorname*{ess\,sup}_{x\in\Omega} |f(x) - g(x)| \tag{78}$$

where the *essential supremum* operator (ess sup) indicates the supremum may be exceeded over a set of measure 0. This definition highlights the fact that if two functions behave differently only on a 0-measure set, their integrals will have the same value.

Often times this is a reasonable trade-off: the functions $f \in \mathcal{F}$ we need to integrate are approximated by a different set of functions $\tilde{f} \in \mathcal{G}$ where $\mathcal{G}$ is spanned by a known basis $\{e_k(x)\}_{k\in\{0,\dots,n\}}$. Examples of function spaces $\mathcal{G}$ that are in common use are polynomials of a given degree, sums of scaled and translated gaussians, sums of sines and/or cosines of integer frequencies, and so on. Notwithstanding the fact that the functions in $\mathcal{F}$ are the ones we need to integrate, it is really the functions in $\mathcal{G}$ that are the ones we *can* actually integrate. So what we need is a method that for each f gives us an approximant $\tilde{f} = \sum w_k e_k \in \mathcal{G}$ so that

$$\mathrm{dist}_\infty(f, \tilde{f}) < \varepsilon \tag{79}$$

for some small value[9] of ε. This would then immediately tell us that

$$|\theta - \sum w_k \theta_k| < \varepsilon \mu(\Omega) \tag{80}$$

Now the question becomes how to determine the w_k (often called the *weights*). The first idea that comes to mind to most people is *projection*, which has its simplest form when the basis for $\mathcal{G}$ is *orthogonal*. In that case, key

[9]There is a case in which the distance could actually be 0, when f differs from some element of $\mathcal{G}$ only over a 0-measure set

ingredient for projection is the *scalar product*, which in this context is represented with angle brackets $\langle\cdot,\cdot\rangle$ and defined as follows:

$$\langle f,g\rangle = \int_\Omega f(x)g(x)\,\mathrm{d}x \tag{81}$$

The scalar product for functions is the continuous version of the scalar product (or *dot product*) for vectors.

The rest follows through the same constructs used in linear algebra: two functions $f,g \in \mathcal{G}$ are orthogonal if their scalar product is 0, a basis for $\mathcal{G}$ is orthogonal if its elements are pairwise orthogonal and is *orthonormal* if it is orthogonal as well as $\langle e_k, e_k\rangle = 1$ for all k (that is, the basis elements are normalized). With these ingredients projecting f onto $\mathcal{G}$ is simply computing weights as

$$w_k = \langle f, e_k\rangle \qquad \forall k \in \{0,\ldots,n\} \tag{82}$$

if the basis elements are not normalized, the expression for the weights needs to compensate for the length of the basis elements.

There exists however an alternative approach: one could choose a set of sampling locations $t_j \in \Omega$, with $j \in \{0,\ldots,n\}$ and set up a system of $n+1$ equations:

$$\begin{cases} f(t_0) &= \sum_{k=0}^n w_k e_k(t_0) \\ f(t_1) &= \sum_{k=0}^n w_k e_k(t_1) \\ \quad\vdots & \\ f(t_n) &= \sum_{k=0}^n w_k e_k(t_n) \end{cases} \tag{83}$$

If we were to say that $\bar{t}$ is the vector of dimension $n+1$ that has the various t_j as its coordinates $\bar{t} = (t_0,\ldots,t_n)$, as well as abuse notation a bit and say that applying a function to a vector returns a vector $f(\bar{t}) = (f(t_0),\ldots,f(t_n))$ adding to the mix $\bar{w} = (w_0,\ldots,w_n)$ and a matrix E containing the evaluations of the basis functions e_k at the sample points t_j:

$$E = \begin{bmatrix} e_0(t_0) & e_1(t_0) & \cdots & e_n(t_0) \\ e_0(t_1) & e_1(t_1) & \cdots & e_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ e_0(t_n) & e_1(t_n) & \cdots & e_n(t_n) \end{bmatrix} \tag{84}$$

we could follow standard practice for linear systems theory and restate the system in equation (83) as

$$f(\bar{t}) = E\bar{w} \tag{85}$$

which would give us a different method of finding our weights w_k by inversion of E:

$$\bar{w} = E^{-1}f(\bar{t}) \tag{86}$$

The large advantage of the method in equation (86) over equation (82) is that it requires only $n+1$ evaluations of f at given single locations, advantage compounded by the fact that f doesn't have to have a known analytical integral form, nor actually be known analytically at all. Further, we can now deal not only with bases $\{e_k\}$ that are not normalized, but also bases that are not orthogonal. There are however two disadvantages, one is that the matrix E must be invertible: this is usually a modest concern, because in typical cases the whole space $\mathcal{G}$ of approximant functions is constructed deliberately with the needed properties. The other, larger disadvantage is that this method won't be able to take into account the behavior of f away from the sampling locations t_j, which in turn makes giving bounds on the distance ε between f and its approximant more complicated and often far more conservative.

2.3.1 Lagrange interpolation

As a classic example of practical application of the ideas expressed above, we're going to look into using *Lagrange interpolation* for numerical integration. We will start with the function space $\mathcal{P}_n$ being the space of polynomials of degree n, which has a fairly obvious basis $e_k(x) = x^k$. This would be a workable basis per se, however if we were to go and construct matrix E from equation (85) we would get a dense matrix in which the rows are geometric series of the sampling locations t_j, a specific structure called a *Vandermonde matrix*. In many cases these matrices can end up having rather large condition numbers, making the numerical inversion behave poorly, a phenomenon described for example in [Gautschi, 1975].

We can do better than this: it is more convenient to use as basis for $\mathcal{P}_n$ the Lagrange polynomials, which are polynomials built as follows: first the set of $n + 1$ sampling locations $t_j \in \Omega, j \in \{0, \dots, n\}$ is chosen, and then our basis $\{l_k(x)\}$ is built as

$$l_k(x) = \prod_{\substack{0 \le j \le n \\ j \ne k}} \frac{x - t_j}{t_k - t_j} \tag{87}$$

It's easy to see how the Lagrange polynomials have the following property

$$l_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \ne j \end{cases} \tag{88}$$

because when $i = j$ all the terms in the product are 1 and otherwise exactly one of them is 0. Given this property, we can immediately conclude that matrix E from equation (85) is actually the identity matrix I, so that our weights $w_k = f(t_k)$.

At this point of course it's natural to ask what's the integration error when using this procedure. As it was said before, a first conservative bound can be estimated from the interpolation error, per equation (80). In fact it can be proven that if the function f has continuous derivatives all the way up to and including order $n + 1$ over domain U, then the behaviour of such derivative $f^{(n+1)}(x)$ characterizes the interpolation error as follows:

$$\forall x \in U \quad \exists \xi \in U \implies f(x) - \tilde{f}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n} (x - t_j) \tag{89}$$

with the equal sign indicating that this is not an upper bound, but a value that is actually attained. We observe this formula makes some intuitive sense if one thinks how the approximant $\tilde{f}$ must be fairly close to the Taylor expansion of f, truncated at degree n, effectively the derivative of order $n + 1$ is a way of observing the behaviour of whatever is left of f once the approximant has been taken out. This gives us a first, *very conservative* bound on integration error as

$$\left| \theta - \int_\Omega \tilde{f}(x)\,\mathrm{d}x \right| \le \mu(\Omega) \max_{x,\xi \in \Omega} \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n} (x - t_j) \tag{90}$$

If we make the specific choice $t_j = \frac{j}{n}$ using this basis, we obtain the well-known *Newton-Cotes quadrature of order n*. Given the interpolation error as above, it can be shown that if f is well-behaved (for example, if f is *analytic*, or if the series of Taylor coefficients is absolutely convergent) the sequence $\tilde{f}^{\,n}(x)$ of projections of f onto the function spaces $\mathcal{P}_n$ as described above converges uniformly to f as n grows:

$$\lim_{n \to \infty} \tilde{f}^{\,n}(x) = f(x) \qquad \text{(for well-behaved } f) \tag{91}$$

Trying to reduce integration error by raising the quadrature order n is thereby problematic when f has shapes that are difficult to capture with polynomials (such as discontinuities, growing Taylor coefficients or other similar traits) because the error introduced by the interpolant doesn't actually decay to 0, making the integration scheme much less useful than it first appears. Carl Runge exhibited in [Runge, 1901] a function for which the approximation error grows without bound as the interpolation order increases, if the sampling locations are chosen in an equispaced manner. Runge's fiendishly simple example is called the *Runge function*:

$$f(x) = \frac{1}{1 + 25x^2} \tag{92}$$

the first few derivatives of this function are already rather problematic:

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{50x}{(1 + 25x^2)^2} \qquad \frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) = \frac{50(75x^2 - 1)}{(1 + 25x^2)^3} \qquad \frac{\mathrm{d}^3}{\mathrm{d}x^3} f(x) = \frac{15000x(25x^2 - 1)}{(1 + 25x^2)^4} \tag{93}$$

it is useful to point out that Runge's phenomenon has many theoretical and practical similarities to Gibb's phenomenon [Gibbs, 1898, 1899, Wilbraham, 1848] of overshooting near jump discontinuities in truncated Fourier series.

Let's look into integration error for Newton-Cotes formulas: if the samples are equally spaces at a distance h, defining $s = \frac{x-x_0}{h}$ we can restate equation (89) as

$$\forall x \in U \quad \exists \xi \in U \implies f(x) - \tilde{f}(x) = \frac{f^{(n+1)}(\xi) h^{n+1}}{(n+1)!} \prod_{j=0}^{n} (s-j) \tag{94}$$

with $\xi \in [x_0, x_n]$ which gives us the following estimation for integration error

$$\left| \theta - \int_\Omega \tilde{f}(x)\, \mathrm{d}x \right| = \frac{f^{(n+1)}(\xi) h^{n+2}}{(n+1)!} \int \prod_{j=0}^{n} (s-j) ds \qquad \exists \xi \in U \tag{95}$$

where the integral uses the variable substitution $\mathrm{d}x = d(x_0 + sh) = hds$ and is extended over the corresponding remapping of Ω.

Given the empirical observation that lower-order derivatives tend to be the most innocently-behaved, a common approach to the practical use of Newton-Cotes quadrature is the so-called *composite rule*, in which the order of quadrature is kept fixed and low, often a value such as 0, 1 or 2, and instead new sampling locations are added by splitting the integration domain Ω into a number of subintervals Ω_j. For example for order $n = 2$ we would have

$$\theta_m^2 = \sum_{j=0}^{m-1} \int_{\Omega_j} \sum_{i=0}^{n} f(t_{j,i}) l_i(x) \tag{96}$$

where, as one expects, we have $t_{j,i} \in \Omega_j$. Indeed it is immediately apparent that the composite rule for Newton-Cotes quadrature of order 0 is effectively a form of the Riemann integral, equation (53):

$$\theta_m^0 = \sum_{j=0}^{m-1} f(t_{j,0}) \int_{\Omega_j} \mathrm{d}x = \sum_{j=0}^{m-1} f(x_{j,0}) \mu(\Omega_j) \tag{97}$$

2.3.2 Improving convergence for numerical quadrature

One of the sources of error for numerical integration comes from the fact that the specific choice of sampling locations affects the approximation error, thereby having a high potential of affecting the integration as well (and certainly at a minimum making it harder to express tight bounds on error). Much research has gone into this aspect, which culminated into a general theory calls *Gaussian quadrature*, that focuses on rules to determine the best placement of sampling locations for integrands of known specific forms. However, the general process requires some information on the specific analytical form of the integrand, so it is of limited usefulness for functions that don't have an analytical expression.

As we have observed before, it turns out that it is often the case that f is locally well approximated by a line segment, in which case the so-called *midpoint rule* can be surprisingly effective: the idea is simply to do an order n Riemann sum θ_n^{mid} over our function where the sampling locations t_i are in the middle of their corresponding interval. With a minimum of thought it's easy to see that the expected integration error in this case is

$$\left| \theta - \theta_n^{\text{mid}} \right| \le \frac{(b-a)^3}{12n^2} \sup_{x \in (a,b)} |f''(x)| \tag{98}$$

further thoughts in this general space will be picked up again in the discussion of Monte Carlo integration, in Section 2.4.4.

As a fairly commonplace inspiration coming from the concept above, an idea that has been around for a long time is to introduce adaptive sampling in the numerical integration process. The bound above can be tightened as follows:

$$\left| \theta - \theta_n^{\text{mid}} \right| \le \frac{(b-a)^2}{12n^2} \sum_{i=0}^{n-1} \sup_{x \in (x_i, x_{i+1})} (x_{i+1} - x_i) \, |f''(x)| \tag{99}$$

which is derived by localizing the general bound from before to the various single intervals. So it's conceivable that given a means to estimate the magnitude of the second derivative of f, one could decide to refine certain segments of a partition and not others, in order to achieve a given budget in integration error. This is the general area of study for *adaptive quadrature*.

Sometimes it so happens that the sampling values t_i end up being close enough to one another to induce excessive stress on the numerics of the implementation, and numerical errors start to affect the stability of the computation of the integral. In this case, *Romberg's method*, introduced in [Romberg, 1955], was developed on the basis of *Richardson extrapolation* [Richardson, 1911]. Richardson extrapolation was conceived to avoid having to work with and excessively tight spacing of sampling locations. Further reductions in the number of evaluations of f can be obtained using rational interpolation as outlined in [Bulirsch and Stoer, 1967].

It is sometimes necessary to estimate integrals to an extremely high degree of precision. The currently best-performing method in this space is called *Double Exponential formula*, and was proposed in [Takahasi and Mori, 1974]. The somewhat counterintuitive process entails expanding the integration domain to the entire real line using the specific change of variable $x = \tanh(\frac{\pi}{2}\sinh t)$, and using trapezoidal integration to compute the result. The improvement in convergence speed and accuracy comes from the fact that the change of variable is effectively conditioning for the better the behaviour of the original function f, especially around large derivatives and singularities at the endpoints.

2.3.3 Integrals in higher dimensions

Turning our attentions to domains of higher dimension the formal structure of the discussion so far does not change. There are however two complications that arise: the first is about the structure of the integration domain, and the second is about the number of function evaluations needed to compute our integral. Usually the structure of integrations domains relevant for problems in graphics is not a problem, as in practice all that is needed is to remap the domain to $\Omega = [0, 1]^d$ and potentially extend the function a little so to evaluate to 0 outside the original domain of definition. However the number of evaluations needed for the algorithms discussed this far, when ported to higher dimensional domains, grows at the same exponential rate: the expectation is that if a function requires n evaluation in one-dimensional form, a d-dimensional integral of a similarly structured function would require n^d evaluations. For example 100 samples in 10 dimensions would require 10^{20} evaluations, hardly a comfortable amount of work for current (and likely future) hardware. We observe 10 dimensions on a path tracing path are likely to not be enough to cover the needs of 3 vertices. The reference estimation for the integration error of a high dimensional quadrature rule is

$$E = O\left(\frac{1}{n^{\frac{r}{d}}}\right) \tag{100}$$

under the assumption that the integrand f has r continuous derivatives.

There is an approach introduced in [Smolyak, 1963] based on *sparse grids*, providing a structure which can be thought of as approach the integral one dimension at a time, as one would have applying Fubini's or Tonelli's theorems on the change of order of integration in the analytical setting [Apostol, 1991, Fubini, 1907, Rudin, 1953, Spivak, 2006, Tonelli, 1909]. In this manner d one-dimensional integrations are performed computing through the recursive operator formula

$$Q_l^{(d)}[f] = \left(\sum_{i=1}^{l}\left(Q_i^{(1)} - Q_{i-1}^{(1)}\right) \otimes Q_{l-i+1}^{(d-1)}\right)[f] \tag{101}$$

where $Q_i^{(d)}$ is the integration operator resulting, and $Q_i^{(1)}$ is a one-dimensional quadrature rule discretized over $O(2^i)$ sampling locations. With this framework, the error estimate for a function f with r continuous derivatives is then estimated as

$$E = O\left(\frac{(\log n)^{(d-1)(r+1)}}{n^r}\right) \tag{102}$$

2.4 Monte Carlo integration

The Monte Carlo method was proposed by Nicholas Metropolis and Stan Ulam, in [Metropolis and Ulam, 1949]. The idea is that the estimator θ_n to our integral over domain Ω can be simply

$$\theta_n = \frac{\mu(\Omega)}{n} \sum_{i=0}^{n-1} f(\xi_i) \qquad \xi_i \in \Omega \tag{103}$$

which would look suspiciously close to the Riemann integral in the special case in which the sampling locations x_i are uniformly spaced (thereby lying at distance $\frac{\mu(\Omega)}{n}$), if it wasn't for the fact that the function f is not sampled with uniform spacing x_i, but at n locations $\xi_0, \dots, \xi_{n-1}$ chosen as the realization of n independent and uniformly distributed random variables $X_0, \dots, X_{n-1}$. All things considered, it is quite possible that some readers would find equation (103) to be instead most similar to the Lebesgue integral, equation (63). This would be an insightful observation, we'll come back to that.

Given that we consider θ_n an approximation of the value θ we're actually looking for, it come natural to wonder how good this approximation actually is. The *estimation error* is thus introduced and defined as expected[10]:

$$\varepsilon_n = \theta - \theta_n \tag{104}$$

and this lets us reason about the convergence of the sequence θ_n to the value θ in terms of the estimation error ε_n. The first result is naturally whether we can say that the error decreases for growing values of n and of course it does, but in a probabilistic sense. Indeed it is an immediate consequence of the strong law of large numbers that:

$$P(\lim_{n\to\infty} \varepsilon_n = 0) = 1 \tag{105}$$

One point to note is that in Monte Carlo integration there is a specific focus on discussing absolute error metrics instead of relative. This is because of how the absolute value of the function to be integrated affects the introduction of error into the estimator: large function values make small sampling defects have a larger effect on total error than small function values. This observation is the starting point for most variance reduction techniques, starting from importance sampling, in which the sampling density of the function is increased around large values and decreased around small ones.

A second property of the integral estimator θ_n is whether it has *bias*, that is whether the expected value $E(\theta_n)$ for constant n across different realizations of the random variables $X_0, \dots, X_{n-1}$ differs from θ. Again, from the strong law of large numbers we can see that the estimators θ_n constructed per equation (103) are *unbiased*, that is to say their bias is 0, and $E(\varepsilon_n) = 0$: this is a consequence of the random variable X_i being independent and identically distributed and the associativity of the sum. In fact, you can see that equation (103) for n that grows towards infinity can be thought both as the definition of θ, as well as the summation needed to compute the average value of a large number of realizations of θ_n, confirming how in the end the two expressions must have the same value.

2.4.1 Estimation error in Monte Carlo integration

The third property of the estimation error ε_n that we will discuss is its magnitude: from the central limit theorem we can show that ε_n must be normally distributed with variance σ^2:

$$\varepsilon_n \sim \frac{\sigma(f)}{\sqrt{n}} \nu(0,1) \tag{106}$$

where $\nu(0,1)$ is a normally distributed variable (of 0 mean and unit variance) and

$$\sigma(f) = \sqrt{\int_\Omega \left(f(x) - \theta\right)^2 \mathrm{d}x} = \sqrt{\|f\|_2^2 - \theta^2} \tag{107}$$

[10]However, do note it is customary to have the expression for the estimation error to be signed, the reasons will become clear in what follows

being the *standard deviation* of f. We will also occasionally use in the text the notion of *variance* of f which will denote $\beta(f) = \sigma^2(f)$. As we were saying, this implies Monte Carlo error is inherently $O(\sigma(f)/\sqrt{n})$ under the hypothesis that the sampling locations x_i are independent and identically distributed. The error estimation in Monte Carlo is a consequence of the Central Limit Theorem, so it's effectively saying that ε_n is within a certain number times the magnitude of the process's standard deviation as a consequence of it being normally distributed. In other words there is about 1% probability (measured across all possible combined realizations of the random variables involved) that the error will be higher than 3σ, for example. By way of contrast we recall the Newton-Cotes error: if the function is well behaved enough (and as we've seen from Runge's function, this can be a counterintuitive notion) the error bound for sampling locations distributed on a regular lattice is deterministically a quantity $O(n^{-\frac{m}{d}})$ where m is the order of the Newton-Cotes formula and d is the dimensionality of the space we're integrating over.

From this estimate you would conclude that for low dimension domains, in particular if $d < 2m$, Newton-Cotes has an advantage over Monte Carlo, whereas if $2m < d$ the opposite is true. In particular, for dimensions $d \leq 3$ and well-behaved functions f (for example f for which derivative magnitudes or Fourier coefficients decay quickly) it is quite clear that classic, non-stochastic integrations schemes can have an efficiency advantage.

2.4.2 Interlude: random number generation

Random numbers should not be generated by a method chosen at random

Donald E. Knuth

There is obviously a question regarding the quality of the random number generators used to implement a Monte Carlo integrator in software, because the hypothesis of independent and identical distribution is to key to prove the estimation error bounds. It turns out (see for example [Caflisch, 1998]) that the lowly linear congruential generator with a decent implementation, say from [Press et al., 2007] or [Knuth, 1997], is viable up to maybe 10^9 samples or so (in other words roughly for 32 bits of precision), and that once the sample counts go past 10^{12} or so, things should be taken seriously. Do note that at these counts some error propagation analysis is probably in order, to make sure that the implementation of the numerics and accumulation algorithms themselves are stable (in other words it's not going to be very useful to have fantastically well distributed random numbers if the numerics are implemented carelessly). For counts this high, better generators should be used, besides the now-ubiquitous *Mersenne Twister* from Matsumoto, introduced in [Matsumoto and Nishimura, 1998], one could maybe consider Marsaglia's *Monty Python* or *Ziggurat* generators, introduced in [Marsaglia and Tsang, 1998] and [Marsaglia and Tsang, 2000] respectively.

There are indeed many questions to be asked as to how the concept of *independent random variable* can or should be reconciled with a fixed numerical procedure such as a linear congruential generator. However, we'll postpone our thinking in this space until we're past our discussion of quasi-random sequences, later in the section.

2.4.3 Variance reduction techniques

We now understand the most important aspect of Monte Carlo integration is keeping the variance σ^2 of the integral estimator θ_n as low as possible, as that's the true key for an accurate estimation of θ in a given computational budget (that is, a given maximum value for n). One perspective that may not be completely obvious at first is to envision θ as the value that minimizes σ: after all we know that $\theta = \theta_n + \varepsilon_n$, and we have an expectation that ε_n be effectively a quantity $O(\sigma)$ (because as we said, we have little control on n). Then it follows that θ_n will of course be closest to θ when σ is at its lowest. There are a few classical methods to do this, which are described in what follows

Antithetic variables A surprisingly effective idea for domains that are symmetric around the origin is for each sampling location x_i to actually use both x_i and $-x_i$. The effect of this method is to make the standard deviation σ_a of the estimator θ_n^a equal to σ^2, which is particularly good news if the situation we were starting from was otherwise achieving already small values of σ. The way this works is easy to see: all we need to do is expand f in a Taylor series around 0 and then apply the change of variable $x_i = \sigma\hat{x}_i$:

$$f(x_i) = f(0) + f'(0)\sigma\hat{x}_i + O(\sigma^2\hat{x}_i^2) \tag{108}$$

when we go to compute our estimator using both antithetic variables x_i and $-x_i$, our integral is effectively

$$\int_{\Omega} f(x)\,\mathrm{d}x = \int_{\Omega^+} f(x) + f(-x)\,\mathrm{d}x \tag{109}$$

when the Taylor expansion is substituted for $f(x)$, the first order terms are equal and opposite and cancel each other out. Now given that the 0-th order terms contribute no variance, because they don't depend on x_i, the standard deviation has now become a quantity of order σ^2.

Control variates Another technique that is extremely effective can be employed when we have available a second function $g(x)$ for which we know that $|f(x) - g(x)|$ is small and either $\int g(x)\,\mathrm{d}x$ is known exactly, or maybe it has a standard deviation $\sigma(g)$ much smaller than $\sigma(f)$. In this case we can write

$$\int_{\Omega} f(x)\,\mathrm{d}x = \int_{\Omega} f(x) - g(x)\,\mathrm{d}x + \int_{\Omega} g(x)\,\mathrm{d}x \tag{110}$$

where the term on the left has high variance, but both the terms on the right have been carefully constructed to have low variance, resulting in a new estimator with low variance.

An interesting point to draw attention to is that it is actually the very fact of $|f(x) - g(x)|$ being small in magnitude that forces $\sigma(f-g)$ to be small: it follows directly from the definition of variance in equation (107) that if the function's oscillation around its average are small, the variance has now way to become large, but just as well the standard deviation and the expectation operators commute with the product with a scalar: $E[\alpha f] = \alpha E[f]$ and $\sigma(\alpha f) = \alpha\sigma(f)$.

Stratified sampling The idea of stratification comes from the intuitive observation that deviation from the local average tends to be smaller than deviation from a global average. More precisely: if the integration domain Ω is partitioned into m subdomains $\Omega_1, \ldots, \Omega_m$ (in this context called *strata*), the expression of the variance over the full domain can be compared with the variance over the m subdomains as follows:

$$\sigma^2(f) = \int_{\Omega} (f(x) - E_{\Omega}[f])^2\,\mathrm{d}x \geq \sum_{k=0}^{m} \int_{\Omega_k} (f(x) - E_{\Omega_k}[f])^2\,\mathrm{d}x = \sigma^2_{\text{stratified}} \tag{111}$$

the inequality is the mathematical version of the intuition in the opening paragraph: it comes from the fact that each expected value for f over the stratum $E_{\Omega_k}[f]$ minimizes the variance of f over that one stratum.

These formulas are possibly a bit opaque about one very important detail: the inequality holds as a consequence of the sampling density being constant over Ω as a whole (as one would expect) which in particular means that the sample count in each stratum Ω_k has to be proportional to its measure $\mu(\Omega_k)$ (as one might instead overlook): in this case the sampling distribution is called *balanced*. On the other hand it possible to measure the empirical variance during sampling of the various strata, and adjust sampling densities until a relation of some interest between the various σ_k is achieved. Naturally when doing this, the weighting of the various samples taken from f needs to be adjusted accordingly.

Importance sampling Given a function $p(x)$ so that $\mu(p^{-1}(0)) = 0$ we can obviously write

$$\int_{\Omega} f(x)\,\mathrm{d}x = \int_{\Omega} \frac{f(x)}{p(x)} p(x)\,\mathrm{d}x \tag{112}$$

which may at first seem a simple algebraic manipulation of terms. However, if $p(x)$ is additionally a probability distribution, one can think of the combined $p(x)\,\mathrm{d}x$ as a change of variable operation on the integral, an operation that effectively alters the density of x. Now, say that X is a random variable distributed so that its sampling locations x_i have density according to p, if one were to compute a Monte Carlo estimator θ_n^p of this variable, the probability of sampling each $p(x_i)$ would need to be balanced out:

$$\theta_n^p = \frac{1}{n} \sum_{i=0}^{n-1} \frac{f(x_i)}{p(x_i)} \tag{113}$$

and this estimator has standard deviation

$$\sigma_p = \sqrt{\int_\Omega \left(\frac{f(x) - \theta}{p(x)}\right)^2 p(x)\,\mathrm{d}x} \tag{114}$$

The first observations is that if the ratio $f(x)/p(x)$ is a constant we obtain $\sigma_p = 0$ (or if the ratio has small amplitude we have $\sigma_p \leq \sigma(f)$, similar argument as in the case of control variates).

The second observation is that $p(x)$ can be employed to concentrate the sampling around difficult areas, or simply areas of Ω that for a reason or another happen to produce high variance. This second point of view is effectively an interpretation of importance sampling as the continuous equivalent of stratified sampling.

2.4.4 Quasi-Random sequences and Monte Carlo

Randomness is a negative property: it's the absence of any pattern

Richard W. Hamming

Up to this point, the critical hypothesis underlying all of our conclusions is that the sampling locations x_i are realizations of a set of random variables X_i which are independent and identically distributed (iid). This is the fundamental underpinning of our variance analysis (in equation (106) and following), because the whole story follows as a direct application of the Central Limit Theorem. Given that this was going well, the next idea is to stop using iid variables and instead move to a specific way of choosing dependent sampling locations called *quasi-random sampling*, the classic reference on the subject being [Niederreiter, 1992].

The impetus for the idea of moving away away from random sampling towards quasi-random sampling stems from the observation that the distribution of samples from Monte Carlo samplers is (arguably by design) uneven in density. As we said the task at hand is to integrate a function f over the interval $[0, 1]$, so we can compute Riemann sums taking an increasing sequence of samples $0 = x_0, \dots, x_n = 1$, and then multiplying the extent of the i-th subinterval

$$\Delta_i = x_{i+1} - x_i \tag{115}$$

by the value $f(t_i)$ of the function at some sampling location $t_i \in [x_i, -x_{i+1}]$. In Monte Carlo this process is simplified by using random values for the sampling locations t_i and assuming a corresponding supporting subinterval of constant size $\frac{1}{n}$. So if we look at a specific instance of the integral estimator with n samples, we can pull it back into the perspective of a Riemann integral placing our partition locations x_i inbetween our sampling locations t_i: $x_i = \frac{t_{i-1}+t_i}{2}$ (for $0 < i < n$). So now we can look at the difference between the Monte Carlo estimator θ_n^{MC} and its corresponding Riemann sum θ_n^R computed through this specific associated partition:

$$\theta_n^{MC} = \frac{1}{n}\sum_i f(t_i) \qquad \theta_n^R = \sum_i f(t_i)\Delta_i \tag{116}$$

and it's easy to see how defining

$$\delta_i = \Delta_i - \frac{1}{n} \tag{117}$$

we can show that

$$\theta_n^R - \theta_n^{MC} = \sum_i f(t_i)\delta_i \tag{118}$$

which is a quantity that in some ways lines up with the intuitive notion of noise in Monte Carlo integration. This shows that especially for regions where the function is large in magnitude $|f(t_i)|$ it is quite important that the spacing of our samples t_i be as close as possible to its corresponding target density $\frac{1}{n}$. There is a very important kind of functions for which there is no difference in the two methods: constant functions. In this case we have $f(t_i) = f(t_j)$ for any two i, j so that the expression for $\theta_n^R - \theta_n^{MC}$ becomes $f(t_0)\sum_i \delta_i = 0$, because the δ_i must sum to 0 by construction.

As much as the Δ_i multipliers do a better job than a constant $\frac{1}{n}$ would do at estimating interesting areas to use in our Riemann sums, it remains the problem that f could have a very large or a very small amount of travel over the subinterval $[x_i, -x_{i+1}]$ around t_i. If the function is well approximated by a line over the range, given that t_i is in the center of the subinterval, we would be effectively using some kind of trapezium rule, or midpoint rule, keeping the integration error under control. However in the common case where a straight line is not a good enough approximation of f, such as near maxima or minima, there is plenty of chance for the error to increase substantially. This is where switching perspectives and considering the situation from the point of view of Lebesgue integration might provide useful insight: the idea here is to reason about the difference between our subinterval $[x_i, -x_{i+1}]$ and the actual support for $y_i = f(t_i)$, which is $f^{-1}([y_i - \varepsilon, y_i + \varepsilon])$. It's clear that the largest the difference in measure between the two sets, the worse the quality of our integral estimator will be. From a practical point of view, the expectation is that although a global inverse won't be available, even as a coarse approximation, arguments could be made for good fits valid for moderately-sized neighborhoods of y_i.

Random sampling is carefully constructed so that it won't distribute the sampling locations t_i to have even density: in fact the reduction of estimation error in Monte Carlo integration comes from the late regime established for large values of n in which it's not so much the case that the distribution is effectively even density (as we said, this is never the case), but much more that the function $f(t_i)$ is sampled so finely that in the support spanned by a few nearby sampling locations it ends up being very nearly constant.

This is where the properties of quasi-random sequences come to help: a quasi random sequence is a structure very carefully constructed in such a manner that the average density of the sampling locations t_i is instead very even, in other words, it's a sequence built so that the various δ_i have a very high probability of being very small.

2.4.5 A quality metric on sequences

As much as sequences are not independent, we need a sense of how effective they are at sampling the given domain Ω that we want to integrate our functions over. This quality metric is called *discrepancy*, it is a property of the process and is defined as follows: first we need a sense for a given subset $J \subseteq \Omega$ of the quality $R_n(J)$ of the distribution of n sampling locations

$$R_n(J) = |\mu(J) - \theta_n^{MC}(\chi_J)| \tag{119}$$

where $\chi_J(x)$ is as before the characteristic function for J (see equation (55)), which implies that $\mu(J) = \int_\Omega \chi_J(x)\,\mathrm{d}x$. The other term $\theta_n^{MC}(f)$ is the Monte Carlo estimator for the integral of the function f obtained using our given sample sequence. Applying the definition to $\chi_J(x)$ one obtains

$$\theta_n^{MC}(\chi_J) = \frac{1}{n}\sum_{i=0}^{n-1}\chi_J(x_i) = \frac{\#(\{x_0, \dots, x_n\} \cap J)}{n} \tag{120}$$

with $\#(X)$ being the function that counts the number of points in X. In other words, R_n quantifies the error that the sequence itself injects in estimating the measure of the subset J. Of course the performance on one given set is not the most useful way of measuring the quality of a sequence, so the discrepancy is actually defined in a couple different ways: one is D_n, being the worst case for R_n over all rectangles $\mathcal{R}(\Omega)$ contained in Ω

$$D_n = \sup_{J \in \mathcal{R}(\Omega)} |R_n(J)| \tag{121}$$

You can think of this as the L^∞ norm of R_n as a function over $\mathcal{R}(\Omega)$, and that makes good background for the other measure for discrepancy T_n, being the L^2 norm (the *RMS integral* if you will) of R_n:

$$T_n = \sqrt{\int_{J \in \mathcal{R}(\Omega)} R_n^2(J)} \tag{122}$$

For completeness it's worth mentioning that there exist so-called *star* forms of both metric, defined by integrating over the set $\mathcal{R}^\star(\Omega)$ of rectangles pinned around a fixed point (as we said, Ω is usually the d-dimensional unit interval $[0,1]^d$ and the pinning happens at 0)

$$D_n^\star = \sup_{J \in \mathcal{R}^\star(\Omega)} |R_n(J)| \qquad T_n^\star = \sqrt{\int_{J \in \mathcal{R}^\star(\Omega)} R_n^2(J)} \tag{123}$$

Now that we have some machinery in hand to discuss qualities of various sequences of sampling locations, we can give a names to various traits of our sampling sequences. First off, we will call a sequence t_j *uniformly distributed* if

$$\lim_{n\to\infty} D_n(t) = 0 \tag{124}$$

whereas we will say the process is *quasi-random* if

$$D_n(t) \leq c\frac{(\log n)^k}{n} \tag{125}$$

for given k and c positive, not dependent on n, but possibly dependent on the dimension of Ω. The common quasi-random sequences have $k = d$.

2.4.6 Error bounds

Now that we have a quality metric on the sequences themselves, we can give some upper bounds on the expected estimation error, the most important bound being the *Koksma-Hlawka inequality* [Hlawka, 1961]:

$$\varepsilon_n(f) \leq \mathrm{V}_{\mathrm{HK}}[f] D_n^\star \tag{126}$$

where $\mathrm{V}_{\mathrm{HK}}[f]$ is the *total Hardy-Krause variation* of f. This bound is in general very conservative and the proof of it in dimensions $d > 1$ is rather technical, due to complications that arise when generalizing the concept of total variation to higher dimensions, a detailed modern treatment is found in [Owen, 2005].

However the fundamental intuition of the inequality, which is the original theorem from Koksma found in [Koksma, 1943], is much simpler to reason about. For one-dimensional functions, the *total variation* over an interval $[a, b]$ is

$$V_a^b[f] = \int_a^b \left|\frac{\mathrm{d}}{\mathrm{d}x} f(x)\right| \mathrm{d}x \tag{127}$$

assuming of course the first derivative of f is absolutely integrable. There exist expressions for the total variation operator suitable for more general functions, these are important because the simple definition given here has trouble even with simple jump discontinuities, for example. The bound is then simply a general form of our statement in equation (118): things cannot go any worse than the whole potential travel of the function over the integration domain multiplied by the measurement error of the domain's measure. As a matter of fact it's quite intuitive to see how this bound is extremely conservative, as it can only be attained if the function were to travel or oscillate with all its might in every single region where the support estimation induced by the sampling sequence was off. A beautiful rendition of the proof of the Koksma-Hlawka inequality is found in [Caflisch, 1998].

The Koksma-Hlawka bound is a key result in quasi-Monte Carlo theory, and has been studied and generalized in a very active field of research over the last century, but the issue is that it becomes far too conservative for functions that vary rapidly or even have a small number of jump discontinuities. In practice it is extremely common to have functions g for which $\varepsilon_n(g) \leq kD_n^\star$ for values of $k \ll \mathrm{V}_{\mathrm{HK}}(g)$.

The second result in this space is the *Woźniakowski identity* [Woźniakowski, 1991] which gives a closed form for the expected (squared) error of the estimator:

$$E(\varepsilon_n^2) = (T_n^\star)^2 \tag{128}$$

note that in this case the expectation for the error is computed integrating over the space of all sufficiently well-behaved functions using the *Brownian Sheet measure*. In other words the Woźniakowski identity can be loosely stated in natural language as: it's likely that the square of the integration error resulting from a given sequence is just about the square of the $T^\star$ discrepancy of that sequence.

Combining the Koksma-Hlawka inequality and the Woźniakowski bound you can see that

$$\varepsilon_n(f) \leq \frac{T_n^\star}{D_n^\star}\mathrm{V}_{\mathrm{HK}}[f] \qquad \text{for a "reasonably behaved" } f \tag{129}$$

2.5 Markov chain Monte Carlo

A different approach to approximating the value of an integral is based on Markov chains. In this context, similar to independent Monte Carlo, we are creating an ensemble of samples, and use these to compute the average of some quantity. In contrast to Monte Carlo, the samples are not independent: tentative samples x_t are instead drawn from a conditional distribution which has the memory of one sample: $T(x_t|x_c)$. Here, x_c is the current sample, the state variable of the Markov chain. x_t is dubbed *tentative* because it is not immediately turned into the next state x_c^{i+1}. There is great freedom to choose the so called transition probability density functions $T(x_t|x_c)$, but we need to take a bit of care to steer the Markov chain to reach the equilibrium distribution we want.

This can be facilitated by the Metropolis-Hastings acceptance probability:

$$a_{c\to t} = \frac{f(x_t)/T(x_t|x_c)}{f(x_c)/T(x_c|x_t)}. \tag{130}$$

The tentative sample x_t is accepted as the next state of the Markov chain only if $\xi < a$ for some uniformly distributed random number $\xi \in [0,1)$. This acceptance probability is designed to maintain *detailed balance*:

$$f(x_t)\cdot T(x_c|x_t)\cdot a_{t\to c} = f(x_c)\cdot T(x_t|x_c)\cdot a_{c\to t}, \tag{131}$$

intuitively taking care that the Markov chain stays in the right distribution once it reached it: we want the Markov chain to generate states with a distribution proportional to $f(x)$ in the limit. equation (131) states that all probability mass that flows out of x_t to x_c is counterbalanced by an equal amount of mass going the opposite way, coming back from x_c. In fact it would be enough to constrain that all probability flowing out of a state somewhere comes back from anywhere. This condition is called *general balance* but is harder to achieve, especially for continuous state spaces.

To make sure the Markov chain actually converges to the desired equilibrium distribution, there is one more condition we need to fulfill: *ergodicity*. This states that all of our state space needs to be accessible via mutation from any starting point. In practice this can easily be achieved by mixing in one mutation strategy which draws a completely independent sample.

There is an excellent introduction to Markov chain Monte Carlo (MCMC) in [Betancourt, 2017, p.11], where three phases of MCMC are discussed:

(1) convergence to the typical set, here we have strong bias.
(2) first round through the typical set, very fast convergence.
(3) draw more samples, take more rounds through the typical set, convergence slows down.

Now we are interested in the error bounds we can derive for this method. We know that in the third phase the central limit theorem holds again and we can compute a probabilistic error estimate as:

$$\varepsilon_{RMS} = \sqrt{\frac{\sigma^2}{ESS}}, \tag{132}$$

where *ESS* is the *effective sample size*. Intuitively this means how many real independent samples your sampling was worth. If the individual samples are too correlated, they contribute less to reducing the error of the estimate. For completeness, the *ESS* can be computed as

$$ESS = \frac{n}{1 + 2\sum_{l=1}^{\infty}\rho_l}, \tag{133}$$

where ρ_l is the lag-l autocorrelation of f over the history of the Markov chain.

Hamiltonian MC There are a couple of shortcomings in random, Metropolis-Hastings style Markov chains. First, the convergence to the typical set can be slow. It would be great if we could assist the chain in finding it faster. Second, if the mutation strategy is ill-adapted to the state space, there can be many rejections, resulting in chains getting stuck, skyrocketing autocorrelation and thus estimation error.

One elegant way of addressing these issues is *Hamiltonian Monte Carlo*. In a sense this a specialised mutation strategy. It comes, however, with a few properties that set it apart from Metropolis-Hastings. First, this method uses analytic derivatives to guide the mutation. Also, in a certain sense it maintains the energy of a state, such that no more samples have to be rejected in theory.

To achieve this, the method employs an analogy from classical mechanics. The random walk is performed in *phase space*, where the state is split into position and momentum. To keep the overall volume of the density measure around a sample before and after mutation constant, it is enough to maintain the Hamiltonian as invariant.

The Hamiltonian corresponds to the negative log joint target function in phase space. We denote the traditional target function as $f(q)$, and artificially extend it to phase space $f(q,p)$ by introducing a dependency on momentum p:

$$f(q,p) = e^{-H(q,p)}. \tag{134}$$

The Hamiltonian H is the sum of kinetic energy and potential energy, and can be written as:

$$\begin{aligned} H(q,p) &= -\log f(q,p) && (135)\\ &= -\log f(q|p) - \log f(q) && (136)\\ &= K(p,q) + V(q). && (137)\end{aligned}$$

We know from classical dynamics that Hamilton's equations will keep H constant. So all we need to do is simulate the corresponding differential equation to draw new samples.

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial H}{\partial p} = \frac{\partial K}{\partial p} \tag{138}$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial H}{\partial q} = -\frac{\partial K}{\partial q} - \frac{\partial V}{\partial q}. \tag{139}$$

Here, t is the integration time, i.e. going from one state to the next sample. To solve these equations, we require $\partial V/\partial q$, the derivative of our target function. We also need to provide a derivative of the kinetic energy we chose to augment our state space to phase space. This usually leads to very simple choices of K, in the simplest case even independent of q. For a more in-depth discussion also about implementation details we refer the reader again to the excellent summary by Betancourt [2017].

2.5.1 Discussion

For MCMC, the smoothness of the integrand is essential. This is more obvious for Hamiltonian Monte Carlo, where the derivatives explicitly go into the design of the mutation strategy. But the whole concept of small step mutations is based on an implicit smoothness assumption. Neither of the two methods help exploring highly fractured discontinuous function spaces with a large number of isolated islands. A similar problem arises for fractal functions. This could be the density of a cloud or the multi-scale displacement of the surface of rough rocks: the derivative at the finest scale will do nothing to guide the sampler into a good direction for larger scales.

Another issue with MCMC is that it usually does not incorporate the concept of an image. That is, the ensemble is generated to estimate an average, not to estimate a set of pixels uniformly well. This is often times visible as clumping artifacts or uneven exploration in image space.

Appendices

2.A Riemann on Riemann's integral

Follows a short extract from [Riemann, 1868], introducing the definition of integral. The excerpt has been translated into modern English, retypeset and the symbols designating the various quantities brought in line with the ones used in this document.

Proceedings of the Royal Philosophical Society at Göttingen
Vol. 13, pg. 101–103, 1868

On the representability of a function by a trigonometric series

Bernhard Riemann

Section 4

On the notion of the definite integral and the range of its validity

The vagueness which still prevails over some fundamental points on the study of the definite integrals forces us to provide some clarifications about the notion of definite integral and the range of its validity.

But firstly: What should be understood under $\int_a^b f(x)\,\mathrm{d}x$? We define this taking between a and b a sequence of values $a = x_0, x_1, x_2, \dots, x_{n-1}, x_n = b$ increasing in size and denote for the sake of brevity $\delta_i = x_i - x_{i-1}$, and by ε_i some positive real quantities. Then the value of the sum

$$\theta_n = \sum_{i=1}^{n} \delta_i f(x_{i-1} + \varepsilon_i \delta_i) \tag{140}$$

will depend on the choice of intervals δ_i and the magnitude of the various ε_i. If it has the property, however the choice of δ_i and ε_i, to approach in the limit for n going to infinity a fixed value θ to arbitrary small difference, as all δ_i become infinitesimally small, then this value is called $\int_a^b f(x)\,\mathrm{d}x$. If it doesn't have this property, then the expression $\int_a^b f(x)\,\mathrm{d}x$ has no meaning.

In several instances it has been attempted to assign meaning to this expression, and from all these extensions of the notions of definite integral is **one** from all mathematicians attained. Namely, if the function $f(x)$ takes values arbitrarily large when its argument x approaches a value c in the interval (a, b), in such a manner that the sums θ_n, regardless of the degree of smallness one prescribes to the δ_i, can attain any arbitrary value, we conclude that $\int_a^b f(x)\,\mathrm{d}x$ as defined above would have no meaning. If however, $\int_a^{c-\sigma_1} f(x)\,\mathrm{d}x + \int_{c+\sigma_2}^{b} f(x)\,\mathrm{d}x$ approaches a fixed limit as σ_1 and σ_2 become arbitrarily small, we will say that this limit will be the value of $\int_a^b f(x)\,\mathrm{d}x$.

Other definitions by Cauchy about the notion of definite integral (given for cases in which the value does not exist according to the fundamental concepts expressed here) may in separate classes of investigations be convenient. However these have not been introduced with appropriate generality and therefore, owing especially to their arbitrariness, are hardly suitable.

References

Tom M. Apostol. 1991. *Calculus* (2nd ed.). Vol. 1. John Wiley & Sons, Inc.

Michael Betancourt. 2017. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv e-prints*, Article arXiv:1701.02434 (Jan 2017), arXiv:1701.02434 pages. arXiv:stat.ME/1701.02434

Nicolas Bourbaki. 1966. *General Topology*. Hermann.

Roland Bulirsch and Josef Stoer. 1967. Handbook Series Numerical Integration. Numerical quadrature by extrapolation. *Numer. Math.* 9 (1967).

Russel E. Caflisch. 1998. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica* 7 (1998), 1–49. https://doi.org/10.1017/S0962492900002804

Manfredo P. do Carmo. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

Guido Fubini. 1907. Sugli integrali multipli. *Atti dell'Accademia Nazionale dei Lincei, Rendiconti, Serie Quinta* 16 (1907), 608–614. Issue 1.

Walter Gautschi. 1975. Norm Estimates for Inverses of Vandermonde Matrices. *Numer. Math.* 23 (1975), 337–347. Issue 4. https://doi.org/10.1007/BF01438260

Josiah Willard Gibbs. 1898. Fourier's series (in Letters to the editor). *Nature* 59 (29 Dec 1898), 200. https://doi.org/10.1038/059200b0

Josiah Willard Gibbs. 1899. Fourier's series (in Letters to the editor). *Nature* 59 (27 Apr 1899), 606. https://doi.org/10.1038/059606a0

Edmund Hlawka. 1961. Funktionen von beschränker Variation in der Theorie der Gleichverteillung. *Ann. Mat. Pura Appl.* 54 (1961), 325–333.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150. https://doi.org/10.1145/15886.15902

Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3 ed.). Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Jurjen Ferdinand Koksma. 1942–1943. Een algemeene stelling inuit de theorie der gelijkmatige verdeeling modulo 1. *Mathematica (Zutphen B)* 11 (1942–1943), 7–11.

Andrey Nikolaevich Kolmogorov and Sergei Vasilyevich Fomin. 1954, 1960. *Elements of the Theory of Functions and Functional Analysis*. Graylock Press. Translation available as 1957, 1961 Graylock Press, reprinted 1999, Dover.

Kazimierz Kuratowski and Andrzej Mostowski. 1967. *Set theory*. PWN.

Henri Lebesgue. 1904. *Leçons sur l'intégration et la recherche des fonctions primitives*. Gauthier-Villars, Paris.

John M. Lee. 2003. *Introduction to Smooth Manifolds*. Springer.

George Marsaglia and Wai Wan Tsang. 1998. The Monty Python Method for Generating Random Variables. *ACM Trans. Math. Softw.* 24, 3 (Sept. 1998), 341–350. https://doi.org/10.1145/292395.292453

George Marsaglia and Wai Wan Tsang. 2000. The ziggurat method for generating random variables. *Journal of Statistical Software* 5, 8 (2000), 1–7. http://www.jstatsoft.org/v05/i08;http://www.jstatsoft.org/v05/i08/rnorrexp.c;http://www.jstatsoft.org/v05/i08/updates;http://www.jstatsoft.org/v05/i08/ziggurat.pdf

Makoto Matsumoto and Takuji Nishimura. 1998. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator. *ACM Trans. Model. Comput. Simul.* 8, 1 (Jan. 1998), 3–30. https://doi.org/10.1145/272991.272995

Nicholas Metropolis and Stanley Ulam. 1949. The Monte Carlo Method. *J. Amer. Statist. Assoc.* 44 (Sep 1949), 335–341. Issue 247.

James R. Munkres. 2000. *Topology*. Prentice Hall, Incorporated.

Harald Niederreiter. 1992. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Art B. Owen. 2005. Multidimensional variation for quasi-Monte Carlo. In *International Conference on Statistics in honour of Professor Kai-Tai Fang's 65th birthday*, J. Fan and G. Li (Eds.).

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing* (3 ed.). Cambridge University Press, New York, NY, USA.

Lewis Fry Richardson. 1911. The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam. *Philosophical Transactions of the Royal Society A* 210 (1911). Issue 459–470.

Bernhard Riemann. 1868. Über die Darstellbarkeit einer Function durch eine trigonometrische Reihe. In *Abhandlungen der Königlichen Gesellschaft der Wissenschaften zu Göttingen*, Vol. 13. Göttingen, 87–132. https://books.google.com/books?id=PDVFAAAAcAAJ&pg=RA1-PA87

Werner Romberg. 1955. Vereinfachte numerische Integration. *Det Kongelige Norske Videnskabers Selskab Forhandlinger* 28 (1955), 30–36. Issue 7.

Walter Rudin. 1953. *Principles of mathematical analysis*. McGraw-Hill.

Walter Rudin. 1987. *Real and complex analysis*. McGraw-Hill.

Carl Runge. 1901. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. In *Zeitschrift für Mathematik und Physik*, Vol. 46. 224–243.

Sergey A. Smolyak. 1963. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Proceedings of the USSR Academy of Sciences* 4 (1963), 240–243.

Michael Spivak. 2006. *Calculus* (3rd ed.). Cambridge University Press.

Hidetosi Takahasi and Masatake Mori. 1974. Double Exponential Formulas for Numerical Integration. *Publications of the Research Institute for Mathematical Sciences* 9 (1974), 721–741. Issue 3. https://doi.org/10.2977/prims/1195192451

Leonida Tonelli. 1909. Sull'integrazione per parti. *Atti dell'Accademia Nazionale dei Lincei, Rendiconti, Serie Quinta* 18 (1909), 246–253. Issue 2.

Henry Wilbraham. 1848. On a certain periodic function. *The Cambridge and Dublin Mathematical Journal* 3 (1848), 198–201.

Henryk Woźniakowski. 1991. Average case complexity of multivariate integration. *Bull. Amer. Math. Soc. (N.S.)* 24, 1 (01 1991), 185–194. https://projecteuclid.org:443/euclid.bams/1183556256

3 A modular path sampling framework

Marc Droske, *Weta Digital*

It is remarkable how long some software architectures persist in practical use even in fast-moving industries with complex and ever-growing requirements such as visual effects production. Prime examples are Pixar's *RenderMan* [Christensen et al., 2018] and *mental ray* that have been used in professional movie production since the late 80's. Probably one of the most important factors for their success over the years is that they are both very flexible architectures that provide a high degree of programmability and could therefore adapt very well to new and often unforeseeable technical challenges.

Since path tracing made its way into the movie industry, the need for programmability has taken a different form. With the need to use BSDFs during light transport, programmable shaders in modern shading languages have shifted predominantly towards calculating the inputs of the BSDFs and defining the layering stack instead of actually calculating a "color" algorithmically as in previous programmable shading, which would have involved sampling, tracing rays and so on. In modern path tracing renderers, light-transport has become increasingly decoupled from shading by separating pattern generation from sampling decisions. We refer to [Pharr and Bala, 2018] for an overview of the current state as well as historical overview of existing production renderers.

This decoupling is crucial for a renderer to make informed decisions and to combine different techniques that are often delicate to implement already in isolation into a common framework. In this section we would like to give an overview over a range of considerations that flowed into the design of *Manuka*, Weta Digital's in-house production renderer.

Rendering software architectures are inherently complex and may have to satisfy a wide range of needs and hardware architectures. Thus, their design can be highly controversial and there is probably no one-fits-all approach. This section summarizes our perspective, built as a result of the work we have done on the movies that have been in production at Weta Digital, which in turn have shaped and influenced the development of our renderer *Manuka* [Fascione et al., 2018]. We would like to extend on the previous sections on the importance of good sampling decisions and to go through some relevant sampling techniques to motivate our modular design. Here, we focus the discussion on the different types of paths that can be generated rather than how the various techniques work in detail, which will be elaborated on in the following sections.

Due to the progressive nature of path tracing, simply increasing the amount of samples will always give the desired result, however the importance of good sampling strategies is recognized throughout the industry. Of course, the two concepts don't contradict each other: for example, the *Arnold* renderer from Solid Angle names itself a *brute-force* renderer but their research team also emphasises the importance of investing in good sampling strategies [Georgiev et al., 2018]. Even though denoising techniques for Monte-Carlo path tracing have become increasingly powerful in recent years, it goes without saying that avoiding noise in light transport by employing good sampling strategies remains the most effective and robust way of reducing render times.

3.1 Path integral formulation including participating media

As a quick recap of section 1.2, a general formulation of the rendering contribution as an integral over the space of paths that connect vertices on surfaces has been given by Veach [1998] and has later been generalized to include participating media [Pauly et al., 2000, Raab et al., 2006]. Path space can be described as the union of the spaces Ω_k of paths of fixed length:

$$\Omega := \bigcup_{k\in\mathbb{N}} \Omega_k \text{ where } \Omega_k := \{\bar{\mathbf{x}} = (x_0, \ldots, x_k) \,:\, x_i \in \mathbb{R}^3\} \tag{141}$$

In this section we enumerate the vertices starting from the light source. Let us consider a partition of three-dimensional Euclidean space into subspaces of codimension 0 and 1, i.e., volumes and surfaces: $\mathbb{R}^3 = \mathcal{V} \cup \partial\mathcal{V}$, where $\mathcal{H}^{d-1}(\partial\mathcal{V}) < \infty$ and $\mathcal{V}$ open and $\mathcal{H}^{d-1}$ denotes the $d-1$ dimensional Hausdorff measure. In practice $\mathcal{V}$ is a finite partition of volumes $\bigcup \mathcal{V}_i$, bounded by closed surfaces in $\partial\mathcal{V}$.

For a measurable subset $A \subset \mathbb{R}^3$, we define $\lambda(A) := \mathcal{L}(A \cap \mathcal{V}) + \mathcal{H}^{d-1}(A \cap \partial\mathcal{V})$, which is used to equip each of the spaces Ω_k with a product measure

$$\mu_k(A) := \int_A \mathrm{d}\lambda(x_0) \cdot \mathrm{d}\lambda(x_1) \cdots \mathrm{d}\lambda(x_k) \text{ for } A \subset \Omega_k. \tag{142}$$

The measure μ on path space Ω is the natural expansion on the disjoint subsets:

$$\mu(A) := \sum_{k \in \mathbb{N}} \mu_k(A \cap \Omega_k), \tag{143}$$

which allows to write the sensor response I_j corresponding to a pixel j compactly as the integral

$$I_j := \int_\Omega f_j(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{144}$$

where f_j is called the *measurement contribution function*, given as

$$\begin{aligned} f_j(\bar{\mathbf{x}}) = L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) T(x_0 \leftrightarrow x_1) \cdot \\ \prod_{i=1}^{k-1} f(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1}) T(x_i \leftrightarrow x_{i+1}) \cdot W_j(x_{k-1} \to x_k). \end{aligned} \tag{145}$$

Here W_j models the sensor response and L_e the emission function which is derived by simply unrolling the recursive nature of the rendering equation. At first glance, this looks exactly like *Veach*'s formulation, and indeed the only difference to his *surface-only* formulation are extensions of some of the terms to the new domain:

- The classic visibility function $V(x \leftrightarrow y)$ has been replaced by the transmittance

$$T(x \leftrightarrow y) := V(x \leftrightarrow y) e^{-\int_0^L \mu_t(r(t)) \mathrm{d}t} \tag{146}$$

  for a ray segment r of length $L = \|x - y\|$ going from x to y and μ_t being the *extinction coefficient*.
- The definition of the geometry term is extended to volumes:

$$G(x \leftrightarrow y) := \begin{cases} \frac{|\cos\theta_x| |\cos\theta_y|}{\|x-y\|^2} & \text{if } x, y \in \partial\mathcal{V} \\ \frac{|\cos\theta_x|}{\|x-y\|^2} & \text{if } x \in \partial\mathcal{V}, y \in \mathcal{V} \\ \frac{|\cos\theta_y|}{\|x-y\|^2} & \text{if } x \in \mathcal{V}, y \in \partial\mathcal{V} \\ \frac{1}{\|x-y\|^2} & \text{if } x, y \in \mathcal{V} \end{cases} \tag{147}$$

- The scattering distribution function is either surface BSDF $f_{\partial\mathcal{V}}$ or a phase-function $f_{\mathcal{V}}$

$$f(x_{i-1} \to x_i \to x_{i+1}) := \begin{cases} f_{\partial\mathcal{V}}(x_{i-1} \to x_i \to x_{i+1}) & \text{for } x \in \partial\mathcal{V} \\ \mu_s(x) \cdot f_{\mathcal{V}}(x_{i-1} \to x_i \to x_{i+1}) & \text{for } x \in \mathcal{V} \end{cases} \tag{148}$$

  where μ_s denotes the *scattering coefficient*.

In practice we also need to integrate over the spectral and time domains, i.e.,

$$I_j := \int_\Lambda \int_{t_0}^{t_1} \int_\Omega f_j(t, \lambda, \bar{\mathbf{x}}) \, \mathrm{d}\mu(\bar{\mathbf{x}}) \, \mathrm{d}t \, \mathrm{d}\lambda, \tag{149}$$

where all parts of the integrand may depend on these variables as well.

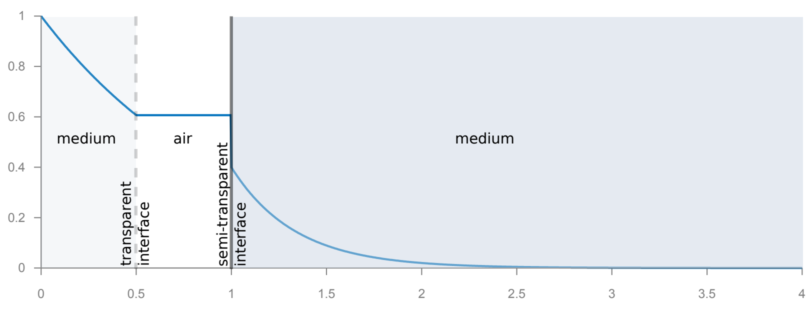


Figure 1: *Transmittance across different volumes bounded by fully transparent and semi-transparent interfaces. Free-path sampling typically aims to sample proportional to that profile, which involves alternatingly sampling within the media and potentially transparent surface boundaries until termination.*

3.2 Computing the integral

3.2.1 The Monte Carlo approach

As outlined in section 1.3 the integral I_j can be numerically computed by the Monte Carlo method, which draws samples from a probability distribution with density p and computes the estimate as

$$\hat{I}_j = \frac{1}{N}\sum_{i=1}^{N}\frac{f_j(\bar{\mathbf{x}}_j)}{p(\bar{\mathbf{x}}_j)}. \tag{150}$$

It is well-known that the quality of the estimator depends to a large extent on how well p represents the shape of f. Using a probability density that resembles the structure of f is called *importance sampling*, and it can easily be shown that if p is proportional to f the variance in the estimator can be eliminated completely.

Probably the most fundamental path sampling technique for paths is to start by sampling a vertex on the camera and extending it incrementally by local sampling, i.e., for a path of length $k+1$:

$$\begin{aligned} x_k &\sim p_{A,\text{sensor}}(\,\cdot\,) \\ x_{k-1} &\sim \overleftarrow{p}_{A,\text{sensor}}(\,\cdot\,|x_k) \\ x_i &\sim p_{\text{loc}}(\,\cdot\,|(x_{i+1},x_{i+2}))\text{ for }0\le i<k-1 \end{aligned} \tag{151}$$

hoping to reach a light source at x_0 (recall that we enumerate vertices starting from the light source). The path probability is then given as the product of initial and conditional probabilities:

$$\overleftarrow{p}(\bar{\mathbf{x}}) := p_{A,\text{sensor}}(x_k)\cdot\overleftarrow{p}_{A,\text{sensor}}(x_{k-1}|x_k)\cdot\prod_{i=0}^{k-2}p_{\text{loc}}(\,\cdot\,|(x_{i+1},x_{i+2})). \tag{152}$$

The local sampling stragegy p_{loc} can be chosen to resemble the product of the material response and the geometry term $G(x_i \leftrightarrow x_{i-1})$ by importance sampling the BSDF with a probability that is commonly given in projected solid angle measure, applying a ray-casting operator to get the next hitpoint and converting the probability to area measure. This technique, in a nutshell, is what is commonly referred to as *path tracing* and is the basic building block for many other techniques. More generally, to account for volumes and transparent interfaces, the ray-casting operator is replaced by *free-path sampling*, which samples a distance according to the transmittance profile (see figure 1). The end point can lie either in a medium with continuous probability, on a transparent volume boundary or the next opaque surface with discrete probability.

Completely analogously, the path can also be initiated by choosing a point on a light source, and extending the path at every interaction with a material and finally explicitly connecting to the sensor. Since this mimics how photons flow in the physical world, this technique is called *light tracing*.

Unfortunately, there are various factors that can make the integrand extremely complex in practice. Complex occlusion, high-frequency detail in geometry, highly directional-dependent material response, focused light sources and high dimensionality give raise to little hope of being able to importance sample the whole integrand in closed form. Indeed it is easy to construct extremely simple scenes with only very basic types of materials and without complex occlusion that are very hard to render with unbiased rendering algorithms that are commonly used in practice (see figure 2). Kollig and Keller [2000] refer to this as the *problem of insufficient techniques*.

3.2.2 Multiple Importance Sampling

The main challenge for efficient path sampling lies in devising sampling distributions that are roughly proportional to f and at the same time efficient to draw samples from. In practice, there are multiple sampling strategies available that are each tailored to resemble a specific part of f accurately while being less accurate on other parts. Veach has proposed a method for combining multiple importance sampling techniques into a single estimate, called *Multiple Importance Sampling*. This approach seeks to draw samples from multiple distributions and combine them in a weighted average that aims to automatically adjust to give preference to the technique that is best suited to sample a specific contribution:

$$\hat{I}_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{t\in T} w_t(\bar{\mathbf{x}}_j)\frac{f_j(\bar{\mathbf{x}}_j)}{p_t(\bar{\mathbf{x}}_j)}. \tag{153}$$

It remains an unbiased estimate as long the weights are chosen to ensure, firstly, that for any sample $\bar{\mathbf{x}}$ with a positive contribution the weights for all techniques sum up to 1:

$$\sum_{t\in T} w_t(\bar{\mathbf{x}}) = 1 \tag{154}$$

and, secondly, that a positive weight implies a positive sampling probability for the corresponding technique. A very useful weighting function is the so called *balance heuristic*, which is defined as

$$w_t(\bar{\mathbf{x}}) := \frac{p_t(\bar{\mathbf{x}})}{\sum_{t'\in T} p_{t'}(\bar{\mathbf{x}})}. \tag{155}$$

Veach showed that it is in some sense the best possible choice in the absence of further information. Intuitively, a technique is assigned a high weight if the probability of sampling a path is large compared to the other techniques, or in other words, if a technique has a difficulty of sampling an important contribution, an alternative technique that handles it better will take over: it's the basis for team-work between the techniques.

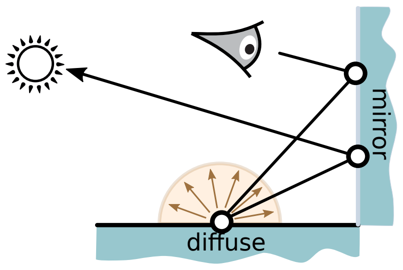
The beauty of MIS is that it allows to modularize light transport into separate sampling techniques for which the weighting adjusts itself automatically. It has hence become a key building block for probably all production path tracing renderers. However, as we will discuss in more detail, even though multiple importance sampling is conceptually simple (it only requires to compute the probability for each path that has been realized by a sampling technique), efficient and extensible implementations are far from being straight-forward and can be heavily constrained by the software architecture. As specifically tailored sampling techniques have proven to be very successful for variance reduction, an important design goal is to remain flexible with regard to adopting new sampling techniques in the future, while keeping the architecture easy to maintain.

3.2.3 Practical MIS

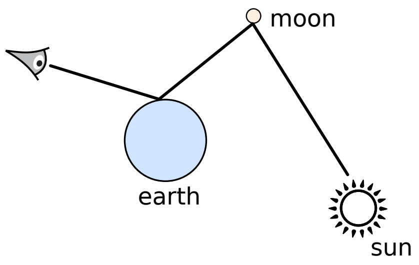
Let us go through some motivating basic examples. For simplified notation, let us define by $\vec{p}_i(\bar{\mathbf{x}})$ the probability of sampling vertex x_i conditioned on the knowledge of the previous vertices $x_j, 0 \leq j < i$ for sampling from the light source towards the sensor (light tracing) and analogously $\overleftarrow{p}_i(\bar{\mathbf{x}})$ for the reverse direction (path tracing). Therefore

$$\begin{aligned}\overleftarrow{p}(\bar{\mathbf{x}}) &= \prod_{i=0}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}) \qquad \text{(path tracing)},\\ \vec{p}(\bar{\mathbf{x}}) &= \prod_{i=0}^{k}\vec{p}_i(\bar{\mathbf{x}}) \qquad \text{(light tracing)}.\end{aligned} \tag{156}$$

Clearly, pure path tracing can become very inefficient in the presence of small, bright light sources, since it relies on the local path extensions to sample directions that will eventually reach a light. Instead, the last vertex can be



(a) The SDS problem.



(b) Indirect light from the moon.

Figure 2: *Two simple examples of notoriously difficult to render paths. Left: Caustics seen through specular bounces, often referred to as* SDS *(specular-diffuse-specular) paths. Right: Indirect lighting of sunlight bouncing of the moon onto the earth. Both are hard to sample even with full bidirectional path tracing.*

sampled by explicitly connecting to a light source vertex, which is commonly referred to as *next-event estimation*. The path probability

$$\overleftarrow{p}_{\text{NEE}}(\bar{\mathbf{x}}) = \prod_{i=1}^{k} \overleftarrow{p}_i(\bar{\mathbf{x}}) \cdot p_{\text{NEE}}(x_0|(x_1, x_2)) \tag{157}$$

where $p_{\text{NEE}}(x_0|(x_1, x_2))$, denotes the probability of sampling a point on an emitter, using some light sampling strategy that may for example depend on the position and vertex normal at x_1 and incoming direction to account for importance sampling of G and material response.

On-the-fly MIS Naturally, it is exploited in practice that the first k vertices of pure path tracing can be used to construct both a next-event estimation path and a path tracing path of length $k + 1$, by sampling a light source or local extension respectively instead of drawing completely independent samples. This means that a simple path tracing algorithm can be built around a loop of locally extending the local path an, at each iteration, applying next-event estimation to the current prefix. When a contribution is found, the MIS weight needs to be computed by evaluating both $\overleftarrow{p}_{\text{NEE}}(x)$ and $\overleftarrow{p}(\bar{\mathbf{x}})$ to plug into (155). Clearly, since these probabilities have a lot of terms in common, some of them cancel out and the weight can be computed based only on the local sampling probability and the light sampling probability, i.e., the history of the path is no longer needed.

The computation of the probability is often a light-weight by-product of sampling, which can be exploited by computing the MIS weight directly when the contribution has been found, allowing to discard the probabilities, which minimises storage because usually the probabilities are no longer needed afterwards. This is particularly appealing for GPU path tracing algorithms.

However, this approach comes with the cost of being difficult to extend to more sampling techniques or longer connections. Let's assume we have to come up with an imaginary sampling technique to sample paths that illuminate objects on the earth indirectly by moonlight scattered from the sun (figure 2(b)) and the VFX supervisor refuses to bake the light on the moon, as this would look completely unrealistic. We could do this more effectively than path tracing by instead sampling a moon-sun connection explicitly: a 2-vertex extensions is made by first sampling a vertex on the moon surface facing the earth and then connecting it to a random point on the sun's surface, i.e.,

$$\overleftarrow{p}_{\text{lunatic}}(\bar{\mathbf{x}}) = \prod_{i=2}^{k} \overleftarrow{p}_i(\bar{\mathbf{x}}) \cdot p_{\text{moon}}(x_1|x_2) \cdot p_{\text{sun}}(x_0). \tag{158}$$

Adding this technique into the mix for on-the-fly MIS computation means that more intermediate calculations need to be carefully stored to be able to compute the missing terms for each technique. For example, when pure path tracing hits the sun, the two previous vertices are required to be able to compute $p_{\text{moon}}(x_1|x_2)$.

This may appear to the reader as a very contrived example, since in conjunction with powerful light sampling (such as described in the next section) and firefly suppression techniques, path tracing paired with next-event estimation is still well-suited in a wide range of situations. Nevertheless this case may be less far-fetched than it seems:

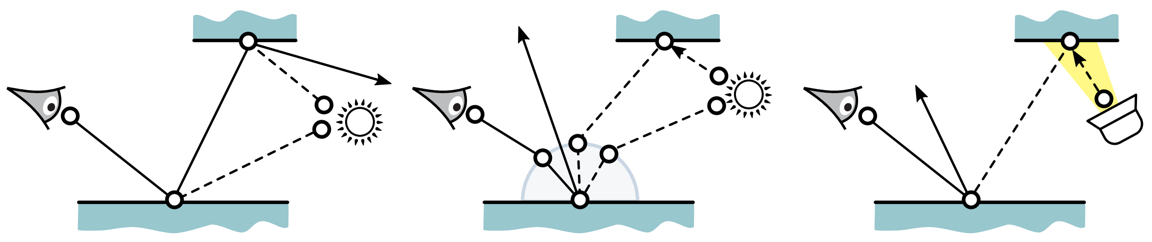

(a) NEE (b) Manifold NEE (c) BDPT

Figure 3: *Some examples of common path extension techniques on surfaces.*

some concrete real-world problems can be very specific and thus need very specific solutions, as we'll outline further in the next section.

3.3 Path sampling technique zoo

Before we discuss the design considerations for a production path tracing architecture, let us briefly categorize some techniques that have proven to be effective in movie production.

3.3.1 Path extensions

Fortunately, sampling paths of various lengths can be simply achieved by forward path tracing and applying different sampling techniques to find "good" extensions, *next-event estimation* (cf. figure 3(a)) being of course the fundamental prototype of that idea. Path tracing by local material sampling, followed by free path sampling along the sampled direction, can be seen as the master technique, which drives the main path forward, while the extension techniques create new branches that terminate at their corresponding end vertex.

A few other useful extension techniques that are not limited to extending by a single vertex are:

- Partial sub-paths starting from the light, generating using the light tracing algorithm, can be connected to partial eye-path to create a complete path, which is called *bidirectional path tracing* (BDPT) . An example connecting only a path of length two with the camera path is shown in its simplest form in figure 3(c), which can improve situations where light sources concentrate their emission onto a small region. Full BDPT first builds independent paths from the eye and from the light source and then creates new paths by connecting all their respective sub-paths. Each index that defines at which vertex a connection between the eye and light subpath is made corresponds to a sampling technique.
  Naturally, the techniques have various terms in common, which Veach [1998] exploited to propose a scheme to efficiently compute balance heuristic weights. It is however serial in nature and requires access to all vertices. van Antwerpen [2011] proposed a new scheme that computes specific quantities while the path is built, that can be used later to compute MIS weights by combining these quantities with local information. It is therefore much more GPU-friendly, however it requires a very specific algebraic structure that is not straightforward to combine with other techniques.
- Hanika et al. [2015] introduced **Manifold next-event estimation** that improves over classic next-event estimation by adding the ability to connect through refractive surfaces (figure 3(b)). The sub-path is found by an iterative solver that searches for a connection that is valid in the sense that it obeys the refraction constraint at each interior vertex. This can potentially add multiple vertices between the receiver and the light source and therefore requires access to multiple vertices and their differential geometry at each iteration. Speierer et al. [2018] extend this approach to connect to light sub paths.
- **Equi-angular sampling** [Kulla and Fajardo, 2012] is a volume sampling technique for single scattering, that first samples a point on a light source and then places a vertex on an open segment from an eye-path as shown in figure 4(a). By sampling the opening angle as spanned by the segment uniformly it effectively importance-samples the geometry term between the light source and the intermediate vertex. Note that the candidate segment is usually chosen to be the ray segment starting from a vertex in the direction of local

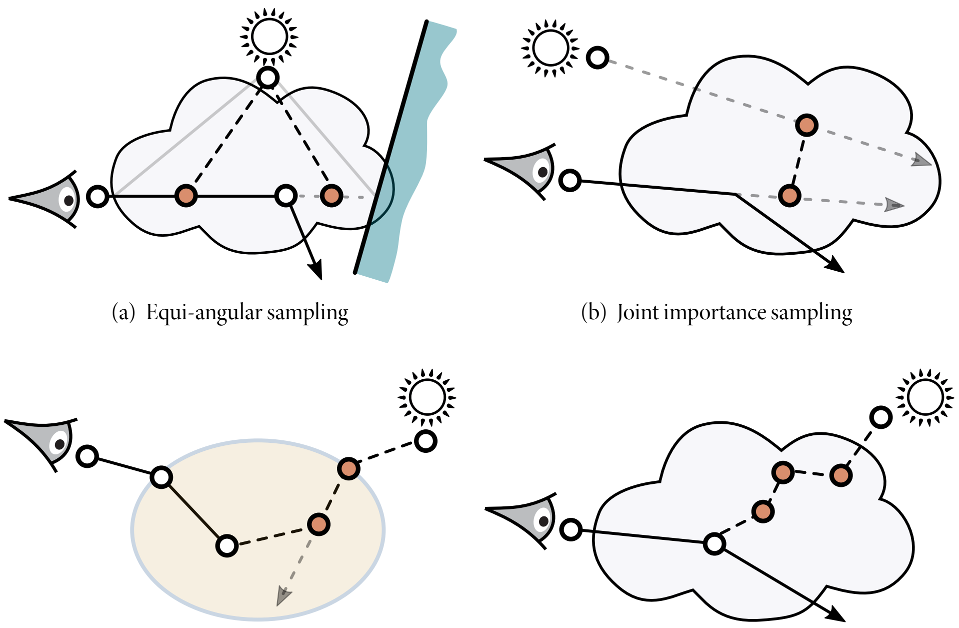

(a) Equi-angular sampling
(b) Joint importance sampling
(c) Subdivision NEE
(d) Multiple Vertex NEE

Figure 4: *Some examples of common path extension techniques on volumes.*

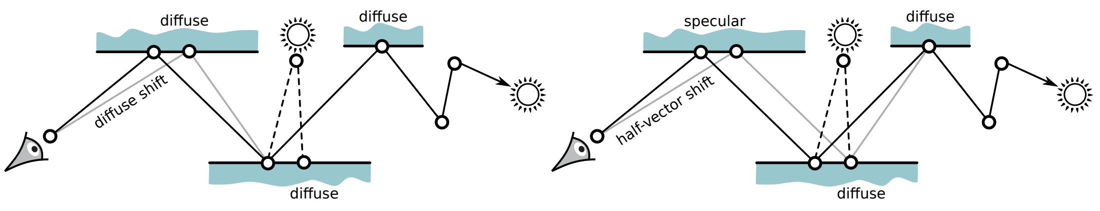
sampling as constructed by path tracing up to the next occluder. This may cause vertices to be sampled behind the vertex that was determined by free-path sampling during path tracing.

- Georgiev et al. [2013] takes this idea one step further by introducing **Joint importance sampling** which samples two vertices from a joint distribution for double-scattering as shown in figure 4(b).
- **Subdivision next-event estimation** [Koerner et al., 2016] aims at creating paths through refractive boundaries from within volumes. As depicted in figure 4(c), it creates paths by sampling an exit point on the surface, refracting backwards into the medium from the light source and connecting the open segment back to the main path using for example equi-angular sampling. Since by construction the extension obeys the material at the interface it is well suited to handle dielectric boundaries. However, since the construction starts from an interior vertex and adds one more vertex in the medium it is not able to sample single-scattering paths.
- **Multiple vertex next-event estimation** (MVNEE) is a technique for improving multiple scattering in relatively dense, forward-scattering media by constructing relatively long path extensions ($\sim 4-10$ vertices in practice) towards the light source as proposed by Weber et al. [2017]. This is achieved by perturbing a seed path deterministally by adding new vertices and shifting them in a way that aims to yield high throughput on the phase-function (figure 4(d)).

Clearly, there is some variation in the number of vertices being appended to the path between the different techniques. Even within a technique the number can vary, which makes it difficult to limit the amount of storage for MIS purposes or to compact the expression (155) a-priori.

3.3.2 Splitting

The efficiency of path tracing can be improved by introducing *splitting*, which can create multiple sub-branches at the current vertex by iterated local extension of the path. This can be beneficial in cases where the random walk has reached a region that is worth exploring further, however the splitting rates must be chosen with care to avoid a combinatorial explosion. Vorba and Křivánek [2016] have proposed a unification of path-termination via *Russian Roulette* and splitting using an estimate of incident radiance to fold into a contiuous stochastic decision process which determines whether to terminate or apply a splitting factor. Intuitively, the efficiency is improved because the prefix of the path can be shared by multiple different path realizations.



(a) Diffuse shift (b) Half-vector shift

Figure 5: *Shift mappings applied to the beginning of the path. Depending on materials on the vertices at the beginning of the path, different shift strategies may applied, which can cause the shifts to touch more vertices.*

3.3.3 Shift mappings

Path tracing generates new independent paths for each pixel. The longer the path becomes, the more likely that it becomes worthwhile to reuse available information to generate new correlated paths by shifting the path prefix. Bekaert et al. [2002] introduced the idea of reconnecting the endpaths to different primary vertices, which can in a sense be seen as splitting applied to the reverse direction.

Gradient-domain path tracing [Kettunen et al., 2015] uses primary shifts similar to those to compute estimates of screen-space derivatives of the measurement function, which are then used to reconstruct the image via the the Poisson equation. The quality of the shift plays an important role in the estimation of the gradient. While a simple primary shift (figure 5(a)) is effective if the primary vertex is diffuse, the shifts have to intrude deeper into the path via so-called *half-vector* shifts if the primary vertices are near-specular (figure 5(b)). They avoid that the throughput of the path breaks down at vertices at which the material is highly directionally dependent by ensuring the incoming and outgoing directions have the same half-vector. The depth of the shift can therefore be variable depending on the type of primary vertices. Generally, the quality of the result hinges upon the design of good shift mappings in various scenarios (for example, shifts in participating media) and naturally these mappings can also be used in an explicit path-reconnection approach as described earlier by employing the shift Jacobian to account for the density changes.

In the next session we will describe how shift mappings are also a natural candidate to introduce path-reuse for multiple cameras and distribution effects such as motion blur. In a previous course [Fascione et al., 2017] Heckenberg described how local path perturbations can be used to accelerate the convergence of depth of field via lens shifts.

Tessari et al. [2017] proposed to apply local shifts to explore the local neighborhood of a path and ensures good stratification by an explicit deterministic sampling scheme. By a similar heuristic, they aim at merging the shifted path as early as possible down the chain to the base path to reduce the amount of recomputation needed to evaluate the shift.

3.3.4 Markov-Chain methods et al.

Metropolis light transport (MLT, [Veach, 1998] and related sampling methods, such as energy-redistribution path tracing [Cline et al., 2005] and primary-sample space MLT [Kelemen et al., 2002] can be powerful techniques to solve complex light transport problems. They are based on applying various different mutation strategies that may modify an existing path at different depths.

They are yet rarely applied to movie production rendering, mostly because of correlation patterns that can cause noticable temporal instability and are harder to denoise than fully stratified sampling. However, this field remains an area of active research and recent papers have demonstrated very significant improvements to address some limitations, in particular introducing the ability to mix previously incompatible frameworks [Bitterli et al., 2017, Otsu et al., 2017, Pantaleoni, 2016] bringing this class of methods closer to practicality.

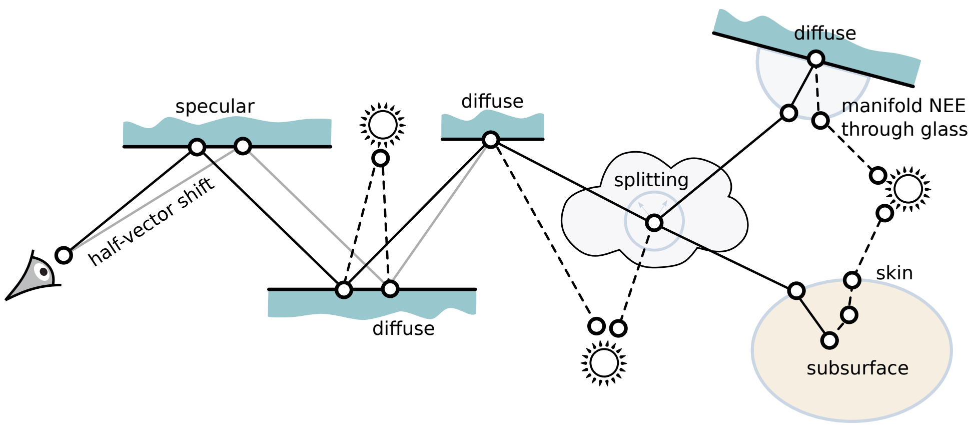


Figure 6: *Example of a combination of extensions, shifts and splitting.*

3.3.5 Multi spectral sampling

As mentioned earlier, chromatic noise can be effectively reduced by computing the measurement for multiple (n) wavelengths simultaneously. This can be achieved by constructing sampling techniques shifted in the spectral domain in such a way that the vertices remain the same [Wilkie et al., 2014]. The overall number of techniques thus multiplies by n: one for each shift, where one contains the "hero"-strategy, which behaves as classic spectral sampling, i.e., importance sampling as well as possible based on the given wavelength.

For computing MIS weights, this implies that the probabilities for sampling the entire path are relevant. Fortunately, this approach is straight-forward to implement: the main difference is that whenever a probability for some technique is computed, it has to be done for all n spectral channels, which is amenable to an efficient SIMD implementation.

3.3.6 Local techniques - path guiding

As mentioned earlier, extending the path by BSDF sampling is an example of a local sampling technique. More information of the light transport incident to the vertex can be exploited by modifying such a local sampling decision. Dwivedi sampling [Meng et al., 2016] and path guiding [Vorba et al., 2014] are some examples. They are typically rather minimally invasive since they can be encapsulated in the local sampling decision, which might become a stochastic decision between these different local techniques or emitter sampling and therefore fit very well into the path tracing eco-system. This might be an important factor as to why these methods have become quite successful in movie production (cf. also the course by Vorba et al. [2019] on *Path Guiding in Production* this year).

3.3.7 Discussion

The overview given above suggests that in practice, even though the fundamental path tracing algorithm is very simple, it would be beneficial to combine the various different techniques into one common framework. Production rendering has to deal with a very wide range of different scenarios and since a single shot may contain a combination of factors as complex lighting, volumetric scattering, distribution effects and dielectric interfaces a sampling situation as shown in figure 6 is not unrealistic. It might be worth noting that a significant portion of the path sampling techniques described has been developed specifically for the needs of rendering in movie production.

The lack of a sampling strategy can often be compensated to some degree by other means, for example discarding some hard to sample contributions such as caustics, manually modifying the scene or introducing approximations. Adaptive sampling can be quite effective by focusing the sample density in problematic areas.

However, being able to apply optimized sampling techniques to specific parts of the integral remains the most effective way to reduce render times [Pharr, 2019]. This directly leads to the question of how a light transport system can scale with the number of sampling techniques to evolve into an ecosystem of methods that can be configured as needed. Even though not all of the techniques might be necessary for all types of scenes it is, for the sake of

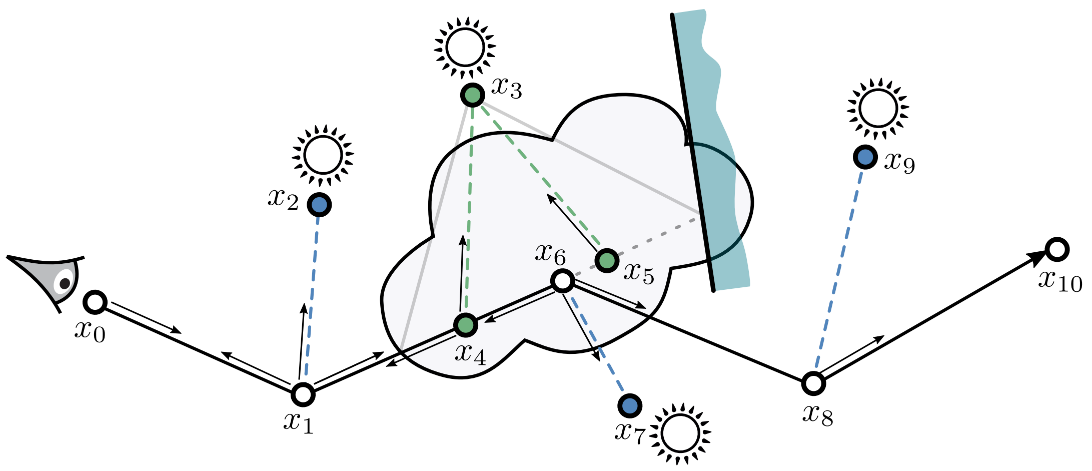


Figure 7: *Combining forward path tracing with next-event estimation extensions (blue) with realizations* (x_0, x_1, x_2), (x_0, x_1, x_6, x_7), $(x_0, x_1, x_6, x_8, x_9)$ *and equi-angular sampling (green) with realizations* (x_0, x_1, x_4, x_3) *and* (x_0, x_1, x_5, x_3).

usability, desirable in practice that all of the different methods can be combined arbitrarily. It is well-known that there is no guarantee that the combination of techniques with MIS has lower variance than any of the techniques in isolation. But even though it is not always optimal, it does improve robustness and removes part of the burden of having to fine-tune the optimal settings.

Practical production aspects When designing a rendering architecture the choice of methods to be combined is an important factor to keep in mind. Apart from scalability and performance, other practical production requirements need to be taken into account. For example, the ability to deduce different output channels (also known as *arbitrary output variables* AOVs, a *RenderMan* name) such as light groups, direct vs. indirect lighting or other path expressions such as caustics, and various specific channels for denoising purposes and compositing workflows. Hint: we haven't mentioned *biased* rendering techniques such as irradiance caching, not because errors would be unacceptable in principle, but because possible artifacts are harder to get under control, they limit the use of AOVs and are sometimes harder to combine with other techniques.

3.4 Modular light transport via the vertex graph

As outlined before, reusing partial paths to create new contributions as well as avoiding redundant computations are important ingredients to efficient path tracing algorithms. Some of this has been previously achieved by eliminating common terms and carefully breaking down the algebraic structure of the MIS weights into quantities that are efficient to compute (see for example [van Antwerpen, 2011, Veach, 1998]). However, this usually comes at the price of introducing a strong coupling between **all** techniques, that can lead to prohibitive growth of code complexity.

We are instead aiming for a modular plug-in architecture for sampling techniques, which requires the techniques to be independent from each other. The weighting scheme in section 3.2.2 already has a very modular structure. A sampling technique needs to do be able to do only two things: sample a path and evaluate the probability for any given complete path. Our design goal was to maintain a code structure that resembles the mathematical structure of the path space integral formulation as closely as possible: to compute the probability of a path, it is desirable to recognize for example equations (156) and (157) in the code.

3.4.1 Path extensions

Extension techniques as described in section 3.3.1 are built on top of a base path that was sampled by uni-directional forward path tracing by local sampling. A somewhat minimal technique interface could look something like this:

```
class ExtensionTechniqueInterface
{
  virtual ReturnCode SamplePath( const Path&      basePath,
                                 VertexGraph*     graph,
                                 GraphCache*      graphCache,
                                 Path*            newPath,
                                 PathRecord*      record ) const = 0;

  virtual double ComputeProbability( GraphCache&        graphCache,
                                     const VertexGraph& graph,
                                     const Path&        path ) const = 0;

};
```

where the input arguments to the SamplePath method carry along any kind of information that could be useful for the path extension, for instance the probability of sampling the current base path. The output record contains a description of the path, and quantities like the measurement and sampling probability. Computing probabilities is usually a relatively cheap side-computation of the sampling method, so it is useful to store it so that it can be used directly in the MIS weight computation. The method to compute the probability needs to have access to all information that is needed to replay all conditions that could have influenced the sampling.

A path vertex provides directionally independent information such as local frame and differential geometry. Crucially, the material BSDF is stored on the vertex as well, since it will be needed by evaluations of other connections and PDF calculations. Hence, it is a relatively heavy-weight structure that we would rather not replicate for each path contribution. Instead, we store every vertex exactly once in a graph (called a DAG[11]) whose edges describe the connectivity of path construction and thereafter describe paths in a light-weight way by referencing the vertices by node ids of the graph vertices.

The main path tracing loop thus simply starts with an empty graph, samples a root node on the camera or light source, then iteratively extends the path by local sampling and calls into the registered extension samplers. Figure 7 illustrates the process of incrementally building up new paths. The conditional, local probabilities for constructing a path are repeatedly required for computations of whole path probabilities. Therefore, we augment the graph with a caching structure that allows reuse of any previously computed quantity and evaluates it lazily only if necessary, which makes sure that the implementation of the probability methods do not need to know about what terms have already been computed. Naturally, all computations that fall out of the sampling methods (such as sampling probability of a BSDF as a path is extended) should be injected into the graph cache when they are generated.

Some typical records in the cache are the following:

- Keyed on the vertex id, incoming and outoing directions, a BSDF record stores quantities like the evaluation and sampling probability of the material. Directions are internally mapped from a pair of vertices onto a directional key to account for the fact that multiple, different vertices may lie on a ray segment. For example, in figure 7, x_6 is sampled by free-path sampling from x_1, while x_4, x_5 by construction of the equi-angular sampling technique share the same direction. The BSDF record may also contain adjoint quantities, depending on the query direction.
- Segments as simply identified by a pair of nodes, can contain information about the volume(s) between the endpoints, volume stack changes, volume sampling probabilities, emission and transmittance.
- Information on the beginning and end of the path, i.e., sensor and emitter data.

3.4.2 Shift techniques

As outlined in section 3.3.3, determining the type of shift mapping to be applied may depend on some prior inspection of the sampled vertices down from the root node. Fortunately, the vertex graph contains sufficient information to make that decision and can be extended by creating secondary root nodes and then merging them into the graph with the existing nodes. The decision of whether to reconnect may depend also on the depth and complexity of the graph: for short path-lengths the amount of work required to create connections relative to tracing completely new paths might be too high to pay off, especially if primary vertices are specular. Therefore, we decided to apply shift techniques independently from the previous path tracing stage as an independent post-process.

[11]This means *directed acyclic graph*, and it is easy to see how this graph is directed, because the edges have orientation, and acyclic, because cycles in this construction cannot arise

3.4.3 Volumes and Transparency

Participating media are first-class citizens in our framework: ray-casting is rarely used explicitly, instead free-path sampling (cf. section 3.2.1) that handles transparency and volumes in a unified manner is applied. Transitions between volumes may not only occur at explicitly sampled vertices, but also at transparent interfaces. Similarly, a transmittance method replaces the classic occlusion query in a very natural way.

This means that at every vertex and potentially at every point along an existing segment, the information about the current volume needs to be available. We track the active list of nested volumes in a *volume stack* structure similar to what is described in [Schmidt and Budge, 2002], which is augmented with various additional information about the volume such as entry-point information. We provide change-list mechanisms to be able to roll-back changes and to reconstruct volume transitions along rays in a light-weight way by storing a volume history for each segment. For example, an extension technique will retrieve the current volume stack at the current end-vertex and needs to roll-back any local changes before continuing in the main path tracing loop. The volume stack may also be important when computing the probability of a technique, which requires reconstructing all conditions that would have influenced the sampling method, for example the end-point of the ray-segment used for equi-angular sampling.

The volume stack structure also carries a mapping that describes which emitters are contained within a given volume and can thus be used for emitter selection, to avoid for example next-event estimation to lights that can never be connected to. We hide the specific structure of a type of volume from path-space sampling as much as possible. Even though some methods may be more efficient than others on certain types of volumes, a path sampling technique should work regardless of whether the volume is a homogeneous fog-volume or an atmosphere model.

3.4.4 Photon mapping / VCM

So far we have mainly looked at typical *unbiased* sampling methods, but since generality is an important design goal we ought to investigate how the decoupling approach could hold up in other cases such as *photon-mapping*. Note that (apart from path guiding) none of the sampling methods described before is yet able to handle the SDS paths in figure 2(a) effectively. The MIS weights computation for VCM [Georgiev et al., 2012, Hachisuka et al., 2012] serves as a good stress-test. Similar to the carefully optimized MIS evaluation schemes for BDPT in section 3.3.1, Georgiev [2012] developed an efficient evaluation scheme for VCM. However, it is restricted to a specific set of techniques and is rather complex to maintain and generalize.

As an exercise, let us see what terms are required to get an idea of how compactly the data on a photon could be represented, without of course having to store full light paths. Given a merge radius r (in the following, for the sake of simplicity, globally constant) the vertex merging sampling probability of a path with a merge occurring at light vertex j is:

$$p_{\mathrm{VM},j}(\bar{\mathbf{x}}) := \pi r^2 \prod_{i=0}^{j} \vec{p}_i(\bar{\mathbf{x}}) \prod_{i=j}^{k} \overleftarrow{p}_i(\bar{\mathbf{x}}). \tag{159}$$

Denoting the set of active techniqes by T, the MIS weight for technique t^* (w.l.o.g. balance heuristic) is

$$w_{t^*}(\bar{\mathbf{x}}) = \frac{n_{t^*} p_{t^*}(\bar{\mathbf{x}})}{\sum_{t \in T} n_t p_t(\bar{\mathbf{x}}) + n_{\mathrm{VM}} \sum_{j=1}^{k-1} p_{\mathrm{VM},j}(\bar{\mathbf{x}})}, \tag{160}$$

which needs to be computed for any vertex merged contribution as well as for paths sampled by other techniques. Here, n_t denotes the number of samples drawn from technique t.

The main task is to determine a compact set of quantities to store on the photon such that the weight can be computed once a connection has been made. Therefore, the sum of vertex merging probabilities at photon index

s can be rewritten as

$$\begin{aligned}\sum_{j=1}^{k-1} p_{\mathrm{VM},j}(\bar{\mathbf{x}}) &= \sum_{j=1}^{s-1} p_{\mathrm{VM},j}(\bar{\mathbf{x}}) + \sum_{j=s}^{k-1} p_{\mathrm{VM},j}(\bar{\mathbf{x}}) \\ &= \pi r^2 \overleftarrow{p}_{s-1}(\bar{\mathbf{x}}) \underbrace{\left(\sum_{j=1}^{s-1}\prod_{i=0}^{j}\vec{p}_i(\bar{\mathbf{x}})\prod_{i=j}^{s-2}\overleftarrow{p}_i(\bar{\mathbf{x}})\right)}_{=:\vec{v}_{s-1}} \prod_{i=s}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}) \\ &\quad + \pi r^2 \vec{p}_{s+1}(\bar{\mathbf{x}}) \underbrace{\prod_{i=0}^{s}\vec{p}_i(\bar{\mathbf{x}})}_{\vec{w}_s} \left(\sum_{j=s}^{k-1}\prod_{i=s+2}^{j}\vec{p}_i(\bar{\mathbf{x}})\prod_{i=j}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}})\right).\end{aligned}$$

Computation breakdown The structure of terms above allows us to seperate the quantities that may be stored with the photon, the quantities that depend on the path up to the merge location and some terms that need to be computed as the connection is being made:

- $\vec{v}_{s-1}$ depends only on photon path vertices → stored in photon map.
- $\prod_{i=s}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}})$ is the iteratively built path probability of the eye path up to the merge vertex and is usually a simple side-computation of path tracing.
- The local sampling probabilities $\overleftarrow{p}_{s-1}(\bar{\mathbf{x}})$ and $\vec{p}_{s+1}(\bar{\mathbf{x}})$ depend on the photon direction and incoming path direction. They can therefore only be computed when connecting the photon to the eye path. In principle, this would also require the BSDF at the vertices involved in the merge. Reconstructing these for each vertex is too costly, so most implementations simply reuse the material at the connecting vertex and pretend it doesn't change too much within the merge region.
- $\vec{w}_s$ depends only on photon path vertices and is a side computation of light tracing → stored in photon map.
- $\sum_{j=s}^{k-1}\prod_{i=s+1}^{j}\vec{p}_i(\bar{\mathbf{x}})\prod_{i=j}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}})$ depends only on eye path vertices and can be computed as a side product of forward path tracing.

Hence, similar to [Georgiev, 2012], a compact photon payload can be achieved by storing the direction $x_{s-1} - x_s$, $\vec{v}_{s-1}$, $\vec{w}_s$ with each photon. This formulation is not optimal in reducing the number of floating point operations but remains relatively simple, and terms like $\vec{v}_{s-1}$ are straightforward to compute efficiently.

Weighting a vertex merge against other techniques For simplicity, let us consider how the most standard techniques can be included in section 3.4.4:

- **Forward path tracing** sampling probability

$$\overleftarrow{p}(\bar{\mathbf{x}}) = \prod_{i=0}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}) = \overleftarrow{p}_{s-1}\prod_{i=0}^{s-2}\overleftarrow{p}_i(\bar{\mathbf{x}})\prod_{i=s}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}).$$

  The first term is computed at connection, the second can be stored in the photon, the last is computed from the connecting path.
- **Forward path tracing with next-event estimation** has the probability

$$\overleftarrow{p}_{\mathrm{NEE}}(\bar{\mathbf{x}}) := \overleftarrow{p}_{\mathrm{NEE}}(x_0)\prod_{i=1}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}) = \overleftarrow{p}_{s-1}\overleftarrow{p}_{\mathrm{NEE}}(x_0)\prod_{i=1}^{s-2}\overleftarrow{p}_i(\bar{\mathbf{x}})\prod_{i=s}^{k}\overleftarrow{p}_i(\bar{\mathbf{x}}).$$

Light tracing can be handled completely analogously. Now, the structure is relatively obvious, the probability can usually be broken down into a part that only depends on the light path the photon traveled, a part that only depends on the eye path up to the merge and a part that needs to be computed dynamically as the connection is made.

Therefore the technique API only needs to be augmented by functions to prepare the photon payload and to compute the probability of a connection path based on the payload, the knowledge of the forward path and some local computation. Some terms are of course common, for example note that $\vec{v}_{s-1}$ and the terms from the two forward path tracing techniques can be aggregated into a single value.

This rather explicit approach also allowed us to extend VCM to multispectral sampling in a straightforward way: recall from section 3.3.5 that this mainly requires the computation of the sampling probabilities for all wavelength channels independently. In practice however, we rely increasingly on path guiding for difficult light transport paths such as caustics [Vorba et al., 2019].

3.4.5 Memory management

The amount of dynamism that is required in such a framework is rather high, making an efficient implementation that is not negatively impeding the performance of light-transport challenging. The vertex graph, the attached caching structure and the volume stack are highly dynamic structures. Furthermore, since we don't want to impose a limit on path length and the size of the graph and the amount of cached data is not bound a-priori. BSDFs may be temporarily allocated during free-path sampling and then attached to vertices when an endpoint is reached.

We heavily rely on memory pools to make sure that (apart from an initial warm-up phase) no heap allocations are required down to the finest level such as allocation of connectivity edges in the vertex graph or control blocks for shared pointers. Path tracing construction for a single vertex graph, including extensions and shifts is scheduled sufficiently fine-grained on a single thread, which allows us to place thread pools in a local thread context without incurring any contention.

3.4.6 Discussion

Pros We found that the proposed approach has proven to be very flexible and has allowed us to experiment and prototype without introducing any coupling with the rest of the system, thus improving robustness. Light transport algorithms can be algorithmically very complex to implement because a lot of details and special cases need to be accounted for. Even though a small amount of computation could in theory be saved by transforming the MIS weight terms into a different form, this is from our point of view and experience not worth the price of having to give up modularity, especially since in practice the vast majority of the computations go into evaluating probabilities for light hierarchy sampling and the BSDF, which the cache structure guarantees not to evaluate redundantly.

Cons Storing vertices, materials, potentially caching volume voxel data along segments increases the memory footprint, which obviously would become a limitation for scaling up the number of threads under memory constraints, so it is in its full generality not straightforward how to map such an approach onto GPU architectures and integrate into a wave-front architecture in particular [Laine et al., 2013]. GPU renderers therefore often deliberately restrict themselves to a smaller set of techniques that can be specifically optimized for and thus still be quite powerful (see for example the design decisions made by Keller et al. [2017]).

Long paths can require extra care for two reasons: First, even though the amount of compute needed to calculate the path probabilities by putting together known values, such as geometry terms and local BSDF probabilities, is negligible compared to other computations, this is no longer true for long paths. This can however be accounted for by caching aggregate path contruction probabilities for interior chains, since extensions and shifts happen usually at the beginning and end of the graph. Secondly, some extra care is necessary to avoid numerical precision problems, for example when accumulating local probabilities for long subsurface scattering chains on which the geometry term is small.

Quest for performance and flexibility Of course, optimizing performance and scalability is extremely important to reduce artist iteration times and rendering costs. Also, having the flexibility to adapt to unforeseen technical challenges is a huge benefit for an in-house production renderer. It is therefore worth paying close attention to avoid imposing too severe algorithmic constraints through a tight coupling of different sampling techniques or by imposing, for example, a complex scheduling system that intrudes deeply into path sampling. In an early prototype stage, *Manuka* in fact employed a wavefront scheduling scheme, which was later on abandoned as the need for more algorithmic flexibility become more and more apparent.

Programmable integrator vs programmable technique architecture A programmable integrator architecture that provides access to rather low-level functionality, such as ray casting and shading, ensure a high degree of flexibility for implementing rendering algorithms for different purposes, such as a final-frame renderer versus a very specific renderer for utility passes.

The programmable technique approach is in comparison more high-level, and allows sharing of a large part of the light transport infrastructure, such as the volume handling, free-path sampling, imaging etc. and facilitates the usage of a unified feature rich integrator. It therefore avoids sitations in which a feature X and Y can't be used in combination because they might be implemented in different integrators, even though they might be completely unrelated.

Acknowledgements

We would like to thank all our visiting researchers that have contributed to light transport sampling in *Manuka* over the years, in particular Tzu-Mao Li, Marco Manzi and Markus Kettunen who challenged and helped shaping the design of the architecture by implementing interesting path reconnection techniques and shift-mappings on top of it, as well as Iliyan Georgiev who managed to bring VCM into *Manuka* way before the framework described here even existed, and Wenzel Jakob, who's work on implementing Manifold Exploration into *Manuka* provided the inspiration for the first prototype of the framework.

References

P. Bekaert, M. Sbert, and J. Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Proc. Thirteenth Eurographics Workshop on Rendering*, P. Debevec and S. Gibson (Eds.). 125–134.

Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. 2017. Reversible Jump Metropolis Light Transport Using Inverse Mappings. *ACM Transactions on Graphics* 37, 1 (Oct. 2017). https://doi.org/10.1145/3132704

Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Trans. Graph.* 37, 3, Article 30 (Aug. 2018), 21 pages. https://doi.org/10.1145/3182162

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy Redistribution Path Tracing. *ACM Trans. Graph.* 24, 3 (July 2005), 1186–1195. https://doi.org/10.1145/1073204.1073330

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3, Article 31 (Aug. 2018), 18 pages. https://doi.org/10.1145/3182161

Luca Fascione, Johannes Hanika, Rob Pieké, Christophe Hery, Ryusuke Villemin, Thorsten-Walther Schmidt, Christopher Kulla, Daniel Heckenberg, and André Mazzone. 2017. Path Tracing in Production - Part 2: Making Movies. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. Article 15, 32 pages. https://doi.org/10.1145/3084873.3084906

Iliyan Georgiev. 2012. *Implementing Vertex Connection and Merging.* Technical Report. Saarland University. http://www.iliyan.com/publications/ImplementingVCM

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* 37, 3, Article 32 (Aug. 2018), 12 pages. https://doi.org/10.1145/3182160

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. 2013. Joint Importance Sampling of Low-Order Volumetric Scattering. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 1–14.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph.* 31, 6 (2012), 191:1–191:10. https://doi.org/10.1145/2366145.2366210

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34, 4 (June 2015), 87–97.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum* 21, 3 (2002), 531–540. https://doi.org/10.1111/1467-8659.t01-1-00703 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.t01-1-00703

Alexander Keller, Carsten Wächter, Matthias Raab, Daniel Seibert, Dietger van Antwerpen, Johann Korndörfer, and Lutz Kettner. 2017. The Iray Light Transport Simulation and Rendering System. *CoRR* abs/1705.01263 (2017). arXiv:1705.01263 http://arxiv.org/abs/1705.01263

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Trans. Graph.* 34, 4 (2015).

David Koerner, Jan Novák, Peter Kutz, Ralf Habel, and Wojciech Jarosz. 2016. Subdivision Next-Event Estimation for Path-Traced Subsurface Scattering. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association.

T. Kollig and A. Keller. 2000. Efficient bidirectional path-tracing by randomized Quasi-Monte Carlo integration. In *MCQMC Methods*.

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. In *Eurographics Symposium on Rendering*, Fredo Durand and Diego Gutierrez (Eds.), Vol. 31.

Samuli Laine, Tero Karras, and Timo Aila. 2013. Megakernels Considered Harmful: Wavefront Path Tracing on GPUs. In *Proceedings of the 5th High-Performance Graphics Conference (HPG '13)*. ACM, New York, NY, USA, 137–143.

Johannes Meng, Johannes Hanika, and Carsten Dachsbacher. 2016. Improving the Dwivedi Sampling Scheme. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 35, 4 (June 2016), 37–44.

Hisanari Otsu, Anton Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. 2017. Fusing State Spaces for Markov Chain Monte Carlo Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 35, 4 (Aug. 2017), 1–10.

Jacopo Pantaleoni. 2016. Charted Metropolis Light Transport. *CoRR* abs/1612.05395 (2016). arXiv:1612.05395 http://arxiv.org/abs/1612.05395

M. Pauly, T. Kollig, and A. Keller. 2000. Metropolis Light Transport for Participating Media. In *Rendering Techniques 2000. Eurographics*, B. Péroche and H. Rushmeier (Eds.). Springer, Vienna.

Matt Pharr. 2019. *Ray Tracing Gems*. Apress, Chapter On the importance of sampling.

Matt Pharr and Kavita Bala (Eds.). 2018. Special Issue On Production Rendering. *ACM Trans. Graph.* 37, 3 (2018).

Matthias Raab, Daniel Seibert, and Alexander Keller. 2006. Unbiased Global Illumination with Participating Media. In *Monte Carlo and Quasi-Monte Carlo Methods*, A. Keller, S. Heinrich, and H. Niederreider (Eds.). Springer, Berlin, Heidelberg.

Charles M. Schmidt and Brian Budge. 2002. Simple Nested Dielectrics in Ray Traced Images. *Journal of Graphics Tools* 7, 2 (2002), 1–8. https://doi.org/10.1080/10867651.2002.10487555

Sebastién Speierer, Christophe Hery, Ryusuke Villemin, and Wenzel Jakob. 2018. *Caustic Connection Strategies for Bidirectional Path Tracing*. Technical Report. Pixar Technical Memo #18-01.

Lorenzo Tessari, Johannes Hanika, and Carsten Dachsbacher. 2017. Local Quasi-Monte Carlo Exploration. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations*. https://doi.org/10.2312/sre.20171196

Dietger van Antwerpen. 2011. Improving SIMD Efficiency for Parallel Monte Carlo Light Transport on the GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics 2011, Vancouver, Canada, August 5-7, 2011*. 41–50. https://doi.org/10.1145/2018323.2018330

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Ph.D. Dissertation. Stanford University. AAI9837162.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses (SIGGRAPH '19)*.

Jiří Vorba, Ondrej Karlík, Martin Sik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph.* 33, 4 (2014), 101:1–101:11. https://doi.org/10.1145/2601097.2601203

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Trans. Graph.* 35, 4, Article 42 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925912

Pascal Weber, Johannes Hanika, and Carsten Dachsbacher. 2017. Multiple Vertex Next Event Estimation for Lighting in dense, forward-scattering Media. *Computer Graphics Forum (Proceedings of Eurographics)* (April 2017).

Alexander Wilkie, Sehera Nawaz, Marc Droske, Andrea Weidlich, and Johannes Hanika. 2014. Hero Wavelength Spectral Sampling. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 33, 4 (July 2014), 123–131.

4 Finding good paths

Jorge Schwarzhaupt, *Weta Digital*

Rendering film-quality images has always been a complicated endeavor, because producing images that realistically merge into plate photography requires simulating the complex interactions between lights and materials. In that sense, the rise of path tracing renderers has been a boon to the visual effects industry, because the generality afforded by Monte Carlo integration allows for solving even the most challenging light transport situations. Unfortunately, this doesn't mean that they can necessarily be solved *efficiently*.

Making path tracing practical for use in visual effects means dealing with noise. The complexity of the scenes that need to be rendered is such that many (*many!*) paths need to be traced for every pixel in the final image so that the end result will have an acceptable level of noise. This complexity comes from many places: complex light rigs illuminating complex, animated characters featuring complex materials inserted within complex environments. The resulting noise is similarly complex, because different parts of the scene *vary* in their complexity, meaning the noise is not uniform across the final image; the illumination hitting one character may be entirely direct, for instance, while a neighboring one only receives indirect illumination, or some surfaces in a scene feature reasonably simple materials while others have interesting sub-surface scattering or other effects. Some parts of the scene may be moving while others remain static, or depth-of-field can be simulated and parts of the scene are in focus while others are not. With this in mind, there are several complementary ways of dealing with the problem of noise within a path tracing renderer.

The simplest one is to give the renderer enough time tracing paths so that the final image is of sufficient quality throughout. This works! Unfortunately, time is generally a very precious resource, and this approach is inherently inefficient, as making sure the most noisy pixels in an image have sufficiently low noise levels necessarily implies that other parts of the image have had more paths traced for them than was necessary. This observation leads quite naturally to the development of another way of dealing with noise — adaptive sampling of the image plane.

Adaptive sampling means the renderer has an understanding of which parts of an image are noisier than others, and can use that knowledge to drive the path tracing engine towards tracing more paths through noisier pixels and fewer through those that are less noisy. It has been our observation at Weta Digital that a good, robust adaptive sampling engine is not just "nice to have"; it is absolutely essential for delivering an efficient workflow to the artists. This is because the levels of noise across an image can vary substantially, sometimes by orders of magnitude, and the days where scenes were sufficiently simple for artists to tweak a couple of parameters affecting local sampling decisions are long gone. Weta Digital's renderer, *Manuka*, allows artists to specify a desired final noise level and let the renderer figure out which parts are noisier. However, notice how this doesn't say anything about *how long* the renderer will take to reach that noise level. Some scenes feature light transport situations so complicated that tens (or hundreds!) of thousands of paths may be required in small or large regions of the image. So what can we do about this?

Perhaps the best (but hardest) way of dealing with noise in a path tracer is to find ways of simply producing less of it, that is come up with techniques and improvements that can make our humble path tracer better at *finding good paths*. Within *Manuka*, we approach this goal from two distinct angles: increasing path-sampling efficiency by re-utilizing the information gathered from a single path across several pixels, and implementing better sampling techniques that reduce the variance of our Monte Carlo integrators.

4.1 Path reconnection techniques for stereoscopic rendering and distribution effects

What if you were to take a look under the hood at what is happening as a path tracer is going about the business of rendering an image? You would probably see many paths being traced through the scene being rendered, with rays either landing on surfaces or crossing through volumes, generating path vertices. From there, new directions are sampled to continue the path and so on, until paths are terminated after fulfilling some condition, such as landing on a light source or escaping the scene into the environment around it. If you were to take a quick snapshot of this process, it is quite likely that these paths will exhibit very little coherence, as each path is sampled in an independent manner. But if you were to observe this process for an extended amount of time it will become glaringly obvious

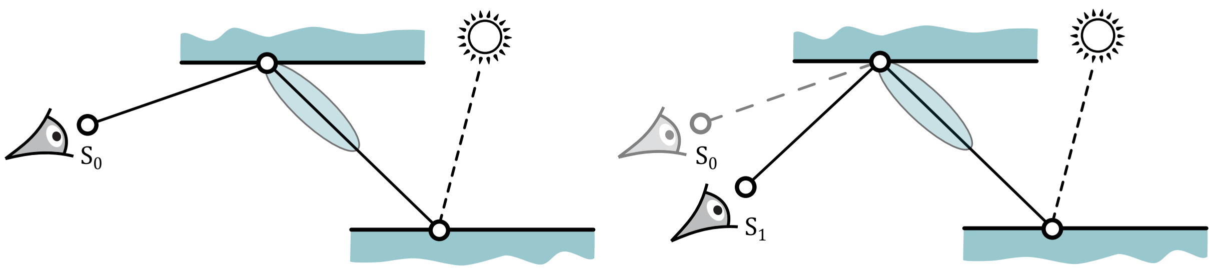


Figure 8: *Rendering from multiple cameras.* ***Left:*** *Path created from sensor* 0 *using classic sampling.* ***Right:*** *The secondary technique starts a path from sensor* 1 *and applies the sampling lobe at the first surface point as if coming from sensor* 0*. Using correlated sampling, the remaining vertices coincide.*

that at least some, and possibly a lot, of the work being performed has been performed previously and is being re-done. For example, the camera from which the paths originate may be moving, such that a single surface element is visible through several pixels, or there may be multiple cameras, as happens in stereoscopic rendering. In these cases, we'll have independent estimators that are evaluating extremely similar — possibly identical — integrals, leading to lots of redundant work being performed. If we are trying to be as efficient as possible, then, it seems that trying to exploit these redundancies is crucial. To paraphrase what artists observed to us repeatedly — "if my stereo renders look almost identical, why should I pay twice the price of rendering them as mono?".

The answer is of course that they shouldn't. But how do we go about exploiting these redundancies in a way that results in acceptably correct images? To that end, the path sampling architecture that is developed in the prior section becomes quite useful, and we exploit it quite severely in implementing what we internally refer to as *Optimized Stereo, Motion Blur and Depth-of-Field.*

4.1.1 Multi-view rendering

Extending a path tracer to support rendering to multiple sensors in a single pass is easily achieved by incorporating the sensor domain into the standard measurement equation:

$$I_{p_i^s} = \int_{\mathcal{R}} W_{e,p_i^s}(\bar{\mathbf{x}}) L_i(\bar{\mathbf{x}}) \,\mathrm{d}\mu(\bar{\mathbf{x}}), \tag{161}$$

where L_i is the incoming radiance, W_{e,p_i^s} corresponds to the importance of pixel i in the sensor s evaluated for a path $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in the path space $\mathcal{R}$. In practice, this can be sampled for all pixels and sensors simultaneously, sampling initial path vertices over the whole domain and assigning the result to the respective pixel via rejection sampling Veach [1997].

This formulation allows a very effective amortization of the various scene preprocessing costs (such as tesselation and acceleration structure creation), while light transport simulation costs remain essentially unchanged. The technique as described is robust and easy to implement correctly, yet remains unsatisfactory with respect to efficiency as previously discussed.

4.1.2 Primary path reconnection

Now, while light tracing makes it trivial to reuse paths emanating from the light source by simply connecting to all sensors, some extra work is required for the case of path tracing. A naïve approach to exploit the low disparity across different sensors is to simply reconnect the primary vertex of a sampled path to all other sensors, which would allow us to amortize all of the potentially expensive computations performed for the secondary vertices of the path.

In the Monte Carlo integration setting, we can achieve this by constructing a new sampling technique that reconnects a path generated by a known sampling technique to the other sensor(s). Specifically for stereoscopic rendering, we define two sampling techniques, with T_0 being the known base technique as described above for sampling from the primary sensor into the scene, and T_1, which after sampling the secondary sensor effectively delegates back to T_0 for the remainder of the path. By applying the same random number sequence to both techniques, under the mild assumption that the samplers reproduce the same values for the same random number,

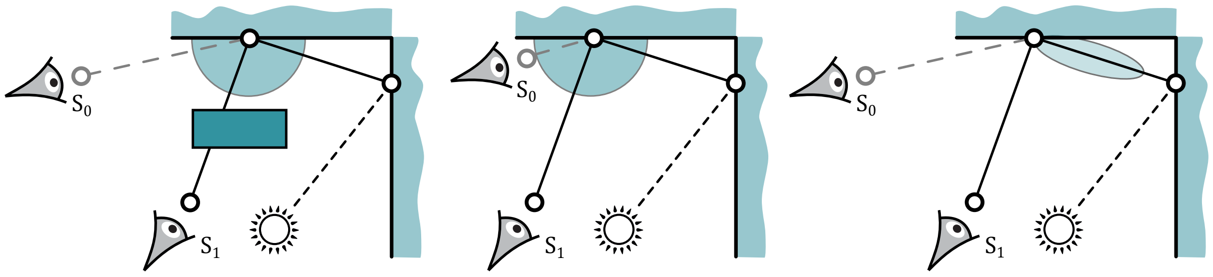


Figure 9: *Cases in which path reconnection is ineffective: occlusion, sample density mismatch, strongly glossy* BSDF.

identical sub-paths $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ are generated. In this initial form, the approach still has a number of limitations as illustrated in figure 9: except for trivial scenes, there are points that are primary visible only in one of the two sensors, either due to occlusion or because they fall outside the image domain. In the end, after adapting the weights of the techniques to avoid introducing bias, it is necessary for the renderer to trace as many paths as it originally would have to resolve the noise in these disparity regions. Within *Manuka*, we deal with this particular limitation by using adaptive sampling, which effectively understands that these disparity regions receive fewer samples overall and compensates accordingly.

A similar issue arises at surface points that have a strongly varying primary sampling density, so that the samples inherited from one of the sensors are sparser than the other, degrading the efficacy of the reconnections. Reevaluation of reconnected paths yields zero throughput in the case of specular materials. For strongly glossy materials, the secondary technique is inefficient, as the lobe is importance sampled as seen from the primary sensor, and hence the variance can become unbounded.

The natural remedy is to apply *multiple importance sampling* (MIS) Veach [1997] to avoid increasing variance at the cost of evaluating the probability of sampling the paths with the other technique, which involves an occlusion test and evaluating the PDF of the surface BSDF at the primary path vertex $\mathbf{x}_1$. Let's see how this operates in practice. First, we use our previously described known technique T_0 to create a path through the scene. This path starts at a sensor s which has been sampled with uniform probability. So far, this is exactly the same as you would do when sampling a path in a stereo path tracer. Once this *base* path $\bar{\mathbf{x}}(s)$ has been sampled, we fire up our secondary technique, T_1, which takes $\bar{\mathbf{x}}(s)$ and creates a *secondary* path $\bar{\mathbf{x}}(s')$, with s' the other sensor, by replacing vertex $\mathbf{x}_{s,0}$ with a new vertex $\mathbf{x}_{s',0}$ on s'. This is done by projecting the position of $\mathbf{x}_1$ to s'. Disregarding for now the jacobian of the projection, the MIS weights for paths $\bar{\mathbf{x}}(\cdot)$ sampled by each of the two techniques are expressed as follows in the case of the balance heuristic:

$$w_0(\bar{\mathbf{x}}(s)) = \frac{p_0(\bar{\mathbf{x}}(s))}{p_0(\bar{\mathbf{x}}(s)) + p_1(\bar{\mathbf{x}}(s))} = \frac{p_1(\bar{\mathbf{x}}(s'))}{p_1(\bar{\mathbf{x}}(s')) + p_0(\bar{\mathbf{x}}(s'))} = w_1(\bar{\mathbf{x}}(s')),$$

(other MIS heuristics are modified similarly) so that, effectively, only two PDFs have to be evaluated to compute the weights for both paths; in simple terms, the MIS weight coincides for both techniques. If we also incorporate next-event estimation at each vertex, we increase the number of PDFs that need to be evaluated by two, as described by Wilkie et al. [2014]. Multiple importance sampling provides a means to avoid increasing variance due to potentially *inefficient* shifted techniques, that is, in situations where reconnection becomes deficient, the weighted combination degrades to the efficiency of the base technique. In these situations we prefer higher variance, which can be addressed by locally increasing sample density, for example by (once again) using an image-space adaptive sampling engine, rather than suffer potential artifacts resulting from forcing path reconnection.

4.1.3 Higher dimensions

The technique as described so far can be interpreted as a simple and robust way to combine regularly-sampled paths with explicitly constructed perturbations using shifted techniques, similar to Bekaert et al. [2002]. This is appealing for other dimensions of the integrand as well. For example, the domains of depth of field and motion blur effects are often small enough to make reuse of long paths attractive in our quest for improved renderer efficiency. Conventionally, the number of PDF evaluations needed to compute weights for MIS grows as the square

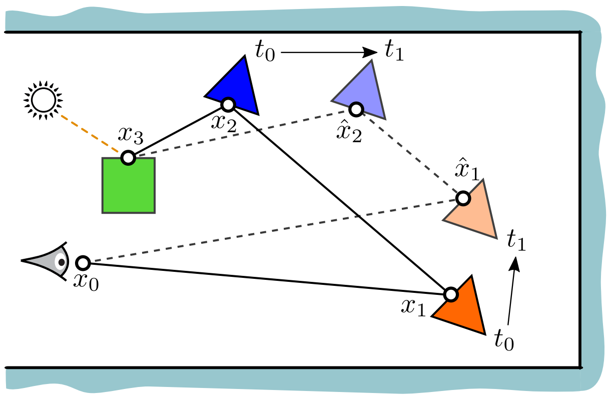


Figure 10: *Path warping: an initial path is sampled and evaluated at* $t = t_0$*. The shifted technique yields the path* $\bar{\mathbf{x}}(t_1) = (\mathbf{x}_0, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots)$ *at* $t = t_1$.

of the number of techniques and it becomes prohibitively expensive to simply raise the dimensionality of the perturbations. In the stereo case, we saw from the previous section that, by construction and due to the sampling correlation, some of the PDFs end up matching and the MIS weights coincide. Employing appropriate perturbations, the same holds for an arbitrary number of techniques, yielding a linear instead of quadratic cost for MIS weight computation with respect to the number of perturbations. So let's try applying this to some distribution effects.

4.1.4 Motion Blur

Incorporating temporal dependencies into a path tracer is formulated as an additional integration over the time domain; that is, each sampled path is traced against the geometry and materials for a specific sampled moment in time Cook [1986]. Though the result looks smoother under motion blur, a significantly greater number of samples are needed to explore the enlarged integration domain in order to obtain a comparable variance to the static case. Even when employing efficient acceleration structures for time-dependent ray-tracing, such as time-interpolated BVHs Christensen et al. [2006], Grünschloss et al. [2011] or hyper-trapezoids Hou et al. [2010], there is still a significant overhead attributable to node interpolation during traversal, as well as the fact that the tree topology is commonly only optimized for a fixed time.

What we want is to apply the correlated sampling framework described above to exploit the special structure of the spatio-temporal lightfield: the variation of non-local lighting changes are typically small along motion trajectories, whereas occlusion causes discontinuities along the time axis Hachisuka et al. [2008], Lehtinen et al. [2011]. In particular, analogous to reusing indirect lighting calculations by reconnection in the stereo case, we reuse indirect sub-paths for a sampled time t_0 to N shifted times t_i within a time domain $\hat{T}$:

- Shift time t to t_i such that $t_i = (t + \frac{i}{N}\hat{T}) \mod \hat{T}$.
- Copy path $\bar{\mathbf{x}}(t_0)$ to $\bar{\mathbf{x}}(t_i)$ and warp the path vertices to time t_i.
- Compute the measurement of path $\bar{\mathbf{x}}(t_i)$ and weight the result using MIS.

Moving path vertices to arbitrary times (cf. figure 10) is a natural operation for most path tracing architectures, where one requirement is the storage of information sufficient to determine the position of a point on a surface at any moment in time. This will generally be a polygon id, *uv* coordinates and sampled time. A further requirement is a *warping method* that, given this information and arbitrary time, results in the correct position and shading frame of the point at the given time.

4.1.5 Computing path throughput

In fully dynamic scenes, moving all vertices to the shifted time and evaluating the throughput would defeat the purpose of the algorithm, because there would be little chance to reuse any secondary subpaths. Because a full re-evaluation involves occlusion queries and material evaluations, the computational cost per sample would make the technique impractical. However, the strategy we have outlined previously allows us to easily implement schemes of different degrees of accuracy with consequently different computational overhead. The simplest scheme is to avoid all material re-evaluations and occlusion tests and use the exitant radiance at vertex $\mathbf{x}_1(t)$ for time t and weight the results equally for all i. However, while the effect of self-occlusion is usually quite subtle, accounting for inter-object visibility is important to avoid leaking light through occluders that are closer to camera. Since occlusion and motion across the image region may prevent some of the techniques from having a positive probability of sampling the warped vertices, it is crucial to incorporate visibility into the MIS weight.

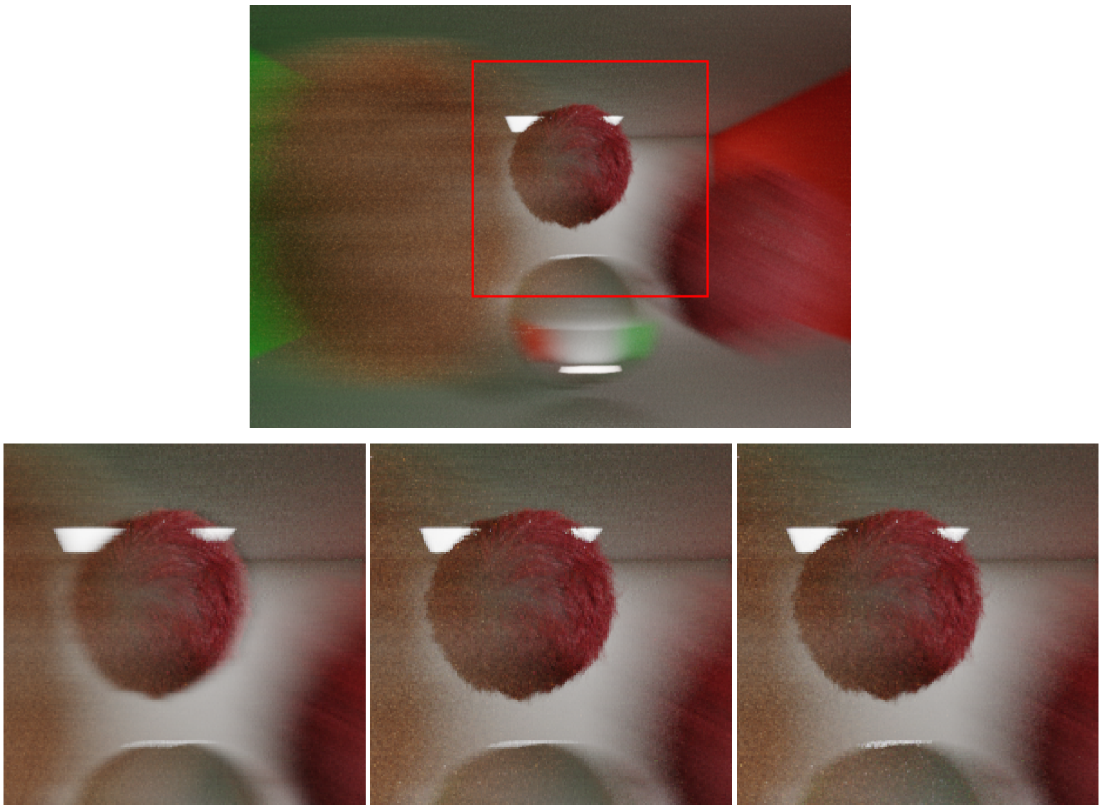


Figure 11: *A scene with furry balls. Top row compares path throughput computation methods for motion blur with varying degrees of accuracy:* ***Left:*** *no material or occlusion;* ***Middle:*** *occlusion only;* ***Right:*** *full evaluation.*

To evaluate the material on the transformed primary vertex at a requested time, the incoming and outgoing directions ω_i and ω_o can simply be computed after transforming the camera vertex $\mathbf{x}_0$ and the secondary vertex $\mathbf{x}_2$. Reevaluating the material for even slightly perturbed directions can have a strong effect on the path throughput at specular and highly glossy vertices, and we have found that the added overhead is low enough that we never recommend our users disable this step. This is because the artifacts that arise from avoiding this computation can be distracting and hard to diagnose, as they don't generally look flat-out wrong, just slightly "off". (cf. figure 11).

4.1.6 Adapting to projected velocity

We choose the number N of shifts applied in the time domain depending on the amount of motion in the scene, because for low velocities the shifted paths are too close to the original path to amortize the additional cost. We can easily achieve this using a pre-processing pass that shoots a fixed number of rays into the scene and estimates the number of pixels covered along the motion trajectory for each hit. Once this global estimate is known, the number of shifted rotations is chosen proportional to it and used for all paths. A better alternative exists though, which is to locally adapt this number; this is valid as long as the criterion is the same for all sampling techniques that are weighted together using multiple importance sampling. That means that we can use a local estimate of the pixel footprint of the motion trajectory, computed by using the warping functionality. This approach is obviously advantageous when the amount of motion in a scene is very non-uniform — on static objects for example, the

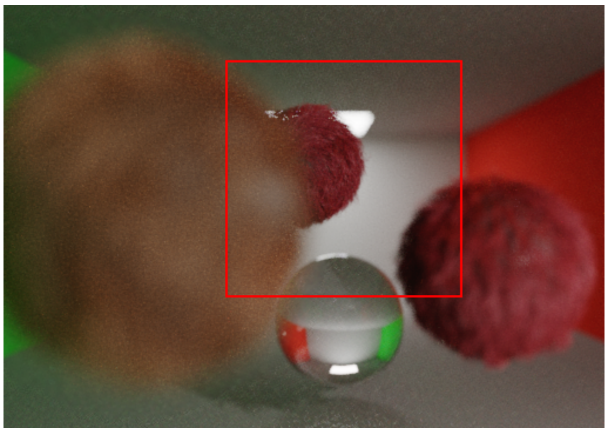
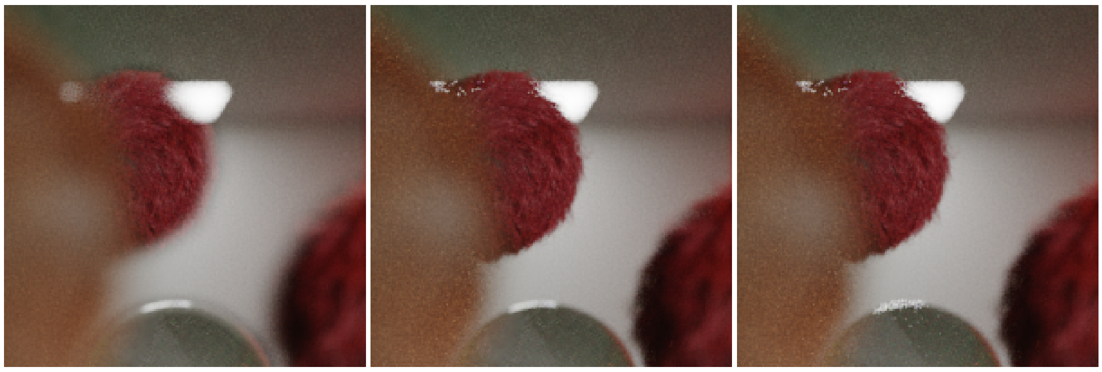



Figure 12: *A scene with furry balls. Top row compares path throughput computation methods for depth of field with varying degrees of accuracy:* ***Left:*** *no material or occlusion;* ***Middle:*** *occlusion only;* ***Right:*** *full evaluation.*

number of perturbations reduces to 1. This reduces the overhead of the method in those regions that benefit less from it.

4.1.7 Depth of Field

As for the case of motion blur, depth of field can be incorporated into a path tracer by adding integration over the lens area domain: each path is traced for a specific sampled position on the lens. While this adds no further complexities to the path tracing algorithms, the enlarged integration domain leads to higher variance for areas of the image that are out of focus. We can apply the correlated sampling framework to this effect as we did for time, under the expectation that the variation of non-local lighting changes will be small for perturbations across the lens domain, with occlusion being the dominant effect. Thus, we reuse indirect subpaths as in the motion blur case. As for stereoscopic rendering, no warping of the primary sampled vertex position is necessary.

The lens shifts can be thought of as connections to secondary sensors, though in this case there is an infinite number of them that must instead be sampled. Therefore, the computation of path throughput and MIS weights is performed similar to that case, except that the lens area PDF must also be accounted for. We have implemented the same throughput evaluation schemes for DOF as we presented for the motion blur case, and found that while occlusion is again an important effect to take into account, the effect of an accurate material re-evaluation is generally small and can be ignored in practice in most cases (see figure 12).

In the case of depth of field effects, the size of the circle of confusion will change with the distance of a sampled scene location to the camera. As in the case of motion blur we locally adapt the number N of shifts for each path: in-focus objects devolve to the standard case having a single path, while out-of-focus objects generate a number of shifts in proportion to how blurry they appear, keeping the overhead of the technique low for cases where it isn't effective.

4.1.8 Simultaneous effects

So far we have derived the application of our framework independently for stereo, motion blur and depth of field effects. In realistic production cases however, all these effects take place simultaneously in a scene, and it is therefore interesting to investigate the implications of applying our method to all techniques at once. The straightforward approach would be to apply the techniques independently, for example by first perturbing the sensor dimension to each of M sensors, then perturbing each of the resulting paths through time N times, and finally perturbing these paths across the lens a further O times. However, this leads to a combinatorial increase in the overhead imposed by the techniques, which may render it impractical for, e.g., surfaces that are simultaneously out of focus and undergoing fast motion. When considering the sensor dimension, this approach seems correct since each sensor will give rise to an independent image, while for motion blur and depth of field a less expensive approach seems desirable.

Our experience has shown that a better approach in this case is to combine the adaptively determined number of shifts to perform for motion blur and depth of field through addition and then warp the samples along both the time *and* lens domains simultaneously. This has the side effect of increasing the sample rate of both effects when compared to the naïve approach, while keeping the computational overhead under control.

4.1.9 Summary

We have developed a simple and efficient path sampling framework targeted at exploiting similarity during sampling of stereo and distribution effects. A key aspect of our approach is its simplicity and minimal invasiveness with respect to integration into a path tracing rendering architecture: it builds upon common BSDF sampling, evaluation and ray tracing components which are readily available in a physically based path tracer. Since shifts are constructed from a single base path, no synchronization with other paths is required, which avoids expensive storage of potentially heavy path state and thread synchronization. Apart from higher per-path splatting throughput, the algorithm does not intrude into the imaging pipeline: important production features such as progressive rendering, the handling of AOVs and alpha channels are unaffected. It is very easy to use, as no un-intuitive parameters have to be exposed to the end user.

For stereoscopic rendering, the proposed method is unbiased. Our experience using this technique at Weta Digital has been that it improves efficiency by an average of 75 percent when compared to rendering the left and right images independently, and a key side effect of our method is that indirect noise is purposefully replicated at corresponding locations in the images. This noise can be interpreted as texture, which is an important secondary depth cue for the human visual system [Hubona et al., 1999] and helps reducing visual strain on the observers.

In the case of motion blur, the gains are less substantial, because while there is almost always motion within visual effects renders, it is usually very non-uniform, with many regions of the image remaining almost static. Nevertheless, the fact that the overhead of the technique is negligible precisely in cases where there is little to no motion allows us to keep it enabled in our default workflow and reap the efficiency benefits for regions of renders that have significant motion. Depth-of-field receives some of the most impressive benefits from this technique, and we have seen render time reductions of over 70 percent in some cases with extremely out-of-focus features.

We should point out that the development of this system within *Manuka* has been made immensely easier by the path sampling architecture described in the previous section. Because the entire path sampling state is available, we have been able to implement this system completely as a group of post-sampling techniques, applied only after the *base* path has been sampled entirely. The vertex-graph interface allows us to query for any required quantities without having to worry about whether they've been computed or not, as evaluation occurs lazily, keeping efficiency high. New path vertices created by the techniques are simply added to the vertex graph and connected to the appropriate existing vertices, allowing the renderer to evaluate the shifted paths without any modification.

4.2 Efficient next-event estimation in Manuka

Next-event estimation is one of the classic techniques that help to reduce the variance of pure path tracing. The main idea is to directly sample a position on an emissive surface in the scene and explicitly connect it to a sampled path vertex, as opposed to sampling an outgoing direction and continuing the free path sampling. It isn't hard to see how this can be much more efficient in some cases, like for example for emitters that subtend a very small solid angle as seen from the path vertex, which would be extremely hard to hit by chance. Combining next-event estimation with traditional BSDF sampling can be done using MIS, resulting in a robust path sampling algorithm that is reasonably good in most lighting situations Veach [1997].

Arguably the most important part of a next-event estimation technique is the one that determines the emissive location that is sampled. The idea is to sample it with probability proportional to its contribution to the specific path vertex we are trying to connect to, as that will minimize the variance of our estimator. This is not too hard to do for some simple cases; the incident radiance due to diffuse emitters, for example, can be evaluated in closed form for polygonal sources; for Lambertian surfaces, the reflected radiance doesn't depend on the incoming and outgoing directions other than by the orientation cosine term, which is well behaved. If the number of emitters in a scene is limited, a possible implementation can even simply sample *all* of the emitters, guaranteeing low variance even while giving up some computational efficiency from evaluating the illumination from poorly contributing sources.

Unfortunately the reality of production rendering for visual effects is almost always more complex. Surfaces have beautiful but complicated material definitions with interesting glossy appearance, there may be hundreds or thousands of independent emitters, and the emission itself may be focused in some way to simulate the effect of spot lights or other real-world fixtures. This makes the implementation of a good, robust next-event estimation technique challenging for any visual effects studio planning to make use of path tracing in production. It is therefore incredibly exciting to see recent work within this area: Estevez and Kulla [2018] describes the approach taken by Sony Pictures Imageworks to solve the challenge of next-event estimation with many light sources; Vévoda et al. [2018] introduces the use of bayesian regression to increase robustness and reduce noise in many-light sampling, while Atanasov et al. [2018] improve sampling of distant sources by using a visibility cache.

4.2.1 A history of NEE in Manuka

The scalability of sampling on the number of emitters is of particular importance for us at Weta Digital. The architecture of *Manuka* relies on tessellating all surfaces down to micropolygon level, which of course includes light sources. This means that even for trivial scenes we usually end up with many thousands of emissive primitives. In complex situations, this number can easily climb to the *millions*. Any method we used to implement next-event estimation in *Manuka* needed to be able to scale accordingly.

Our first implementation started off by building a hierarchy over all emissive micropolygons in a scene, similar to a BVH, where each node in the hierarchy stored a small amount of relevant information aggregated over all primitives within its subtree: the aggregated light flux and an approximate value for the angle of maximum variation in normal orientation, as well as the axis-aligned bounding box. A second step then created a global *cut* through this hierarchy which stored the 16 highest-level nodes for which the flux was most similar. This was done by starting at the root node and descending into its children, then further descending into the child node with the highest flux, continuing until we gathered the 16 nodes we were interested in.

Sampling then proceeded in three distinct steps. First, we evaluated an approximate measure of the incident radiance $L(\mathbf{x}, \leftarrow)$ towards the position $\mathbf{x}$ we're sampling from, which was derived from the stored light flux amount multiplied by the solid angle subtended by the node AABB. At this point we could also cull away nodes that would fall completely below the upper hemisphere at $\mathbf{x}$ (if it was a surface), as well as nodes whose surface orientation information would tell us that it would not emit any illumination towards $\mathbf{x}$. The evaluations would be gathered into a CDF, which would then be sampled to obtain the subtree from which to continue sampling. From there, we would keep evaluating $L(\mathbf{x}, \leftarrow)$ for the left and right child of each node and sampling one in proportion to those measures, until we reach a leaf node containing actual emissive geometry, usually in the order of 6 to 8 micropolygons. Finally, we would estimate $L(\mathbf{x}, \leftarrow)$ due to each emissive element and sample again proportional to that measure. This method worked quite well and, because of its hierarchical nature, allowed NEE sampling in *Manuka* to scale sub-linearly with the number of emitters. However, we had strong intuitions that we could do better than this, which led to the first rewrite of our light-hierarchy sampler in early 2015.

Our main observation was that a single, global cut through the hierarchy was probably suboptimal in many cases, in particular when the distribution of emitters around a scene was highly irregular, and that having a dynamic cut could instead improve the sampling efficiency and reduce noise. We modified our algorithm to discard the global cut creation and instead the algorithm would create a new, bespoke cut when we drew a sample from the light hierarchy. This relied on three methods: the first would take a node in the hierarchy and cull it using the node clustered normal as well as the surface normal at the sampling position $\mathbf{x}$, to avoid expending any processing on nodes that would provably not contribute any illumination; the second method would quickly gauge whether a given node was suitable for evaluating directly, and the third would estimate $L(\mathbf{x}, \leftarrow)$ due to a node towards $\mathbf{x}$. The cut creation algorithm worked in a recursive fashion: starting from the hierarchy root node, we would first try to cull the children nodes. Nodes not thus culled would be tested for evaluation suitability; if this test was positive, then the node became part of the cut, otherwise we would descend into its children and repeat. Once descent terminated, such that all tested nodes were either culled or became part of the cut, we evaluated $L(\mathbf{x}, \leftarrow)$ for all cut nodes and built a sampling CDF using those evaluations. This CDF would be sampled as described in the previous paragraph, and the rest of the sampling algorithm remained unchanged.

A second important observation was that by considering all emissive micropolygons as a single "soup" we were discarding too much information. Considering instead that it's a collection of fixtures, each of which contains one or more emissive primitives, would allow us to improve both how the node hierarchy is built and to use more specific and accurate methods to estimate $L(\mathbf{x}, \leftarrow)$ from any given node; for example, knowing that a node contains a spherical emitter tells us immediately that at least half of its aggregate flux can't reach any given location in a scene, and it also allows us to bound it more accurately than by using an AABB. We'll discuss later how we estimate $L(\mathbf{x}, \leftarrow)$ for different types of fixtures later. This observation also gave rise to the current way in which we build the light hierarchy, which we'll go into detail further on. We have continued to refine and improve various aspects of our light hierarchy through the years (and there remain many new ideas for improving it further). Now let's dive in to some practical details.

4.2.2 Building a light hierarchy

The first step towards sampling a light hierarchy is of course to build it. As detailed previously, *Manuka*'s light hierarchy in its current incarnation is a hierarchy of hierarchies, with bottom-level trees built over fixture collections of micropolygons, and the top level tree being built over them. Apart from the benefits outlined above with respect to sampling efficiencies, this also allows us to support instanced fixtures through the use of intermediate reference nodes. We would be remiss not to thank and acknowledge the work of our colleague Shijun Haw, who has been the main developer of our instancing system and who spent countless days making sure instanced lights work correctly within our light hierarchy implementation.

First some definitions. Let p_i be the i-th emissive micropolygon and f represent a set $\{\mathrm{p}_0, \mathrm{p}_1, \ldots, \mathrm{p}_k\}$ that we have so far referred to as a fixture. S will be the set $\{\mathrm{f}_0, \mathrm{f}_1, \ldots, \mathrm{f}_l\}$ of all fixtures in the scene.

$\forall \mathrm{f} \in \mathrm{S}$, we first go through all $\mathrm{p}_i \in \mathrm{f}$ and gather some information for each; we initialize its material and evaluate the radiant exitance $\mathrm{M}(\mathrm{p}_i)$, converting it to flux $\Phi(\mathrm{p}_i)$ which we can then aggregate across primitives. We also evaluate and cache the emissive material lobe's exponent factor, which is used by artists to focus light emission and corresponds to a Phong lobe's exponent parameter, as well as the primitive's geometric normal $\bar{n}(\mathrm{p}_i)$ and its axis-aligned bounding box B. For geometric quad lights, we also build and store an ortho-normal basis that serves to orient the space in which the hierarchy will be built.

Once this step is complete, we build the hierarchy for f using a bespoke BVH-inspired binned top-down recursive builder. We spatially partition the primitives into 32 bins across the three cardinal axes, except that for quad area lights we do this in only two dimensions in the fixtures' tangent space, with the tangent and bi-tangent oriented according to the principal axes of f. Constraining ourselves to a single fixture for simplicity of exposition, we let $B(\mathrm{f})$ be the axis-aligned bounding box for the fixture and say that $B(\mathrm{f})_x = [x_{min}, x_{max}]$ describes its spatial extents along the x axis. We also define $E_x := x_{max} - x_{min}$ to be its range. We can then define that

$$b_{x,j} := [x_{min} + j\frac{E_x}{32}, x_{min} + (j+1)\frac{E_x}{32}]$$

describes the j-th bin along the x axis. Analogous definitions hold for the other axes. We also define a function that tests whether a primitive belongs to a bin:

$$Bin(\mathrm{p}, b) := \begin{cases} 1 & \text{if } \mathrm{p} \in b \\ 0 & \textit{otherwise} \end{cases} \tag{162}$$

For each bin, we compute the aggregate flux

$$\Phi(b) = \sum_{i=0}^{k} Bin(\mathrm{p}_i, b)\Phi(\mathrm{p}_i) \tag{163}$$

and average geometric normal

$$\bar{n}(b) = \mathrm{normalize}(\sum_{i=0}^{k} Bin(\mathrm{p}_i, b)\bar{n}(\mathrm{p}_i)) \tag{164}$$

For the case of arbitrary geometric fixtures we have a second step that cycles through the primitives one more time in order to compute the cosine of the maximum deviation for any primitives' normal to the average one for the bin

$$c(b) = \min_{\forall \mathrm{p} \in b} (\bar{n}(\mathrm{p}) \cdot \bar{n}(b)) \tag{165}$$

We must then select the partitioning point. Our initial implementation did this using only the aggregated bin flux values, which resulted in a nicely balanced tree from that point of view, but we have found that we can do better by taking a few more things into account. Let l be a possible partitioning bin, where $0 < l < 32$, and we say that the left side then corresponds to all primitives that belong to bins $[0 \ldots l)$ and the right side to those belonging in bins $[l \ldots 32)$. First we compute the ratio of flux in the left and right side to the total flux being considered

$$\begin{aligned} \mathit{fluxRatio}_L &= \frac{\sum_{j=0}^{l-1} \Phi(b_j)}{\Phi(\mathrm{f})} \\ \mathit{fluxRatio}_R &= \frac{\sum_{j=l}^{31} \Phi(b_j)}{\Phi(\mathrm{f})} \end{aligned} \tag{166}$$

where we've abused our notation a bit and use $\Phi(\mathrm{f})$ to denote the aggregate flux over all primitives in f. We also compute two penalty factors, one that depends on the maximum normal deviation from the average normal, as well as a measure of the anisotropy of the bounds. For the normal deviation we have:

$$\begin{aligned} \mathit{Penalty}_{\bar{n},L} &= 1 + 3\sqrt{1 - (\min_{j=0 \to l-1} c(b_j))^2} \\ \mathit{Penalty}_{\bar{n},R} &= 1 + 3\sqrt{1 - (\min_{j=l \to 31} c(b_j))^2} \end{aligned} \tag{167}$$

while the anisotropy penalty is simply the ratio between the maximum and minimum extents of the bounds for each side:

$$\begin{aligned} \mathit{Penalty}_{A,L} &= \frac{\max(E_{x,L}, \max(E_{y,L}, E_{z,L}))}{\min(E_{x,L}, \min(E_{y,L}, E_{z,L}))} \\ \mathit{Penalty}_{A,R} &= \frac{\max(E_{x,R}, \max(E_{y,R}, E_{z,R}))}{\min(E_{x,R}, \min(E_{y,R}, E_{z,R}))} \end{aligned} \tag{168}$$

where we've again abused our notation a bit and denote by $E_{x,L}$ the extents along the x axis of the union of the bounds for all bins belonging to the left side, likewise for the other axes and for the right side. We then evaluate the "cost" of the left and right sides and sum them up:

$$\begin{aligned} \mathit{Cost}_L &= \mathit{fluxRatio}_L \times \mathit{Penalty}_{n,L} \times \mathit{Penalty}_{A,L} \log_4(\sum_{j=0}^{l-1} \sum_{i=0}^{k} Bin(\mathrm{p}_i, b_j)) \\ \mathit{Cost}_R &= \mathit{fluxRatio}_R \times \mathit{Penalty}_{n,R} \times \mathit{Penalty}_{A,R} \log_4(\sum_{j=l}^{31} \sum_{i=0}^{k} Bin(\mathrm{p}_i, b_j)) \\ \mathit{Cost}(l) &= \mathit{Cost}_L + \mathit{Cost}_R \end{aligned} \tag{169}$$

We include the base-4 log of the number of primitives within each side, as that roughly translates into the cost of later traversing down that subtree. This causes the builder to prefer building shallower subtrees when appropriate, which improves sampling performance. We choose the splitting axis and bin that minimizes the overall traversal cost as defined above, and partition the primitives accordingly.

We have thus described the first step of our fixture hierarchy builder. The builder then proceeds recursively, executing the same procedure on the left and right sub-sets of primitives, until 8 or fewer primitives remain in the working set. At that point, a leaf node is created, which stores references to each primitive it includes. Leafs are aggregated in sets of 4 as the recursion unwinds (that is, fixture-level hierarchies have a branching factor of 4), creating inner nodes which contain references to their children, the aggregate bounds, and some extra information that is used during traversal to estimate the incident radiance $L(\mathbf{x}, \leftarrow)$ at a scene position due to that node

- *directionalFlux*, which is a 6-dimensional vector containing the total flux towards positive and negative x, y, and z directions for non-planar emitters.
- *flux*, which is the total flux towards all directions.
- $\bar{n}$, the average normal within the node.
- *threshold*, the sine of the maximum deviation from $\bar{n}$ within the node.
- *exponent*, the flux-weighted focusing exponent for the node.
- *minExponent*, the minimum focusing exponent for any primitive within the node.
- *maxExponent*, the maximum focusing exponent for any primitive within the node.
- *cullCosAngle*, the cosine of the aperture half-angle for a cone used during culling.

Of these quantities, *cullCosAngle* deserves some further explanation. During traversal, being able to cull nodes that don't effect illumination on a sampling location is important to improve efficiency. In the case of focused lights, we can do this by building a cone that bounds the node's illumination; though a Phong lobe emits illumination towards its entire upper hemisphere, for even mild values of the exponent the exitant radiance quickly falls towards zero for off-axis directions. We compute the cosine of the angle within which an amount m of the emission is contained as follows

$$cullCosAngle = (1 - m)^{\frac{2}{2+exponent}} \tag{170}$$

and use that to create the bounding cone for culling. We use 99.99% for m. While this means that we sometimes cull nodes that actually emit a negligible amount of energy towards a sampling location, this is robustly handled by MIS between next-event estimation and BSDF direction sampling.

Once all fixture-level hierarchies have been built, any instanced fixtures have their reference nodes created, which include the instance's transform as well as *scaled* data for *directionalFlux* and *flux*. Note that scaling these quantities only works well if the scaling transformation is uniform; we warn users if they use non-uniform scaling transforms that the sampling quality will deteriorate.

Finally, the top-level hierarchy is built. This proceeds similarly to the fixture-level build with one important difference; our top-level hierarchy has an arbitrary branching factor, which we use to place important fixtures higher up in the tree to maximize traversal and sampling performance. The first set of important fixtures for these purposes are those whose flux is more than either 10% or $\frac{1}{N}$, whichever is higher, of the total flux of all fixtures being considered (with N the number of fixtures). It's easy to see that there can't be more than $\min(N-1, 9)$ fixtures with such high flux, and those are added immediately as children of the current node. The second set includes all fixtures whose focusing exponent is higher than 50. These are aggregated together and placed into their own sub-tree.

4.2.3 Sampling the light hierarchy

With our freshly-built light hierarchy in hand, we can now proceed to draw samples from it. The procedure for this was quickly sketched out in sec. 4.2.1 and we shall further describe it now.

The first step is to build the initial sampling CDF, which consists of the estimated values for $L(\mathbf{x}, \leftarrow)$, towards the sampling position $\mathbf{x}$, of a number of nodes within the hierarchy which collectively define a *cut* through the hierarchy. Starting at the root node, we traverse into all its children, and for each perform two operations:

Culling This operation discards nodes from which we can ascertain that no illumination will be received at **x**. First we test to see if **x** is contained within the bounds of the node, in which case we never discard it. If we have a culling normal for **x** (there may not be one if, for instance, the surface at which we're doing the sampling has both reflective and transmissive material lobes, or if **x** is on a volume), then we test all points on the node's bounds and discard it if they all lie behind the half-space defined by **x** and the culling normal. Otherwise, we check to see if we have a valid orientation normal for the node (again, this could be invalid if the normals within the node span the entire sphere, as in the case of spherical emitter fixtures); if so, we use that normal and the previously described *cullCosAngle* value to discard the node if **x** can't receive illumination from this node. For quad area lights culling is also simplified, as we can quickly test whether **x** lies in the half-space towards which the fixture illuminates and discard the node if it doesn't.

Suitability and incident radiance evaluation This operation tells us whether a given node is suitable for evaluation of $L(\mathbf{x}, \leftarrow)$ and, if so, also provides it. Suitability is determined mainly geometrically and should be thought of as an approximate "score" of how accurate we expect the estimate of $L\mathbf{x}, \leftarrow)$ to be. As in the case of culling, if **x** is contained within the bounds of the node then we say it's unsuitable, as it results in fairly inaccurate evaluation, except that if the node is a leaf node it must always be evaluated (as there are no children left to descend into), in which case they get assigned a score of ∞. If **x** is not contained within the bounds, then we compute the solid angle subtended by the node as seen from **x** and compute a suitability score:

$$suitabilityScore = \frac{0.75}{solidAngle(node)} \tag{171}$$

the formula for which we have determined empirically. This means nodes subtending a solid angle of more than $0.75sr$ receive scores lower than one, while smaller ones receive a higher score. $L(\mathbf{x}, \leftarrow)$ is given from the product of the node's *flux*, its subtended solid angle and the receiver's cosine term if there is a normal provided for it. With $\vec{d} = |(\text{center}(B) - \mathbf{x})|$ the normalized direction vector from **x** towards the bound's center we have:

$$L(\mathbf{x}, \leftarrow) = flux \times solidAngle(node) \times \bar{n}(\mathbf{x}) \cdot \vec{d} \tag{172}$$

where $\bar{n}(\mathbf{x})$ is the normal at **x**, not to be confused with $\bar{n}$, the node's average normal. If no normal is provided, then this factor is omitted. If the node includes *directionalFlux* information (as mentioned previously, we don't store this for planar emitters), then we can obtain a better evaluation,

$$L(\mathbf{x}, \leftarrow) = -\vec{d} \cdot directionalFlux \times solidAngle(node) \times \bar{n}(\mathbf{x}) \cdot \vec{d} \tag{173}$$

where we abuse dot product notation for brevity and note that we create a 3-dimensional vector from the *directionalFlux* that only keeps components that match the sign of components of $-\vec{d}$. For cases where a node also has a focusing exponent larger than 1, we improve the accuracy of the estimate by dividing the *flux* or *directionalFlux* by the *angularNorm*

$$angularNorm = \frac{2\pi}{exponent + 2}$$

and also taking into account the exponent in the cosine term for the orientation of the node (note that we have omitted this cosine term above as it is implicitly accounted for an exponent of 1 in the computation of solid angle):

$$L(\mathbf{x}, \leftarrow) = \frac{flux}{angularNorm} \times solidAngle(node) \times \bar{n}(\mathbf{x}) \cdot \vec{d} \times (\bar{n} \cdot -\vec{d})^{exponent-1} \tag{174}$$

with the same modification done for the case where we instead use the *directionalFlux*.

The *suitabilityScore* and $L(\mathbf{x}, \leftarrow)$ for evaluated nodes are stored in an array, along with a reference to the pertinent node. At this time, we cycle through all elements in the array and, if there are any for which the *suitabilityScore* is zero, we descend into their children and repeat the previous operations, proceeding in this manner until we only have elements with non-zero suitability scores or we have accumulated more than a maximum allowed number of nodes, which is set by the user and defaults to 128. If we have not yet reached this maximum number of nodes, then we proceed with a second refinement step that proceed as follows. For a number k of elements in the array, we accumulate the sum of the $L(\mathbf{x}, \leftarrow)$ values

$$L(\mathbf{x}, \leftarrow)_{sum} = \sum_{i=0}^{k} L(\mathbf{x}, \leftarrow)_i$$

then compute for each element i a *descentScore* as follows

$$descentScore_i = suitabilityScore_i \times \frac{L(\mathbf{x}, \leftarrow)_{sum}}{4L(\mathbf{x}, \leftarrow)_i}$$

and descend into the node whose *descentScore* is highest if it's greater than 1. This procedure continues until we reach the maximum number of nodes allowed or we have no nodes for which the *descentScore* is high enough, and has the effect of descending deeper into subtrees of the hierarchy that are expected to provide the most important illumination. As mentioned previously, leaf nodes have a *suitabilityScore* of ∞; these nodes are skipped during *descentScore* evaluation.

We now have an array of nodes we consider the *cut* through the hierarchy and appropriate evaluations of $L(\mathbf{x}, \leftarrow)$ for them. We build the sampling CDF array from the array:

$$\begin{aligned} CDF_0 &= \frac{L(\mathbf{x}, \leftarrow)_0}{L(\mathbf{x}, \leftarrow)_{sum}} \\ CDF_i &= CDF_{i-1} + \frac{L(\mathbf{x}, \leftarrow)_i}{L(\mathbf{x}, \leftarrow)_{sum}} \end{aligned} \tag{175}$$

and use that to select a node from which to continue the sampling procedure. This is quite simple. We descend into all children of the node and run our culling routine on them, evaluating $L(\mathbf{x}, \leftarrow)$ using eqn. (174) for nodes that are not culled, then build a CDF using those values and sample a child to continue our descent. This continues until we reach a leaf node. Sampling within the leaf node follows the same general approach, except that culling and evaluation of $L(\mathbf{x}, \leftarrow)$ is done for the primitives instead of further nodes. Once we have selected a primitive, we sample a position within it with uniform distribution, and sampling is complete. We return the sampled position to the renderer and we're done.

4.2.4 Occlusion and radiance biasing

There are a number of sources of noise resulting from next-event estimation, of which one of the most significant is the sampling algorithm's obliviousness to the existence of occlusions within a scene — that is, when one of our carefully sampled light positions actually contributes no illumination at all at $\mathbf{x}$ because there's another piece of geometry between them. In some cases the effect of this can be quite pronounced; because we sample in proportion to the incident radiance discarding occlusion effects, bright light sources can overwhelm the sampler and cause almost all samples to be drawn from within them. When there's an occluding piece of geometry between such a bright source and $\mathbf{x}$, but a secondary dimmer emitter actually reaches $\mathbf{x}$, the result is high variance as the *effective* incident radiance at $\mathbf{x}$ does not match the quantities we are using for sampling.

A second source of noise, that can be quite severe for hierarchical sampling schemes such as the one we use, is when our estimates for $L(\mathbf{x}, \leftarrow)$ for a given node end up being inaccurate, which is not too hard when we see that some nodes will invariably end up being aggregations of very dissimilar emitters, some of which may be focused, some not, and being oriented differently. If our estimates of $L(\mathbf{x}, \leftarrow)$ were correct, we'd generally expect that $L(\mathbf{x}, \leftarrow)$ for a given node would correspond roughly to the sum of the estimates of $L(\mathbf{x}, \leftarrow)$ for its children. In practice we have found that this is sometimes not the case. The end result again is a noisy estimator, as the sampled importance doesn't correspond to the actual importance we would like to sample.

So how can we deal with this problem? For a long time we didn't, as our trusty adaptive sampler came to the rescue again and allowed artists to keep receiving nicely converged images, but we knew we could do better and improve the overall efficiency of the renderer.

Our solution is to use an image-space *occlusion and radiance biasing cache* that biases the $L(\mathbf{x}, \leftarrow)$ evaluations such that they are a closer match to the *effective* incident radiance, similar to Donikian et al. [2006]. The first step towards using this is to define a structure to hold the data we want:

```
struct BiasCache
{
 struct BiasCacheElt
 {
  float    radianceBias;
  unsigned radianceBiasCount;

  float    occlusionBias;
  unsigned occlusionBiasCount;
 };

 BiasCacheElt* elts;
 unsigned     xCount;
 unsigned     yCount;
};
```

and add a pointer to it to the node structure. This pointer can be null if biasing is not enabled, or it can point to a per-node BiasCache. Each BiasCache stores and array of xCount × yCount BiasCache elements. We allocate a fixed 256MB of memory to this technique. When the light-hierarchy build phase is completed, we check to see if biasing is enabled and, if so, traverse through the hierarchy top-down in a breadth-first manner, allocating the BiasCache elements such that nodes higher in the hierarchy have a higher image-space resolution (in our case, the highest resolution is 64 × 32 elements) and progressively reducing the resolution as we move down the hierarchy until we either use up our allotted memory or all nodes have had their BiasCache's allocated. BiasCache elements are initialized so all their members are zero.

The BiasCache information is both used and updated during our sampling steps outlined previously, and we make no attempt at cross-thread synchronization to avoid its expense, but have found this does not result in objectionable artifacts in practice.

The way it operates is as follows. When estimating $L(\mathbf{x}, \leftarrow)$ for a given node, its BiasCache is queried for the relevant BiasCacheElt. This is done by projecting the sampling location $\mathbf{x}$ to the sensor that originated the path in which it exists and using the NDC coordinates of this projection as an index into the BiasCacheElt array such that

$$index = \lfloor NDC(\mathbf{x})_y \times yCount \rfloor \times xCount + \lfloor NDC(\mathbf{x})_x \times xCount \rfloor$$

We then query the stored quantities:

$$\begin{aligned} radianceBias &= \begin{cases} radianceBias & \text{if } radianceBiasCount > 32 \\ 1 & otherwise \end{cases} \\ occlusionBias &= \begin{cases} occlusionBias & \text{if } occlusionBiasCount > 32 \\ 1 & otherwise \end{cases} \end{aligned}$$

and define a new quantity

$$L(\mathbf{x}, \leftarrow)_{biased} = L(\mathbf{x}, \leftarrow) \times radianceBias \times occlusionBias$$

that is subsequently used instead of $L(\mathbf{x}, \leftarrow)$ for all sampling decisions. We avoid using the biasing information until we have done at least 32 updates for a given cache record to minimize sampling artifacts. But what is it that we store for these quantities?

In the case of the *radianceBias*, we store the average ratio between the sum of the estimates of $L(\mathbf{x}, \leftarrow)$ of a node's children and the estimated $L(\mathbf{x}, \leftarrow)$ for the node itself. If we have that N is a node and we say that $\text{child}(N, i)$ refers its i-th child and that $L(\mathbf{x}, \leftarrow N)$ is the incident radiance due to illumination from N then

$$radianceBias(N) = \frac{\sum_{i=0}^{k} L(\mathbf{x}, \leftarrow \text{child}(N, i))}{L(\mathbf{x}, \leftarrow N)} \tag{176}$$

In essence this means that we believe the estimates we get for a node's children are more accurate than the estimate we get for the node, which makes sense intuitively as a node's children aggregate fewer emissive primitives and thus there is less uncertainty when estimating $L(\mathbf{x}, \leftarrow)$ for them. Once we have this biasing factor we update the cache; the stored *radianceBias* value is updated using a running-mean operation and the *radianceBiasCount* is incremented by 1. Note that we only compute bias factors for nodes we actually traverse during sampling, which reduces the overhead of the technique but implies the cache is updated relatively slowly.

In the case of the *occlusionBias*, we store the average transmittance between **x** and hierarchy nodes. This quantity is computed by the renderer after we have returned a sampled location, so we have a secondary method that is called on the light hierarchy to update these values. During sampling traversal we build a list of indices that records the path we took through the hierarchy and update the *occlusionBias* values similarly as for the *radianceBias* case.

The BiasCache has been used in generating production renders for the past 2 years resulting in significant efficiency improvements, but we noticed there were some scenes where noise from next-event estimation seemed to *increase* when the feature was enabled. Investigating the cause of this issue we found that in some cases there is very high variance in the $L(\mathbf{x}, \leftarrow)$ estimates for some locations in the scene. When they share BiasCache records, this can result in a sharp increase in variance of the estimator. If we additionally track and store the minimum and maximum values for $L(\mathbf{x}, \leftarrow)$ for a given cache record, we can deal with this robustly:

$$radianceBias = \begin{cases} radianceBias & \text{if } radianceBiasCount > 32 \text{ } and \text{ } \frac{maxRadianceBias}{minRadianceBias} < 6 \\ 1 & otherwise \end{cases} \tag{177}$$

One major limitation of the BiasCache as we have implemented it is that it only works well for *primary* vertices, that is, the first vertex after the sensor within a path. This is not too much of a problem in practice, because variance at the primary path vertex is usually the most noticeable, but in some cases it does reduce sampling efficiency severely. A straight forward solution to this is to store the cache records in world- or scene-space, and we are investigating approaches to do that.

4.2.5 Product sampling

Another significant source of noise in next-event estimation arises when the material at the sampling position is far from matching the assumptions of our sampling routine — that is, it's not a Lambertian reflector. For cases with low material roughness this isn't much of a problem, as forward path tracing is usually efficient in that case, and MIS combination of the techniques is fairly optimal. Even then, for surfaces illuminated with very small light sources, there can be significant sampling noise. In cases where the material is in a middle regime between diffuse and specular, neither next-event estimation nor BSDF direction sampling are efficient, requiring many samples to resolve the noise.

A relatively simple solution to this problem is to incorporate material information into our sampling decisions, such that we no longer sample in proportion to the *incident* radiance $L(\mathbf{x}, \leftarrow)$ but instead sample in proportion to the *reflected* radiance

$$L(\mathbf{x}, \omega_i) = L(\mathbf{x}, \omega_o) \times f(\mathbf{x}, \omega_i, \omega_o) \tag{178}$$

where $L(\mathbf{x}, \omega_o) = L(\mathbf{x}, \leftarrow)$ with ω_o the direction towards the light fixture, ω_i the direction from **x** towards the previous path vertex, and $f(\mathbf{x}, \omega_i, \omega_o)$ the material response at **x**.

The main problem with this approach is the expense of evaluating the material, which needs to be done every time we need an estimate of $L(\mathbf{x}, \omega_i)$, and which may have many lobes. In production cases at Weta Digital, it's usual for materials to have *tens* of BSDF lobes, layered together. So how can we do this?

As we start the process of sampling the light-hierarchy, we decompose the material at **x** and inspect the individual BSDF lobes. We discard any fully specular lobes immediately, as these can have no contribution for next-event estimation samples. We query the roughness of remaining lobes and remap the value using a BSDF-specific scale that causes a roughness value of 1 to correspond roughly to a lambertian surface while a value of 0 would be perfectly specular. We have found that lobes with roughness higher than 0.9 are usually well represented by a lambertian BSDF, so we query their albedo and layering weight and store the sum of their product, across all such rough lobes, to represent the aggregate diffuse material response.

For the remaining lobes, we evaluate each BSDF towards the mirror direction with respect to the lobe's shading normal and ω_i, which results in the highest possible evaluation for each lobe. We then compare the product of this value and the lobe's layer weight to the aggregate diffuse response we computed previously; if it's larger than $\frac{1}{2}$ of the diffuse response then we say the lobe is important and store a pointer to it. If not, we query its albedo and add it to the aggregate diffuse response. We have found that this generally leaves us with only one or two BSDF lobes that will need to be evaluated during the light-hierarchy traversal, which is much more reasonable. For nodes we consider important, we also store the layer weight, mirror reflection direction and the evaluation of the BSDF we just computed.

We should note that an important aspect in making evaluation of these BSDF lobes practical within this setting has been the implementation of highly-optimized, vectorized BSDF evaluation functions that allow us to perform the evaluation for several outgoing directions at once. We have implemented these functions for the BSDFs that are most often used in production at Weta Digital; some BSDFs remain that have no such implementations but are hardly ever used. When we encounter a material that has "important" lobes that use a BSDF for which we haven't implemented the optimized evaluation method we locally disable the product sampling feature.

Armed with the deconstructed material information we are now ready to perform the sampling operation. The sampling algorithm presented previously remains mostly unaffected, except that we replace the $L(\mathbf{x}, \leftarrow)$ estimates by $L(\mathbf{x}, \omega_i)$ as in eqn. (178), though we don't directly evaluate the material but instead compute an approximation of the material evaluation using the information we have stored for it.

This is done in a couple of steps. For a given node N we first gather a number of points on the surface of its bounds — the four corners for quad area lights and the eight corners for other nodes. To these we add one more point located at the center of the bounds. If we let P be the set of all points so chosen for a node, we then have

$$f(\mathbf{x}, \omega_i, \omega_o) \approx \frac{1}{|P|} \sum_{\forall p \in P} \left(\mathit{diffuseResponse} + \sum_{\forall bsdf} \mathrm{bsdf}(\mathbf{x}, \omega_i, \omega_p) \times \mathit{layerWeight}(\mathit{bsdf}) \right)$$

where we have again abused notation slightly to say that ω_p is the direction from $\mathbf{x}$ towards a point p. Note that we had previously also stored the "important" BSDF lobe evaluations in the mirror direction. We use that here by creating a ray from $\mathbf{x}$ towards that direction and testing the bounds of N for intersection with that ray. If the test is positive, then we also add that evaluation to $f(\mathbf{x}, \omega_i, \omega_o)$. We have noticed that this is important as it biases the sampling more strongly towards emission coming from the direction towards which the material has the highest response, further reducing variance.

Another thing we noticed was that, even with the use of the optimized evaluation procedures we implemented for the BSDFs, evaluating the material response at every step where $L(\mathbf{x}, \omega_i)$ is estimated results in too high of an overhead. To alleviate this, we further keep track of the maximum and minimum BSDF evaluations towards all points within a given node. When the ratio of these quantities falls below a certain threshold for a node we choose to descend into, we locally disable the product sampling feature. We have found a threshold of 5 works well.

4.2.6 Summary

We have developed through this section a complete and functional system for generating high-quality next-event estimation samples for scenes consisting of many light sources while maintaining high efficiency.

While the sampling method described in section 4.2.3 gives good results, augmenting it with the extensions described in sections 4.2.4 and 4.2.5 delivers, in some cases, orders of magnitude higher overall efficiency, and we believe there are still many improvements possible. We are particularly interested in drawing inspiration from Vévoda et al. [2018] and modifying the biasing cache to be stored spatially rather than in image space, as well as using the bayesian framework to improve the robustness of our algorithm.

We should note that, while we have described how we do next-event estimation sampling starting from surface path vertices, we use the same light hierarchy to draw samples for path vertices sampled within participating media. Section 5.2.3 describes how this changes the quantities that need to be estimated for light hierarchy nodes.

References

Asen Atanasov, Vladimir Koylazov, Blagovest Taskov, Alexander Soklev, Vassillen Chizhov, and Jaroslav Křivánek. 2018. Adaptive Environment Sampling on CPU and GPU. In *ACM SIGGRAPH 2018 Talks*.

P. Bekaert, M. Sbert, and J. Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Proc. Thirteenth Eurographics Workshop on Rendering*, P. Debevec and S. Gibson (Eds.). 125–134.

P. H. Christensen, J. Fong, D. M. Laur, and D. Batali. 2006. Ray Tracing for the Movie 'Cars'. In *IEEE Symposium on Interactive Raytracing*.

R. L. Cook. 1986. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics* 5, 1 (1986), 51–72.

Michael Donikian, Bruce Walter, Kavita Bala, Sebastian Fernandez, and Donald P. Greenberg. 2006. Accurate Direct Illumination Using Iterative Adaptive Sampling. *IEEE Transactions on Visualization and Computer Graphics* (2006).

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* (2018).

L. Grünschloss, M. Stich, S. Nawaz, and A. Keller. 2011. MSBVH: an efficient acceleration data structure for ray traced motion blur. In *Proc. of the ACM SIGGRAPH Symposium on High Performance Graphics*.

T. Hachisuka, W. Jarosz, R. P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. W. Jensen. 2008. Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing. In *ACM Transactions on Graphics (SIGGRAPH 2008)*.

Qiming Hou, Hao Qin, Wenyao Li, Baining Guo, and Kun Zhou. 2010. Micropolygon ray tracing with defocus and motion blur. *ACM Transactions on Graphics* 29 (2010). Issue 4.

G. S. Hubona, P. N. Wheeler, G. W. Shirah, and M. Brandt. 1999. The relative contributions of stereo, lighting, and background scenes in promoting 3D depth visualization. In *ACM Transactions on Computer-Human Interaction*.

J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand. 2011. Temporal Light Field Reconstruction for Rendering Distribution Effects. *ACM Trans. Graph.* 30, 4 (2011).

E. Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford University, California.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Trans. Graph.* (2018).

A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. 2014. Hero Wavelength Spectral Sampling. In *Computer Graphics Forum (Proc. of Eurographics Symposium of Rendering)*, W. Jarosz and P. Peers (Eds.), Vol. 33. 123–131.

5 Volumes

Chris Kulla, *Sony Pictures Imageworks*

Figure 13: *Stills from Spider-Man: Far From Home. ©Columbia Pictures, 2019, featuring heavy use of volumetric elements.*

Volumes are essential to describing real world scenes as they encode what happens to rays as they travel between surfaces. This includes interaction with smoke and fire but also simpler homogeneous media like air or water that can surround the scene. Volumetric properties are also used to describe the interaction with translucent materials like marble or skin where one cannot assume that light enters and exits from the same point on the surface. In addition to the complexity of sorting out the light transport through volumes, film production rendering must rely on large datasets to match the scope of the dramatic sequences they portray. The scene depicted in figure 13 relied on several hundred distinct volume elements, totalling tens to hundreds of gigabytes on disk, per frame. These volumes must be illuminated by complex light sources (often represented as either detailed meshes or volumes themselves) that lie inside the volume, in addition to the environment lighting required for integration with the surrounding scene.

Modern path tracers must treat volumes as an integral part of the scene description so that all shadowing and inter-reflections between surfaces and volumes are properly accounted for. This poses a number of challenges for renderer authors which we hope to cover in this section.

Section 1.1 presented the formalism behind the radiative transfer equation (RTE) and some of the key physical effects it models. Our goal in this section is to focus less on the mathematical formalism of the RTE and more of the algorithmic and software design considerations one has to consider when integrating volumes into a renderer. While the text will be discussing many of the implementation decisions made in Sony Imageworks' *Arnold* renderer [Kulla et al., 2018], we will also try to contrast our design decisions with that of other production renderers such as *Manuka* [Fascione et al., 2018a], *Hyperion* [Burley et al., 2018], and others [Christensen et al., 2018, Fascione et al., 2018b, Georgiev et al., 2018] where appropriate.

An excellent introduction to volume rendering (and path tracing methods in general) can be found in the PBRT textbook [Pharr et al., 2016][12]. For a more in-depth survey specific to Monte Carlo sampling methods for light transport in volumes, we recommend the following reports: Novák et al. [2018a], Novák et al. [2018b]. Finally, Eugene d'Eon has an excellent book [d'Eon, 2016] on multiple scattering that collects references from outside computer graphics.

5.1 Intersecting Rays with Volumes

Surface intersections are typically represented with a primitive ID together with a parametric distance t along the ray that indicates where the intersection occurred. On the other hand, for volumes there is no such singular point. Instead one usually needs to know the *interval* of distances where a volume covers the ray. The natural representation is simply a pair of values: $[t_0, t_1]$. Unlike with surfaces, it is possible for multiple such intervals to occur along a ray, thus requiring the storage of a *list* of such records.

Very simple convex primitives like spheres or cubes naturally bound a region of space. Their intersection tests

[12]Now freely available in its entirety online: http://www.pbr-book.org/!

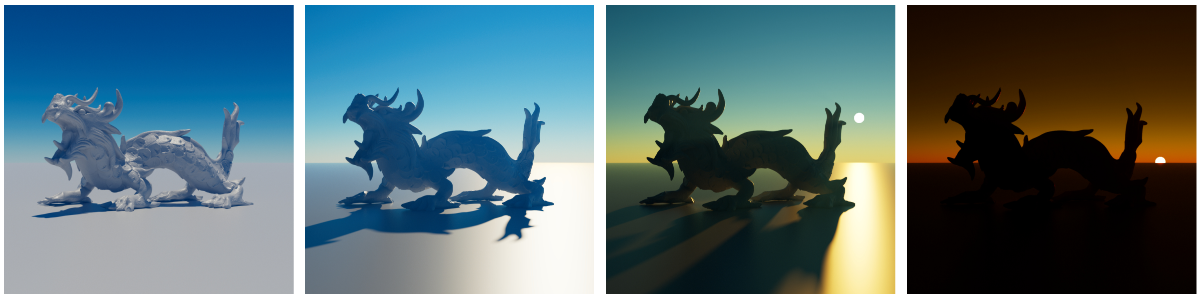


Figure 14: *Coarsely modelling the atmosphere as a volume with blue scattering density is sufficient to reproduce the main features of daytime and sunset illumination. A single distant light provides all illumination in this scene, with all fill light coming from multiple scattering inside a planet-sized sphere volume.*

already produces a pair of distances so it is very easy to adapt the code from surface intersection to volume intersection.

The most typical volume primitive used in production rendering is likely the voxel grid. The details of how such (sparse) grids are stored is detailed in Section 5.3.1. For the purpose of intersection, the main goal is to identify which groups of voxels the ray traverses, and quickly discarding rays that miss the volume entirely. A simple DDA traversal of the coarse level of the grid is usually sufficient for this purpose. We discuss perspective warped grids (frustum buffers) in Section 5.3.1 as they require a bit more care to quickly determine overlap bounds.

Meshes may also be rendered as volumes by simply collecting all hits along the ray and pairing them up. If an even number of hits is found, the ray must have started outside the model, while an odd number of hits indicates the ray started inside the mesh[13]. In the latter case the first hit is paired with the origin: $[0, t_0]$, before pairing all remaining pairs: $[t_i, t_{i+1}]$. In this context, no surface hits are generated as the ray enters the volume. Counting hits in this manner will automatically figure out the inside/outside status of a volume for every ray (which avoids the stack tracking discussed in Section 5.1.3) but does require tracing every ray all the way through the mesh which can be costly if the model has a wrinkled boundary or complex topology.

Finally, it is common for rendering software to provide a global atmosphere that encompasses the entire scene (with infinite extents). Such a volume is inherently non-physical and can be problematic to combine with distant light sources. Either the light will get fully attenuated due to the large distance, or if attenuation is ignored the ray will collect an infinite amount of in-scattered light. As such, these types of convenient features are falling out of use in exchange for simply modelling atmosphere containers around the scene. For outdoor landscapes, modelling the sky as a planet sized sphere with a proper density distribution of air particles [Kutz, 2013] will produce realistic results and allow seamless transition to planet scale rendering [Fascione et al., 2018b, Section 8.2]. Even without precise spectral modelling, artistically defined exponential densities of blue scattering particles can also produce pleasing results (see figure 14).

5.1.1 Dealing with Overlap

When a ray returns from the intersection routines, it will typically contain a list of several, potentially overlapping intervals. To enable efficient front to back ray marching, it is worth dividing this list into disjoint segments as shown in figure 15. As the number of segments overlapping any given ray is typically not too large, a basic $O(n^2)$ algorithm is usually preferable to more sophisticated alternatives that have lower asymptotic cost, but higher overhead.

It is also worth pointing out that this sorting step is only required if a front to back traversal is important. When estimating the throughput between two points (along shadow rays for instance) only the *transmittance* integral needs to be estimated which can be done on each volume independently. Therefore the overlapping segments can be marched independently without sorting them first. More details of transmittance estimation are discussed in Section 5.2.

[13]This presumes the mesh describes a closed model and that no intersections are missed for numerical reasons. This requirement for well-formed data can be a hindrance, particularly with geometry generated procedurally or through simulations. However because this method resolves overlap from scratch for every ray, any misclassification usually do not "spread" beyond a single path segment.

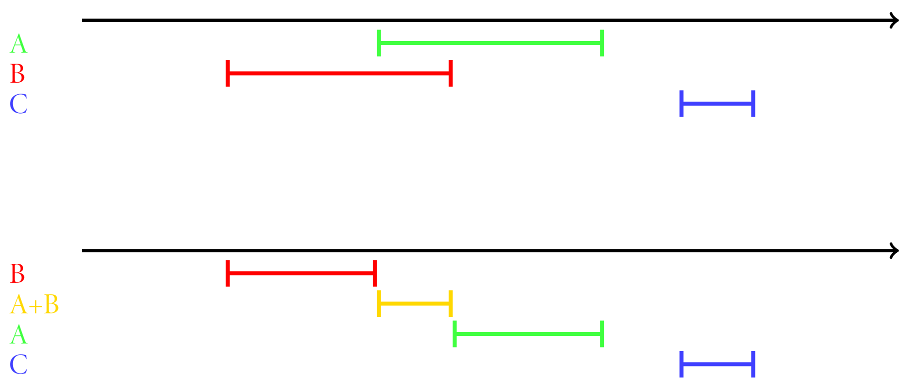


Figure 15: *A list of unsorted overlapping ray intervals representing volume hits (top) should be split into sorted disjoint segments (bottom) so ray marching can proceed from front to back.*

5.1.2 Aggregate Volumes

An alternative design to intersecting volumes by intervals and having to sort the hits into overlapping segments for every ray is to build an acceleration structure upfront over the entire scene that partitions space into regions inside of which the list of volume primitive is known. This approach [Fong et al., 2017, Section 4.4.2] has a number of advantages as it allows the raytracer to rely on non-overlapping bounds along the ray and provides a convenient place to store density bounds (discussed in Section 5.2.2). Instead of implementing a ray intersection function that returns an interval, one must implement a function that detects if a given bounding box overlaps the given primitive. Then a kd-tree (or octree) can be produced that partitions the volume primitives into smaller and smaller cells until some termination criterion is met. In contrast, the previously mentioned approach can utilize a regular bounding volume hierarchy and terminates on the granularity of a single top-level object.

5.1.3 Boundary Driven Volumes

In the previous sections, we described handling of volumes whose densities should simply be summed when overlapping. This is the desired behavior when mixing clouds, atmosphere and smoke and other such standalone media. However, when describing volumes on the interior of surfaces, *exclusive* overlap is more desirable. In other words, only a single set of volume properties should be active at a time.

This is commonly accomplished by using a variant of constructive solid geometry (csg) based on user-defined priorities [Schmidt and Budge, 2002] (see figure 16). The idea is to let artists slightly overlap meshes and leave the renderer in charge of clipping away the volumetric regions where two (or more) objects co-exist. This greatly simplifies the modelling problem, particularly for situations involving liquids inside glass. A short stack is maintained on every ray which defines what should happen when rays are transmitted to the other side of a mesh boundary. A small set of rules that take the incoming ray's stack together with the currently hit object's properties can be run to update the stack in place as well as accept or reject the current shading point. In case the hit is rejected, the ray is simply continue forwards with an updated stack. When a hit is accepted, the renderer can consult the stack to know the medium properties on each side of the boundary. This means the proper index of refraction can be computed, and the right volume properties can be applied to reflected and transmitted rays as needed.

The use of user-configured priorities is important for very precise control of which medium which win over which (see figure 17). Despite this, it is always possible to craft degenerate input that will be ambiguous depending on the viewing direction if the meshes do not represent closed volumes. If performance and ease of use are more critical than low level control, it is possible to forgo object priorities in favor of adopting a "first object wins" policy that will still render many common cases accurately [Haines and Akenine-Möller, 2019, Chapter 11]. In the implementation of our renderer [Kulla et al., 2018], we have augmented the priority based rules with a few extra cases not discussed in the original paper [Schmidt and Budge, 2002]:

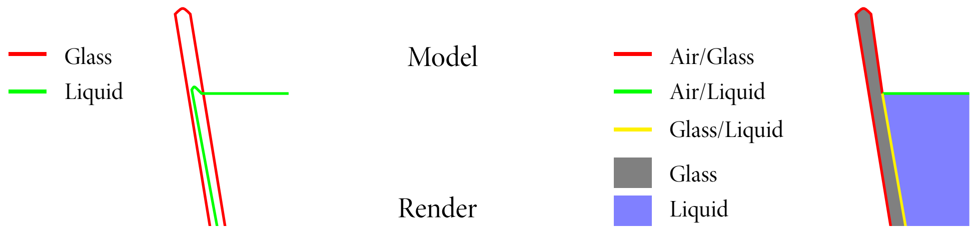


Figure 16: *Medium tracking allows a geometric model mode up of slightly overlapping meshes to turn into a proper volumetric partition of the scene with well defined interfaces between them.*

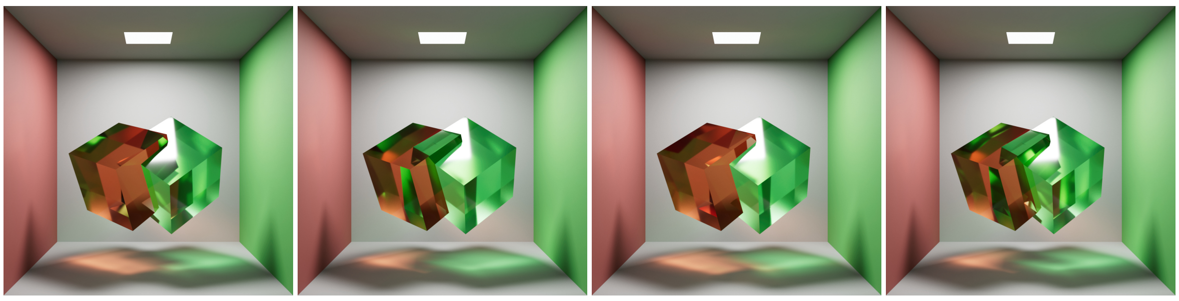


Figure 17: *Priorities give artists flexibility over which volume "wins" in overlapping regions, while the default "Off" mode improves performance when overlap handling is not critical.*

- When two objects have non-zero *equal* priorities, we discard all internal surfaces. This produces a volumetric "union" of objects modelled as separate pieces (figure 17(c))
- Priority level 0 (our default case) corresponds to always accepting the surface hit, which is cheaper to execute than consulting the stack and therefore enables early exits in the code as it overrides the priority of all other values (figure 17(d)).

To make the user interface more intuitive for artists, we present the integer priority as a simple drop down with aliased names like Off $= 0$, High $= 100$, Normal $= 200$, or Low $= 300$. We specifically chose those values to give ourselves room to fine tune priorities in the event of a complex scenario involving many nested objects. To date, it has never been necessary to use this workaround, and the set of preset values has been sufficient to render all cases we have encountered. The remapping from integers to strings has the other benefit of hiding from the artist the inverted order of priority integers (smaller numbers are higher priority) which are convenient from an implementation standpoint but can quickly lead to confusion in conversation.

One drawback of the stack-based exclusive volume overlap resolution is that it assumes the presence of a valid stack for every ray. This leave open the question of how to initialize such a stack for camera rays (or light rays in the case of bidirectional methods). Again, a heuristic must be employed here, as there may not be a single answer valid for all rays. We solve for the starting medium by firing a single ray from the camera origin upwards and searching for *all* enclosing hits. This is motivated by the main use case which is to automatically detect a camera having crossed a water surface (usually modelled without "sides"). In corner cases where the camera is close to the water surface and must see both above and below the water at the same time, the concept of a near clipping plane can be extended to enclose the camera origin in a small cube with highest priority. This emulates what the camera housing would do in the real world and permits both air and water medium to be seen at the same time (see figure 18).

5.2 Light Transport in Volumes

Once a ray has been determined to pass through one (or more) volumes, the renderer must decide how to integrate the lighting along that ray. Our description of volume integration will assume basic familiarity with surface path tracing. When implementing a path tracer, the simplest approach is to only rely on BRDF sampling and let rays find light sources by chance. While this is sub-optimal, it is easy to implement and forms a good basis for comparing to

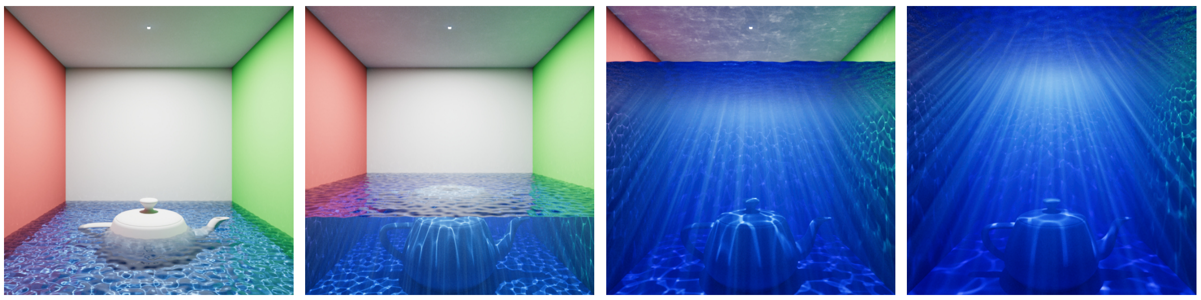


Figure 18: *Automatically resolving the initial medium simplifies the handling of animations where the camera must see both above and below the water at the same time. The water surface is a single displaced plane (it has no "sides") so render correctness depends on the initial state of the medium stack.*

more advanced techniques. We recommend following a similar approach when implementing a volumetric path tracer. The step of sampling the BRDF is replaced by sampling of the phase function, and we add a new step of randomly choosing a point along the ray for volume scattering (or skipping the volume and scattering off the next surface).

We begin with a discussion of sampling transmittance in both homogeneous (Section 5.2.1) and heterogeneous (Section 5.2.2) media, then move on to strategies for next event estimation (single scattering) (Section 5.2.3) and special techniques to accelerate multiple scattering (Section 5.2.4).

We assume the reader is familiar with the physical processes underlying the RTE detailed in Section 1.1. To quickly summarize the most important quantities are the extinction coefficient:

$$\mu_t = \mu_a + \mu_e + \mu_s$$

which gives the probability of the ray being blocked by a virtual particle per unit distance travelled (units are inverse distance). There are three sources of radiance loss: absorption (μ_a), emission (μ_e) and scattering (μ_s). These coefficients will sometimes be referred to as *density* because higher values lead to a more opaque appearance (through a greater chance of "bumping into" a particle).

The ratio between scattering and extinction coefficients is the albedo:

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

which is a unit-less quantity representing the amount of radiance transferred in a single scattering event. As long as all densities $\mu_x \geq 0$, the albedo will be a value between 0 and 1 which ensures energy conservation.

The last important quantity is the phase function, the most popular of which is the Henyey-Greenstein lobe which is easy to importance sample exactly and natively parameterized by its mean cosine g (see Section 1.1).

5.2.1 Sampling homogeneous transmittance

In a homogeneous medium, the change in radiance along the ray is proportional to the extinction coefficient μ_t. This simple fact allows us to derive the transmittance along the ray:

$$T(t) = \exp\left(-\mu_t t\right)$$

which is known as Beer's Law. This is a smooth function with a rapid falloff. This is what makes distant objects in a volume appear darker (while scattering is what leads to the hazy appearance). The natural choice when sampling a distance for further interaction is to choose a point proportionally to this transmittance function. Normalizing the transmittance $T(t)$ into a pdf we obtain:

$$\begin{aligned} p(t) &= \frac{T(s)}{\int_0^\infty T(s)\,ds} \\ &= \mu_t \exp\left(-\mu_t t\right) \\ &= \mu_t T(t) \end{aligned}$$

This gives us the following cdf $c\left(t\right)$ and sample generation function $t\left(\xi\right)$:

$$\begin{aligned} c\left(t\right) &= \int_0^t p\left(s\right) ds \\ &= 1 - \exp\left(-\mu_t t\right) \\ t\left(\xi\right) &= -\frac{\log\left(1-\xi\right)}{\mu_t} \end{aligned}$$

Using the last equation to turn a random value ξ in $[0, 1)$ into a distance tells us where we have interacted with the volume. We still need to consider the possible presence of a surface however. Supposing the nearest surface along our ray lies at distance t_{surf}, the probability of generating a distance t beyond t_{surf} is:

$$\begin{aligned} P\left(t \geq t_{\text{surf}}\right) &= \int_{t_{\text{surf}}}^{\infty} p\left(s\right) ds \\ &= \exp\left(-\mu_t t_{\text{surf}}\right) \\ &= T\left(t_{\text{surf}}\right) \end{aligned}$$

In other words, the probability of generating a volume sample behind the surface is exactly equal to the transmittance up to the surface. Therefore the weighting by transmittance cancels with the probability and we the overall path weight is unchanged if $t\left(\xi\right) \geq t_{\text{surf}}$.

In the event that $t\left(\xi\right) < t_{\text{surf}}$, we can proceed assuming that we have hit the volume and choose a scattering direction by sampling the phase function (which is analogous to BRDF sampling). The path contribution for volume scattering is $\mu_s T\left(t\right)$ divided by the pdf which is $\mu_t T\left(t\right)$. After cancellation, all that remains is the albedo α. The phase function does not appear in the weight at all as long as perfect importance sampling is possible. Notice how this very simple sampling strategy only ever creates weights between 0 and 1, and therefore all variance in the rendering will be as a result of the low probability of randomly reaching the light sources.

Already in this simple setting, it is possible to experiment with a number of simple variations such as always adding in the emissive contribution of the surface (instead of risking to randomly reject it) or increase efficiency by first sampling the volume and tracing short rays only up to $t\left(\xi\right)$.

We have glossed over the fact that all this elegant cancellation of terms only happens if μ_t is a scalar quantity as opposed to a color. We will discuss chromatic media in more detail in Section 5.2.2.

Before moving onto the more complicated case of heterogeneous sampling, we present a few more equations for cases where a homogeneous volume only has finite extents. We previously assumed the pdf is defined over the entire ray domain: $[0, \infty)$, but such infinite extents are actually physically implausible.

If the ray overlaps a volume only within the finite interval $[a, b]$ where $0 < a < b < t_{\text{surf}}$. We have the following expressions for transmittance, pdf, and sample generation (valid for $a \leq t \leq b$ and $0 \leq \xi < 1$:

$$T\left(t\right) = \exp\left(-\mu_t\left(t-a\right)\right) \tag{179}$$

$$p\left(t\right) = \frac{\mu_t \exp\left(-\mu_t\left(t-a\right)\right)}{1-\exp\left(-\mu_t\left(b-a\right)\right)} \tag{180}$$

$$t\left(\xi\right) = a - \frac{\log\left(1-\xi\left(1-\exp\left(-\mu_t\left(b-a\right)\right)\right)\right)}{\mu_t} \tag{181}$$

We recommend using the C library functions `expm1` and `log1p` to improve the numerical stability of these expression when dealing with participating media with very small μ_t (low density).

As in the case where the medium covers the entire ray, a fair amount of cancellation between the path contribution and its pdf will occur when the density is a simple scalar. Still, we recommend keeping in mind which terms are canceling each other out as it becomes important when generalizing the algorithms to handle colored extinction or additional homogeneous segments. We also point out that unlike the previous pdf defined over the entire ray, the probability of choosing the surface should now be explicitly modelled. While this may seem less convenient at first, it also generalizes better when considering a path segment that hits multiple transparent surfaces or passes through several volumes.

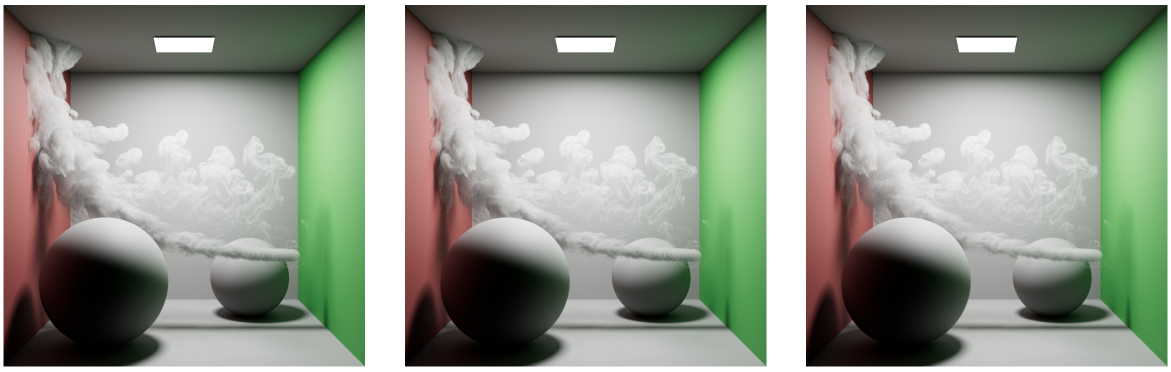


Figure 19: *The step size is quadrupled from left to right. Eventually, a small bias appears and the volume appears too bright due to insufficient transmittance. On the other hand, no extra variance is added and the render is 16 times faster.*

5.2.2 Sampling heterogeneous transmittance

When we remove the limitation that μ_t is constant across the volume, the expression for transmittance is itself an integral:

$$T(t) = \exp\left(-\int_0^t \mu_t(\mathbf{o} + s\mathbf{d})ds\right) \tag{182}$$

Here the density μ_t is a function of position and is evaluated at the parameterized position s using the ray origin **o** and direction **d**. Most of the complexity and expense of volumetric rendering comes from having to handle this nested integral.

It is worth pointing out that this integral *can* be evaluated in closed form in some cases. In particular for a voxel grid with constant or linear interpolation, or bounded volumes with exponentially varying density along a fixed direction. This approach can be useful, particularly if the voxel resolution is well matched to the amount of detail and final image resolution (see Section 5.3.1). On the other hand, this approach leaves no room for controlling performance and precludes a shader from procedurally adding detail to the density field.

Biased Ray Marching One simple way to approach the problem (at the cost of a small bias in the solution) is to turn the heterogenous volume into a collection of short homogeneous segments by ray marching. This technique was dubbed *decoupled ray marching* Kulla and Fajardo [2012] because the execution of the shaders along the ray is *decoupled* from the generation of samples for light integration.

The bias manifests as an under-estimation of the transmittance. In many cases, this can be an acceptable limitation in exchange for a very simple way to boost performance. In scenes dominated by the cost of ray marching, increasing the step size by a factor of n can make the overall rendering time decrease by that factor. Moreover, despite the extra bias that is incurred when the step size is too large, the transmittance estimate is guaranteed to lie in $[0, 1]$ and therefore no extra variance is incurred from the approximation (see figure 19).

Efficiency can be further increased by forcing uniform regions of the volume to only take a single step. When using voxel data, it is helpful to identify blocks of constant regions and automatically maximize the step size for rays passing through these regions.

To sample a ray interval $[a, b]$ with a step size Δ, only two (stratified) random numbers are required:

$$\begin{aligned} n &= \left\lfloor \frac{b-a}{\Delta} + \xi_1 \right\rfloor \\ t(i) &= a + \frac{i + \xi_2}{n}(b-a) \end{aligned}$$

The first equation computes how many steps to take, with random rounding to avoid visible transitions in the number of steps. The second equation simply uniformly distributes jittered points along the ray interval. Being able to use stratified points is another key advantage of this biased approach.

Once all steps through a volume have been taken, we are left with a long array of short homogeneous segments. We can importance sample the transmittance by first choosing a segment and then a point within it using Equations 179, 180 and 181. In fact one can do better than simply sampling the transmittance and sample $\mu_s T\,(t)$ instead which improves distribution of samples near the entrance of a volume. The discrete pdf can even be extended to include surface hits as well. This gives a unified way of treating both volumes and surface transparency (which presents many of the same challenges).

Unbiased Ray Marching Despite some of the advantages highlighted in the previous section, removing the bias from the method described above can still be desirable to avoid the potential for misconfiguration of a step size. To evaluate (or sample) from equation 182 in an unbiased way we must rely on the *null-scattering* formalism introduced to computer graphics by Kutz et al. [Kutz et al., 2017]. The idea is to fill up the heterogeneous volumes with extra particles so as not to change the light transport, but permit the use of the equations for homogeneous volumes. The only requirement is to have some guess on the largest possible value $\mu_t\,(\mathbf{x})$ might take on throughout the scene (or for any given region of a scene).

The central idea of null-scattering is to add a new kind of particle (so called *null* particles) with density μ_n to the existing scattering, emissive and absorptive particles (which have density μ_s, μ_e and μ_e). Unlike the others, hitting such a particle has no effect on radiance (it simply flows forward, unchanged) which guarantees there will be no change to the radiance distribution. Starting from an overall bound $\overline{\mu}$ on density, we have:

$$\mu_n\,(\mathbf{x}) \quad = \quad \overline{\mu} - \left(\mu_s\,(\mathbf{x}) + \mu_a\,(\mathbf{x}) + \mu_e\,(\mathbf{x})\right)$$

In other words, the density of null particles is implied by what is left over after accounting for the actual extinction. For any arbitrary point inside the volume we can define the probability of choosing any of the four event types as:

$$\begin{aligned}
p_{\text{scatter}}\,(\mathbf{x}) &= \frac{\mu_s\,(\mathbf{x})}{\overline{\mu}} \\
p_{\text{absorb}}\,(\mathbf{x}) &= \frac{\mu_a\,(\mathbf{x})}{\overline{\mu}} \\
p_{\text{emit}}\,(\mathbf{x}) &= \frac{\mu_e\,(\mathbf{x})}{\overline{\mu}} \\
p_{\text{null}}\,(\mathbf{x}) &= 1 - p_{\text{scatter}}\,(\mathbf{x}) - p_{\text{absorb}}\,(\mathbf{x}) - p_{\text{emit}}\,(\mathbf{x})
\end{aligned}$$

The importance of defining $\overline{\mu}$ as a true upper bound is visible in the last equation: it avoids having a negative probability for p_{null}. It is possible to avoid the negative probabilities by reformulating the equations slightly [Kutz et al., 2017] which greatly improves the applicability of the method, though deviations from the bounding value can increase variance substantially. In practice, it is recommended to compute and store tight upper bounds for density together with the voxel data to avoid such issues. For procedural volumes it is recommended to estimate upper bounds in a spatially localized fashion before rendering begins.

Defining a random walk through the volume is now as simple as generating candidate locations along the ray according to density $\overline{\mu}$, followed by a stochastic decision on which type of event to follow depending on the results of the shading lookup at the chosen position. Each step through the volume will require two independent random variables (which prevents straightforward stratification, unlike the biased scheme).

The random choice can also be replaced by a change in the path weight to encourage deeper exploration instead of stochastic termination (which only produces binary estimates). The difference between the weighted approach and the stochastic one was first explored in computer graphics in the context of transmittance evaluation [Novák et al., 2014] before being generalized to full random walks [Kutz et al., 2017]. Recent surveys [Novák et al., 2018a,b] gives a more thorough accounting of the lineage of these methods.

A weakness common to these unbiased approaches is the impact of $\overline{\mu}$ on overall efficiency. The more conservative it is, the smaller the average step will be. As a result, it is critically important to breakup the volume into subregions whose density can be bounded accurately. Of course, a similar requirement exists for biased approaches where the importance of the step size is directly related to the total variation in density within a region of space.

Decomposition Tracking The basic unbiased ray marching scheme above takes steps proportionally to the mean free path $\overline{\mu}$ and therefore can become quite expensive in very dense media as each step must probe the spatially varying density.

The decomposition tracking approach [Kutz et al., 2017] observes that we can importance sample from two (or more) overlapping mediums by sampling a distance from each proportionally to transmittance and simply taking the smallest of the resulting distances (a detailed proof is provided in the supplemental material of that paper). This has a few very interesting consequences from an engineering standpoint.

Firstly, when multiple volumes overlap it is possible to sample a distance from each one proportionally to transmittance independently. The distance from the first sampler can provide an early exit distance for the following. This type of early exit is very beneficial and also mimics how nearest hits are processed in a ray tracer[14].

Secondly, if we have a *lower* bound (usually termed *control bound*) on the total volume density of a given volume (in other words, density never falls below a certain value μ_c) we can take samples from this cheaper control component and use the found distance to take fewer steps in the *residual* component of the volume (the density left over, which must be in the range $[\mu_c, \overline{\mu}]$. In practice, this means that many steps through dense regions of the volume only need to sample the (homogeneous) control volume which can significantly reduce the number of lookups (or shader executions) required. As with the computation of $\overline{\mu}$, the control component μ_c can be stored within the acceleration structure and should be as tight as possible to maximize performance.

Spectral Ray Marching We have so far concentrated on describing techniques as if the volume density was equal for all wavelengths. But supporting colored densities is critical to describing common participating media such as atmosphere, water, or skin.

Even in homogeneous media, the nice simplifications between path contribution and pdf are no longer possible when the densities are colors because the pdf itself must always be a scalar. The solution to this problem is to incorporate multiple importance sampling to combine the probability of sampling from each wavelength separately. Repeating the equations from Section 5.2.1, we indicate which quantities are spectrally varying by incorporating the wavelength λ:

$$\begin{aligned} T(t,\lambda) &= \exp\left(-\mu_t(\lambda)\, t\right) \\ p(t,\lambda) &= \mu_t(\lambda)\, T(t,\lambda) \\ t(t,\lambda) &= -\frac{\log(1-\xi)}{\mu_t(\lambda)} \end{aligned}$$

We therefore have a pdf for each of the n wavelengths being carried together along the path. In traditional RGB rendering the wavelength λ is just a placeholder for the three color components ($R = \lambda_1, G = \lambda_2, B = \lambda_3$). To choose which of the n pdfs we will actually use to draw a distance t, we must first select among the three wavelengths according to some discrete probability: $P(\lambda_i)$ and combine everything together using the one-sample MIS technique, which is equivalent to use a combined pdf of:

$$p(t) = \sum_{i=1}^{n} P(\lambda_i)\, p(t,\lambda_i)$$

The discrete probabilities should incorporate how important each wavelength is to the overall path. In a unidirectional path tracer, it is worth using both the path weight as well as the albedo [Chiang et al., 2016, Kutz et al., 2017] to minimize variance.

When moving to the heterogeneous setting, the biased approach of treating the medium as a collection of small homogeneous segments can be extended to choose both the wavelength and the segment which leads to very high quality samples as the transmittance is very accurately sampled among all channels.

When using unbiased trackers, more care is required as the choice of probabilities is incorporated into the ray marching process itself. Kutz et al. [2017] adapt their weighted tracking schemes to work on multiple wavelengths at once by conservatively extending the interval bounds $\overline{\mu}$ to cover the maximum across all wavelengths.

[14]In fact, early work on brute force volumetric path tracing [Morley et al., 2006] took advantage of this trick to incorporate volumes directly into the ray tracer by letting the stochastic distance sampling act like a surface ray intersection routine.

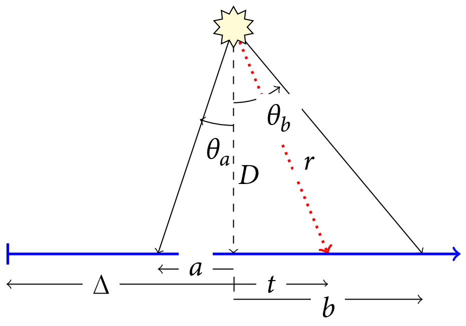


Figure 20: *Quantities involved in computing scattering from a point light source*

This restriction can be lifted [Miller et al., 2019] if one considers the null particles as part of the path integral formulation itself, which permits reasoning about the path probabilities even for unbiased trackers (as opposed to just the weight).

5.2.3 Next Event Estimation in Volumes

The techniques discussed so far allow the generation of samples along the ray for further scattering in a "blind" way. No information outside of what is known along the ray is used. This enables only the simplest variant of path tracing where the lights are intersected by chance. A natural first improvement on this is to run next-event estimation from the chosen t value along the ray and combine sampling of the light source solid angle with sampling of the phase function (as is done with the BRDF in surface rendering).

Unfortunately, this scheme will not bring as large of a variance reduction as it does on surface because the radiance does not vary smoothly along the ray. Therefore the choice of the distance t that we sample the lights from would ideally take the light source position into account as well.

Equiangular sampling for point lights

A simple point light source embedded in a volume is a good case study for this problem. The $1/r^2$ decay of the point light causes a hotspot of high contribution near the source, with a rapid falloff away from it. Positioning samples purely by looking at the transmittance (as we have discussed up until now) is counterproductive as only points close to the light will have a high contribution.

Equiangular sampling [Kulla and Fajardo, 2012] is a way to choose distances along the ray proportionally to the radiance distribution of a point source (the $1/r^2$ term). This allows the geometric term of the path contribution to cancel with the pdf, giving much better variance reduction for such paths.

The basic geometric configuration is shown in figure 20. To keep the equations relatively simple, we shift the origin of the ray by Δ to be underneath the projection of the point onto the ray. From there we assume that integration takes place in the range $[a, b]$. The distance t along the ray can be decomposed into: $t^2 + D^2 = r^2$ where D is the shortest distance between the point light and the ray.

Solving for a pdf proportional to $1/r^2$ leads to the following equations for the pdf and sample generation functions:

$$\begin{aligned} \text{pdf}(t) &= \frac{D}{(\theta_b - \theta_a)(D^2 + t^2)} && (183) \\ t(\xi) &= D \tan\Big((1-\xi)\theta_a + \xi\theta_b\Big) && (184) \\ \theta_x &= \tan^{-1} x/D && (185) \end{aligned}$$

From the final form of the sample generation function $t(\xi)$ we observe that a linear interpolation of the angles towards the integration bounds $[a, b]$ appears. Thus, equal steps along the ray take equal steps in *angle* and the strategy is therefore termed *equiangular sampling* Kulla and Fajardo [2012].

This sampling technique greatly improves convergence when lights are directly embedded in a volume compared to only relying on next-event estimation from distances sampled according to transmittance (see figure 21).

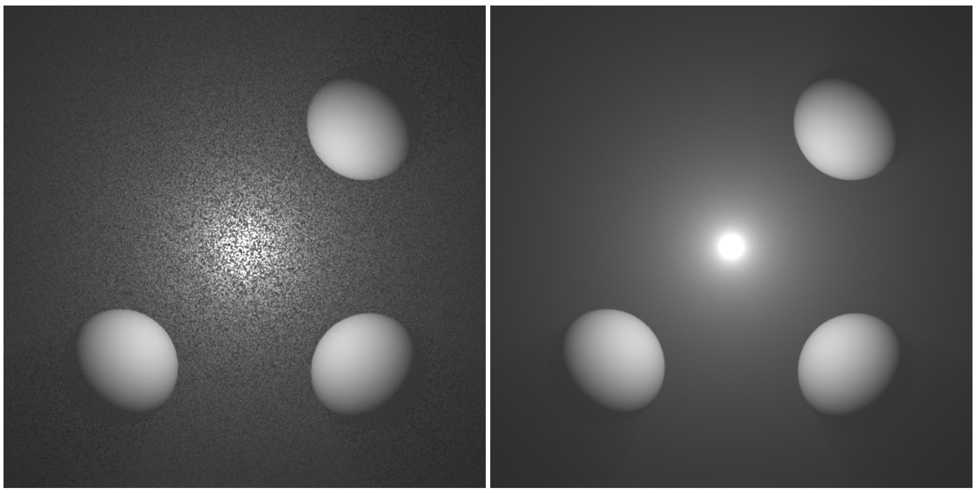

(a) Transmittance Sampling (b) Equiangular Sampling

Figure 21: *Point light source in homogeneous media (16 paths/pixel)*

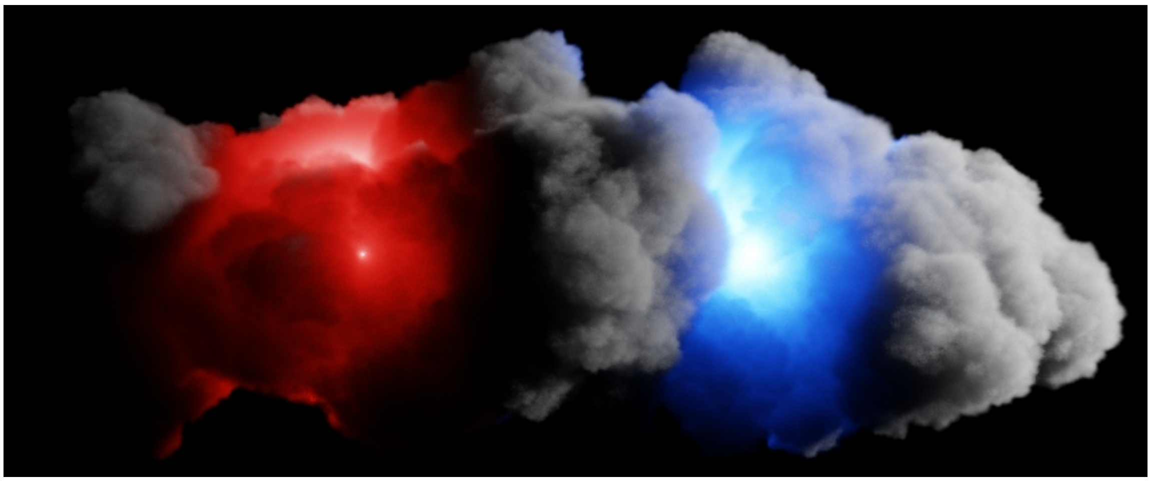


Figure 22: *Heterogeneous medium rendered using decoupled ray marching which allows the evaluation of pdfs and therefore combines well with equiangular sampling (16 paths/pixel)*

As with spectral distance sampling (Section 5.2.2), multiple importance sampling is still important to guard against more complex cases, particularly in dense heterogeneous media. The only requirement is to be able to evaluate the pdf for each technique individually. Decoupled ray marching Kulla and Fajardo [2012] was in part designed specifically to combine easily with equiangular sampling: see figure 22, 23. Very recent work has shown that a similar combination is possible for unbiased ray marching techniques as well [Miller et al., 2019].

The basic principle of equiangular ray marching carries over to area lights as long as importance sampling from the chosen point can be done proportionally to the solid angle Ω, since it also varies as $1/r^2$. In cases where solid angle sampling is difficult to perform, one can choose a point on the surface of the light first and then apply equiangular sampling from the chosen point.

Equiangular sampling for many lights Production renderers must contend with large numbers of light sources. The volumes from figure 13 are illuminated from within by multiple mesh lights, each composed a few million triangles. Naively looping over all lights in turn is not practical, nor is simply selecting a lights randomly according to their power (again due to the rapid distance falloff). Most production rendering instead relies on a lighting hierarchy (see Section 4.2 for a discussion of the implementation in *Manuka* [Fascione et al., 2018a]). Here we briefly discuss the light hierarchy choices made in Sony Imageworks' *Arnold* renderer [Conty and Kulla,

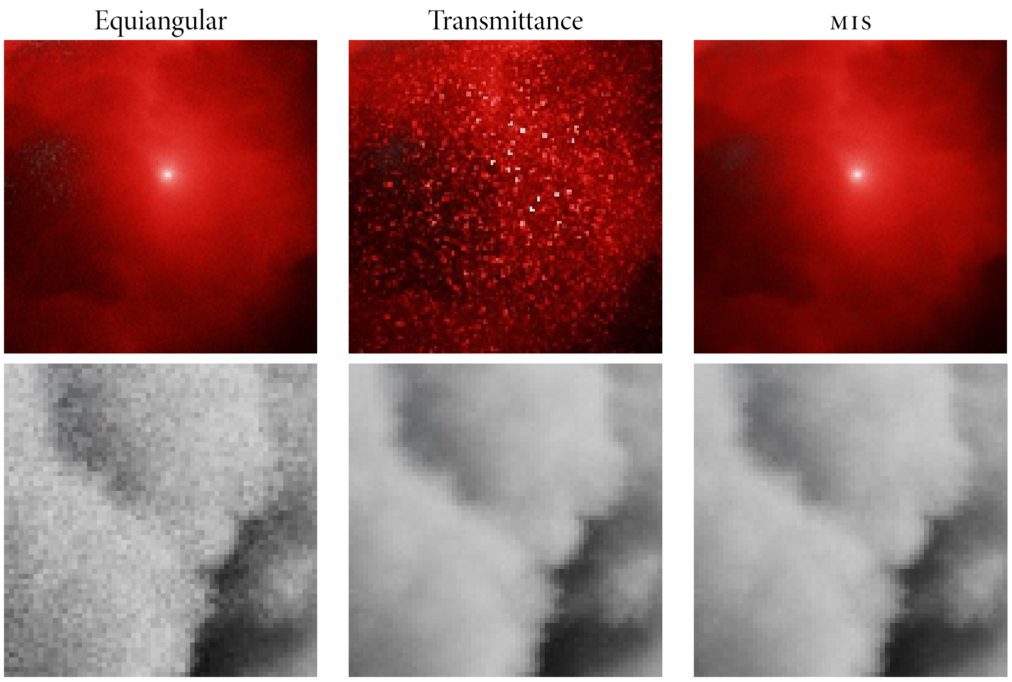


Figure 23: *Closeups of figure 22. MIS combines the strengths of each line-sampling technique.*

2018].

On surfaces, the light hierarchy tries to select light sources proportionally to their distance and intensity to the current shading point. In volumes on the other hand, we must select a light source *before* knowing where along the ray we will place our sample. Therefore, the query is driven by the ray beam instead of a single point. The hierarchical estimate must instead try to minimize variance while knowing the $1/r^2$ term in the final integral will be canceled out by equiangular sampling. After division by the pdf (see section 5.2.3), the dominant term on variance will be the normalization constant D which is different for every light. The light picking therefore tries to select lighting proportionally to $1/D$ with the aim of canceling out with this normalization factor after the light is chosen. An example of the light hierarchy for a mesh light is shown in figure 24.

5.2.4 Beyond single scattering

Doing a full simulation of all bounces of light within a volume remains an expensive proposition, even when taking into account all recent developments in importance sampling techniques. And yet the appearance of many common types of participating media depend on these extra bounces. Clouds and snow are two prominent examples as they are both made up of water and ice droplets that scatter nearly all incoming light (see figure 25).

Stochastic Path Sampling When constructing deep paths through highly scattering media (often several thousand bounces are required), one can observe that many vertices make similar contributions to the final image. The idea of decoupled ray marching [Kulla and Fajardo, 2012] can be extended from building a CDF along a single ray to building a cdf along an entire *path*. This allows skipping costly light loops and focusing more samples on those path segments that need it the most. This optimization does introduce some additional variance so it should be carefully balanced and only kick in if paths grow sufficiently long.

Similarity Theory A surprising aspect of volume rendering is that the appearance space for participating media contains some amount of redundancy. In other words, it is possible to obtain very *similar* output from different values of the core simulation parameters. The investigation of so called *similarity relations* predates computer graphics [van de Hulst, 1974]. One simple relationship that is very easy to implement is:

$$(1-g)\,\mu_s = (1-g^\star)\,\mu_s^\star$$

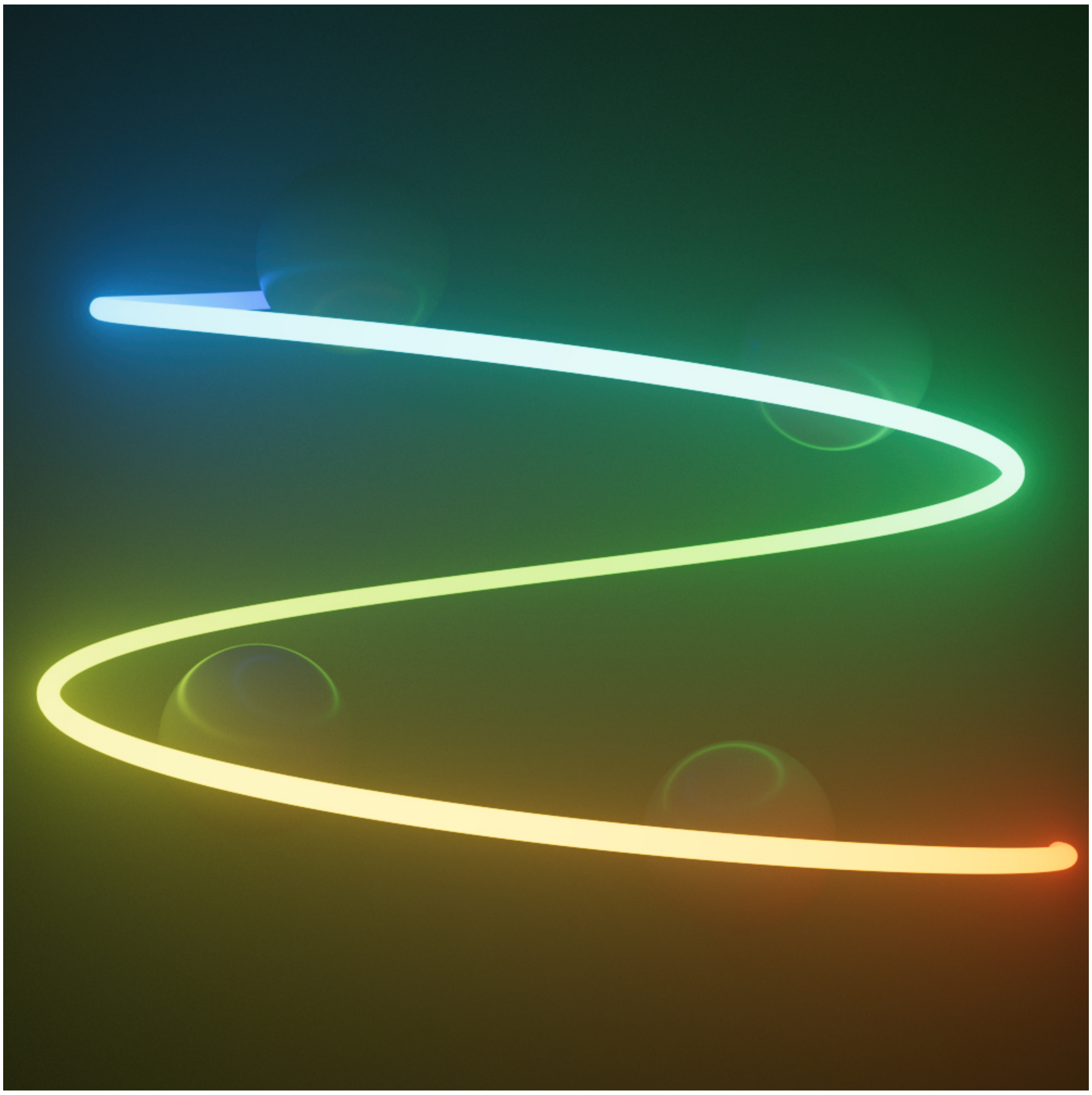


Figure 24: *A meshlight composed of many triangles illuminates a homogneous volume. Each path only connects to a single point on the mesh.*

Figure 25: *Snow and cloud rendering in the movie Smallfoot ©Warner Bros., 2018*

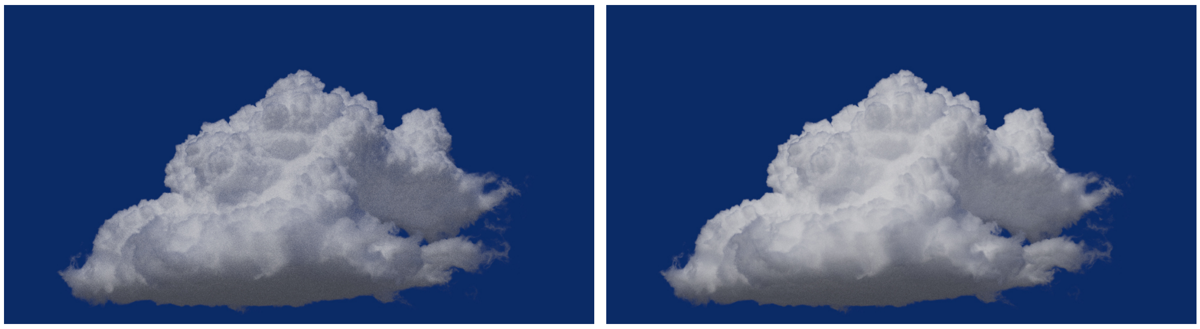


Figure 26: *Similarity theory can greatly accelerate the rendering of dense, forward scattering media such as clouds. The image on the left is rendered by brute force and still contains visible noise and reaches path lengths over 50,000. The image on the right is much more converged and reached a maximum path length of around 5000 thanks to the increased mean-free path in deeper bounces. Cloud dataset courtesy of Walt Disney Animation Studios.*

where g stands for the mean cosine of the phase function. The reduced parameter $g^\star$ is usually chosen to be 0 to replace strong forward scattering with isotropic scattering. In this case the scattering density $\mu_s^\star$ becomes smaller, increasing the mean free path which allows direct light sampling to be more effective as well as promoting the chance of escaping the medium.

Naturally these similarity relationships remain approximations. It is therefore important to deploy them gradually, usually only after a number of bounces. The *Hyperion* paper [Burley et al., 2018] describes a set of heuristics for applying this relationship in cloud rendering that is surprisingly effective (see figure 26).

Zhao et al. [2014] investigated higher order similarity relations that can further improve upon the basic linear relation shown here, though to our knowledge this work is not widely used in production rendering contexts as a custom phase function must be numerically derived through a pre-process.

5.2.5 Emissive Volumes

Explosions are an essential ingredient of modern blockbuster films (see figure 27). The spatially varying emission term in the RTE is usually accounted for by simply adding in the emission L_e during ray marching (both biased and unbiased schemes presented earlier allow this). One must remember however that the emission term typically dominates the path contribution and therefore sampling the emission term stochastically (by first sampling transmittance alone) is likely to lead to high variance. Instead it is preferable to always add in the emissive contribution [Simon et al., 2017] (usually given by the volume shader at the same time that the other densities are queried).

To accurately sample the effect of emissive media on nearby surfaces (and volumes), a 3D cdf can be con-

Figure 27: *Rendered fire simulation in Suicide Squad, ©Warner Bros. 2016*

structed to randomly select positions within the volume [Villemin and Hery, 2013]. For dense media, it is important to account for transmittance when choosing the emissive vertex as it could be obscured by dense smoke [Simon et al., 2017].

5.3 Production Features

5.3.1 Large Datasets

As mentioned in the introduction, the volumetric datasets invoked in production frequently grow to very large sizes, sometimes far beyond what is required for a particular camera distance. Some of the reasons for this discrepancy include:

- a requirement to deliver very rich visual detail leads to elements crafted at as high of a resolution as possible
- a very fluid production process, where elements are designed before camera angles are locked off.
- multiple artists working on different aspects of a complex shots, leading to many disparate elements needing to be combined at render time
- incremental look development were elements are layered together to achieve a desired look

For all these reasons, a production renderer must avoid constraining the user with limitations if they can be solved some other way. Out of core approaches have to date found little use in practice due to the nature of data access in a global illumination context. Even when just simulating direct lighting, a handful of area light sources illuminating the volume from different direction can be sufficient to require touching nearly every voxel on every progressive sampling pass.

Instead the focus is usually placed on improving the fundamental representation of the volumes in memory, and restoring view-adaptivity where possible.

Two (and multi) level grids

The simplest form of data compression is to simply to avoid storing homogeneous (particularly empty) regions. Sparse grids (sometimes called micro/macro voxels grids) are a simple, two level data structure that already brings a big savings to most cases. Instead of a dense array of $W \times H \times D$ voxels, they are broken into two levels, with a top level of macro blocks of varying resolution and a micro level of usually fixed resolution grids (usually between 8^3 and 32^3). This approach is implemented in the Field3d open source library [Wrenninge, 2009].

The *OpenVDB* volume container format [Museth, 2013] takes this principle and extends it even further, letting each micro block be itself a sparse grid of (compile-time) adjustable depth and resolution to gain even more adaptivity which can be beneficial when rendering very thin volumes or thin level sets. For the vast majority of cloud or smoke elements however, the easier to implement two level solution is frequently sufficient.

In either case, the structure naturally encodes where the volume exists or not, which provide a natural way to avoid ray marching empty regions. When using unbiased sampling techniques which require bounds on density, the coarse level of the structure is a natural place to store these bounds.

Rasterized View Dependent Volumes

Instead of supporting a multitude of formats and procedural primitives in the inner loop of the ray marcher, some renderers choose to re-sample all volumes into a common format, which also allows adapting the resolution to the camera's frustum [Fascione et al., 2018a] to minimize storage in memory. When combining this strategy with instancing, great care should be taken to be conservative in the view-dependent resolution that will be re-used across all instances. Similar trade-offs and approximations must be made for view-adaptive tessellation of instanced meshes [Kulla et al., 2018].

The benefit of supporting only a single structure is greater efficiency from a single-use data structure, as well as improved efficiency from having a voxel density exactly matched to the target resolution. The potential downsides of such an approach are the higher upfront processing cost before any rays can be fired, and the need for higher fidelity lookups when preparing the structure to avoid any visual artifacts from the resampling process.

Frustum Buffers

Another way to maximize resolution for a particular camera viewpoint is to use frustum buffers, which simply warp a 3d grid by the camera's perspective transform. This gives an elegant way to maintain constant detail in screen x and y coordinates, while reducing it along Z where it is less important. To avoid light leaking effects from off-screen, the frustum is usually slightly larger than that of the camera.

Frustum buffers are quite simple to implement as only an extra perspective division is required when mapping between world space and grid space, however they do complicate ray marching as finding which range of voxels a ray passes through is no longer a straightforward march through with DDA techniques. A straight ray becomes curved after the perspective divide. This can be solved by observing that while arbirary lines becomes curves, axis aligned planes are still planes after the perspective transform. It is therefore possible to build a kd-tree in the axis-aligned grid space, and warp it to a world space kd-tree on the fly during traversal. Axis aligned plane equations can be easily extracted from the perspective matrix, leading to a rather simple and efficient implementation in practice [Wrenninge et al., 2013].

5.3.2 Motion Blur

Motion blur is essential to rendering animated content. Most production path tracers distinguish between two types of motion: transformation and deformation. Transformation blur is typically handled by sampling the transform hierarchy at the ray's current time. This accounts for cases where the entire volume is being moved around (for example the exhaust of a jet engine). During the ray intersection phase (see Section 5.1), the matrix inverse is applied to the ray so that bounding information can be precisely applied even for moving volumes. This is identical to how transform motion blur is handled for regular shapes. Deformation blur on the other hand captures the blur caused by fast internal motion inside the volume itself. This is particularly important for explosions, which are generally highly energetic, produce motion in all directions, and expose to bright, high dynamic range pixels in the final frame.

It should be noted that great care needs to be taken when combining both forms of motion at once. The deformation blur should be computed in object space (ie: without the effect of the transform) to avoid a double application of the two types of blur. Likewise, it is important to keep track of the units in which fields such as velocity are expressed during the simulation (usually distance per second) and properly adjust them for what the renderer expects (usually distance per frame).

Eulerian Blur

The first, and simplest to implement way to obtain deformation blur is so called Eulerian motion blur. Here the volume should be paired with a velocity field describing the evolution of its contents. When sampling the volume at a position $\mathbf{x}$, one first reads $\mathbf{v}(\mathbf{x})$ to obtain the velocity before reading the actual desired shading signal from $\mathbf{x} - \delta_t \mathbf{v}$. In other words, we take a backward step along the velocity vector to backtrack to where the density was at the provided instant in time δ_t.

While this technique mirrors backtracking techniques invoked during simulation – in the context of rendering it does not always produce satisfactory results. First, in the case of very rapid motion - it is possible to backtrack off the edge of the volume, leading to clipping artifacts. In simulation, this problem can be solved by sub-sampling the simulation, which would be prohibitive to do during rendering.

Temporal Volumes A better, yet slightly more complicated to implement strategy is to store a full, time-varying density field that allows sampling points in the volume using both the position $\mathbf{x}$ and the current time t. Conceptually, every voxel of the grid must store a function $f(t)$ instead of a single scalar value. In order for this method to be tractable, the function must be compressed. The method implemented in the Field3D library [Wrenninge, 2009] (version 2.0 and higher) is inspired by deep opacity compression [Lokovic and Veach, 2000], and is detailed in the related paper [Wrenninge, 2016]. The density evolution in most volumes turns out to be very compressible and therefore an overall small number of voxels need to store a long list of values.

This method, while accurate, still requires a deep integration with the simulation process itself to properly compress the time-evolution of the volume within the sub-frame. Most simulation processes take several step within a single rendered frame and these distinct steps should all be captured into the compressed 4D output volume for optimal results.

5.4 Future Work

Despite the fact that all production path tracers in use today have some kind of volume support, a number of interesting challenges remain.

5.4.1 GPU implementation

All the methods described so far tend to map fairly well to a GPU as the bulk of the expense is typically in stepping through dense 3d textures that can be accelerated in hardware. Still, the optimal tradeoffs in sampling algorithms and representations may not be the same between CPU and GPU and algorithms will likely have to be revisited to achieve maximal performance in both cases. We expect open source libraries such as *GVDB* (GPUs) and *OpenVKL* (CPUs) to play a key role in enabling high performance implementations in the years to come, similarly to the role that Embree and Optix have played in enabling high performance surface intersections. As with the latter, achieving good performance depends as much on system level design decisions as it does on the kernels used in the inner loops.

5.4.2 Optimal Raymarching Approach

As we have touched on in these notes, there are a variety of possible estimators for sampling and evaluating transmittance in heterogeneous media. As with light transport algorithms in general, it is unlikely that there is a single approach that can be made to work optimally in all cases. The overall runtime is both a function of the number of steps taken along the average ray, but also the overall variance of the estimator (and how quickly it can be compensated by tracing more paths overall). As such, any comparison of approaches is intrinsically tied to engineering decisions made in the renderer as a whole.

Nonetheless, we believe there is more room for a principled comparison of the different (biased and unbiased) methods in a controlled fashion to determine which is preferable depending on the type of scene.

5.4.3 Correlated Media

While this course has not dived too deeply into the mathematics of the light transport, the sampling techniques mentioned in Section 5.2 are all based on the simplest form of the volume rendering equation in which the infinitesimal particles that make up the volume are assumed to be *uncorrelated*. This means the probability of bumping into a particle is independent from point to point. This leads to the simple exponential shape of the transmittance curve in homogeneous media, and keeps the heterogeneous case somewhat tractable as densities can simply be summed together to handle overlap and we can use the null scattering formulation to simplify unbiased ray marching. While fields outside of graphics have studied generalizations of the RTE [d'Eon, 2014], adapting these methods into a form suitable for image formation requires revisiting the foundations of the RTE.

Several recent papers have tackled this very problem and studied the impact of *correlated scattering* particles on the way light scatters in the volume [Bitterli et al., 2018, d'Eon, 2018, Jarabo et al., 2018]. For artists, this proposes a new level of control and flexibility when designing volumetric effects.

The main obstacles to widespread adoption in production include figuring out the proper way to deal with overlapping volumes, finding new optimal sampling techniques and parameterising the effect in a way that is both expressive and intuitive for artists. We anticipate production renderers to quickly adopt these methods if these challenges can be met.

5.4.4 Granular Media

Beyond the integration of correlated media mentioned above, there is the possibility of looking at other collections of objects as volumes. One such case involves densely packed grains such as sand or snow. Research on this topic [Meng et al., 2015, Moon et al., 2007, Müller et al., 2016] has found it is important to have a geometric component to the representation for fidelity in the low order scattering response (glints, and surface detail) while the higher order scattering can be accurately handled by switching to a volumetric representation.

The main obstacles to widespread adoption in production again center around integration and authoring issues for artists. For example, linking the work on granular media to the work on correlated media in a principled way and simplifying the authoring step (the papers mentioned above capture the bulk scattering behavior of small grains through precomputation which complicates the integration into production workflows).

5.4.5 Improve Boundary Sampling

In this course we have focused mostly on volumes without a boundary. When sampling sub-surface scattering effects, the presence of the boundary introduces a new wrinkle in the design of efficient sampling techniques. Many renderers choose to side-step the issue by either assuming a diffuse boundary (same assumption made by many BSSRDF models) or ignoring the boundary altogether when connecting shadow rays between the medium and outside lights. A more sophisticated solution is to use specialized sampling techniques such as Manifold Next Event Estimation [Hanika et al., 2015] to account for the refractive event. Because of the performance and look differences between these approaches, artists may be faced with difficult choices between performance and quality. Further research towards fast and accurate treatment of general boundaries would definitely be welcome.

References

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. A radiative transfer framework for non-exponential media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (nov 2018), 225:1–225:17. https://doi.org/10.1145/3272127.3275103

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Trans. Graph.* 37, 3, Article 33 (July 2018), 22 pages. https://doi.org/10.1145/3182159

Matt Jen-Yuan Chiang, Peter Kutz, and Brent Burley. 2016. Practical and Controllable Subsurface Scattering for Production Path Tracing. In *"ACM SIGGRAPH Talks"*. "ACM Press", Article 49, 2 pages. https://doi.org/10/gfzq7h

Per H. Christensen, Julian Fong, Jonathan Shade, Wayne L. Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Trans. Graph.* 37, 3 (2018), 30:1–30:21. https://dl.acm.org/citation.cfm?id=3182162

Alejandro Conty and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 25 (Aug. 2018), 17 pages. https://doi.org/10.1145/3233305

Eugene d'Eon. 2016. *A Hitchhiker's Guide to Multiple Scattering*. Self-published.

Eugene d'Eon. 2014. Computer graphics and particle transport: our common heritage, recent cross-field parallels and the future of our rendering equation. *Digipro* (2014).

Eugene d'Eon. 2018. A Reciprocal Formulation of Nonexponential Radiative Transfer. 1: Sketch and Motivation. *Journal of Computational and Theoretical Transport* 47, 1-3 (2018), 84–115. https://doi.org/10.1080/23324309.2018.1481433 arXiv:https://doi.org/10.1080/23324309.2018.1481433

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018a. Manuka: A batch-shading architecture for spectral path tracing in movie production. *Transactions on Graphics* 37, 3 (2018). https://doi.org/10.1145/3182161

Luca Fascione, Johannes Hanika, Rob Pieké, Ryusuke Villemin, Christophe Hery, Manuel Gamito, Luke Emrose, and André Mazzone. 2018b. Path Tracing in Production. In *SIGGRAPH 2018 Courses*. Article 15, 79 pages. https://doi.org/10.1145/3214834.3214864

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering: SIGGRAPH 2017 Course. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 2, 79 pages. https://doi.org/10.1145/3084873.3084907

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* 37, 3 (2018), 32:1–32:12. https://dl.acm.org/citation.cfm?id=3182160

Eric Haines and Tomas Akenine-Möller (Eds.). 2019. *Ray Tracing Gems*. Apress. http://raytracinggems.com.

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Comput. Graph. Forum* 34, 4 (July 2015), 87–97. https://doi.org/10.1111/cgf.12681

Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A Radiative Transfer Framework for Spatially-Correlated Materials. *ACM Transactions on Graphics* 37, 4 (2018).

Christopher Kulla, Alejandro Conty, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. *ACM Trans. Graph.* 37, 3, Article 29 (Aug. 2018), 18 pages. https://doi.org/10.1145/3180495

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. *"Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)"* 31, 4 (2012), 1519–1528. https://doi.org/10/f35f4k

Peter Kutz. 2013. The Importance of Ozone. http://skyrenderer.blogspot.com/2013/05/the-importance-of-ozone.html

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 111 (2017), 111:1–111:16 pages. https://doi.org/10.1145/3072959.3073665

Tom Lokovic and Eric Veach. 2000. Deep Shadow Maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385–392. https://doi.org/10.1145/344779.344958

Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-Scale Modeling and Rendering of Granular Materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 34, 4 (jul 2015). https://doi.org/10.1145/2766949

Bailey Miller, Iliyan Georgiev, and Wojciech Jarosz. 2019. A null-scattering path integral formulation of light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 38, 4 (jul 2019). https://doi.org/10.1145/3306346.3323025

Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. 2007. Rendering Discrete Random Media Using Precomputed Scattering Solutions. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 231–242. https://doi.org/10.2312/EGWR/EGSR07/231-242

R. Keith Morley, Solomon Boulos, Jared Johnson, David Edwards, Peter Shirley, Michael Ashikhmin, and Simon Premože. 2006. Image Synthesis Using Adjoint Photons. In *Proceedings of Graphics Interface 2006 (GI '06)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 179–186. http://dl.acm.org/citation.cfm?id=1143079.1143109

Ken Museth. 2013. VDB: High-resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (July 2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (dec 2016), 168:1–168:14. https://doi.org/10.1145/2980179.2982429

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2014)* 33, 6 (Nov. 2014), 179:1–179:11.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018a. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37, 2 (may 2018).

Jan Novák, Iliyan Georgiev, Johannes Hanika, Jaroslav Křivánek, and Wojciech Jarosz. 2018b. Monte Carlo Methods for Physically Based Volume Rendering. In *ACM SIGGRAPH Courses*. https://doi.org/10.1145/3214834.3214880

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3 ed.). "Morgan Kaufmann".

Charles M. Schmidt and Brian Budge. 2002. Simple Nested Dielectrics in Ray Traced Images. *J. Graphics, GPU, & Game Tools* 7 (2002), 1–8.

Florian Simon, Johannes Hanika, Tobias Zirr, and Carsten Dachsbacher. 2017. Line Integration for Rendering Heterogeneous Emissive Volumes. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 36, 4 (June 2017), 101–110.

H. C. van de Hulst. 1974. The spherical albedo of a planet covered with a homogeneous cloud layer. *Astronomy and Astrophysics* 35 (Oct. 1974), 209–214.

Ryusuke Villemin and Christophe Hery. 2013. Practical Illumination from Flames. *"Journal of Computer Graphics Techniques (JCGT)"* 2, 2 (2013), 142–155.

Magnus Wrenninge. 2009. Field3D. https://github.com/imageworks/Field3D

Magnus Wrenninge. 2016. Efficient Rendering of Volumetric Motion Blur Using Temporally Unstructured Volumes. *Journal of Computer Graphics Techniques (JCGT)* 5, 1 (31 January 2016), 1–34. http://jcgt.org/published/0005/01/01/

Magnus Wrenninge, Christopher D. Kulla, and Viktor Lundqvist. 2013. Oz: the great and volumetric. In *International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2013, Anaheim, CA, USA, July 21-25, 2013, Talks Proceedings*. 46:1. https://doi.org/10.1145/2504459.2504518

Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. 2014. High-order Similarity Relations in Radiative Transfer. *ACM Trans. Graph.* 33, 4, Article 104 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601104

6 The Ins of Production Rendering at Animal Logic

Daniel Heckenberg, *Animal Logic*

As path tracing has become the standard paradigm for production rendering, representation of scene data has changed. After the long dominance of the *RenderMan Interface* specification or Ri, newer scene descriptions achieve different tradeoffs for computation efficiency, workflow, complexity management and expressivity. We'll discuss Animal Logic's progression from *RenderMan Interface*, to *Glimpse*'s Scene Stream GSS and Pixar's *Universal Scene Description* USD [Pixar, 2019]. Our emphasis has been on achieving interactive rendering throughout the production pipeline.

6.1 The RenderMan Interface

The *RenderMan Interface* was introduced by Pixar in 1988 and its legacy can be seen in every modern production renderer and scene description. It allows scene geometry and imaging to be specified through a now familiar hierarchical state machine model, similar to OpenGL. It evolved continuously and became a common interface for many production renderers. Innovations included shader-defined behaviour for many aspects of the rendering process using the *RenderMan Shading Language* RSL, the *RenderMan Interface Bytestream* RIB for serialization, high-level structure through referencing and procedurals and deferred scene loading based on visibility.

6.1.1 MayaMan at Animal Logic

The Ri allows a rich description of a scene for a single image but does not represent a full animated sequence. Nor does it provide means to read back and edit its data. This makes it essentially an output format to be produced by translation from other sources. Animal Logic developed, and for some time sold commercially, a translator called *MayaMan* to push data from Autodesk *Maya* to Ri. Many other products work in a similar way: taking data represented in an editable and animated form usually in a Digital Content Creation DCC application and generating the time-slice of data in a renderer's input format to produce a single image.

A variety of approaches can be used with Ri to increase efficiency and pipeline compatibility of rendering at production scale, including:

- proxy objects in DCC to avoid duplicate representations
- geometry caches and procedurals with direct Ri output
- streaming of scene data using deferred loading
- late scene manipulation using Ri Filter plugins

Typically our translation process looked like:

1. load *Maya* scene containing proxies and geometry caches (like *Alembic*)
2. import to native *Maya* only those scene elements required for editing
3. translate to multiple passes of RIB, with deferred procedural references for proxies and geometry caches to avoid translation

This produces completely standalone RIB descriptions of the scene which can be loaded on another machine, or in a separate process without necessarily keeping the DCC in memory at the same time.

In a rasterising rendering scheme as we were using (Pixar's *RenderMan* in REYES mode) this has the following characteristics:

- many passes may be required to be rendered in the correct order (e.g. shadow maps)
- duplicate storage of scene objects in multiple formats is minimal
- the working set for the scene geometry may be much smaller than the total amount of data due to streaming load / discard

6.2 Glimpse and GSS

Glimpse was born out of the lack of interactivity in our *MayaMan* processes. Interactivity was stymied by the complex pipeline of translation and pass dependencies, iteration through this pipeline without effective short-cuts for incremental edits and a systematic lack of priority on fast feedback (such as time to first bucket / sample iteration) versus overall throughput. A more complete history of *Glimpse* may be found in a previous iteration of this course [Fascione et al., 2017].

Path tracing offers the crucial foundation of a single-pass rendering process but the rest of the rendering pipeline also needed to be updated to achieve practical interactivity. Path tracing also imposes the constraint of having the whole (visible) scene in memory which is conveniently well-aligned with interactive rendering. *Glimpse* and its scene description, GSS for *Glimpse Scene Stream*, were developed originally as a complementary system for interactive rendering set up but eventually replaced our other rendering approaches.

Our *Glimpse* model is:

- GSS API is designed for in-memory editing and updates
- DCC native objects may be bridged to GSS objects
- bridged objects are tracked to propagate fine-grained incremental edits
- proxy objects in DCC to avoid duplicate representations
- exploit GSS representation and *Glimpse* render for viewers and editors

6.2.1 Memory

Memory overhead is a challenge as data may pass through the following forms, and updates may come from various stages in different interactive workflows:

1. serialised representation (e.g. *Alembic* on disk)
2. geometry cache representation (e.g. *Alembic* in RAM)
3. DCC representation (e.g. *Maya*'s MFnMesh in RAM)
4. GSS representation
5. render representation (e.g. subdivided mesh)
6. acceleration structures (e.g. *Glimpse* BVH)

Glimpse uses the following principles to balance memory overhead and interactivity through minimal updates:

1. Use GSS representation wherever possible. This form must be rich enough for general scene description with support for transform hierarchies, referencing etc.
2. one-to-one bridge objects when using native DCC editing to minimise complexity of state tracking and updates
3. Discard intermediate data but preserve structure. e.g. the GSS representation of mesh vertex data can be transient in many contexts

6.2.2 Latency

Low latency for start up and edits is also necessary for interactivity. The best and obvious approach is to minimise and optimise the work required on scene objects prior to rendering. Once again this motivated us to make GSS as rich as possible to be used directly as a scene source. When bridging to native DCC objects the same approaches that are used to minimize memory and aid performance. Typically we create one-to-one peer objects for direct change tracking and incremental dependent updates.

Multithreading is possible in many parts of the GSS API and is facilitated by avoiding the stack-based paradigm of Ri. This is analogous to bindless *OpenGL* interfaces. Opportunities for parallelism are exposed through scene structure (objects, references and procedurals) at a high level.

6.2.3 Instancing

Path tracing renderers can exploit instancing to achieve visual complexity with minimal scene load time and memory. For scene description, the challenge of instancing is to present the feature in a way that is easy to exploit but not fragile. At one extreme, instancing is a late-stage optimisation like de-duplication that doesn't prohibit any earlier operations or edits on the scene. At the other extreme, instances must be directly expressed in the scene and require explicit de-instancing for edits.

Instancing in the scene description domain may have different constraints compared to any particular renderer:

- path tracers typically support per-instance specialisation of some subset of visibility, attributes, user data and material bindings
- objects may be identical except for render-time operations such as subdivision and displacement
- some renderers may not support these render-time operations at all (e.g. *OpenGL* preview)
- describing per-instance specialisations is challenging in a hierarchical scene description (e.g. nested diamond instancing) where all nodes are shared by multiple instances
- nested instances are powerful for compact (therefore small, fast) descriptions of repeated structures like buildings and trees. It may be beneficial even if a renderer doesn't support multilevel instancing

Glimpse uses hierarchical state and path-based instance overrides to allow instances to be addressed.

6.2.4 Expressiveness

A more subtle aspect of interactivity relates to the expressiveness of scene description mechanisms. The more directly and concisely an outcome can be expressed in a target scene description, the fewer other layers or operations are required for the user and the system as a whole. This must of course be balanced with the complexity and frequency of the operation: one that is rarely applied or is slow may be better performed in another system. In GSS, material assignment is a good example of this tension: assignments are hierarchical and layered but we don't currently support wildcard assignments due to performance. If required, wildcard assignments may be represented and mapped to explicit assignment by a higher level system.

We've found that the mechanisms in GSS described for instancing usually work well for concise, expressive scene manipulation. Hierarchically inherited state allows for the scope of manipulation to be chosen, within the limits of the existing scene structure. Instance overrides allow for a large category of instancing-compatible changes to be made to particular instances independent of scene structure.

However, what works well for interactive control for a particular scene may not be effective for reuse in other scenes or robust to changes in scene content. These are the concerns of production lighting and rendering teams and may be seen in the difference between direct manipulation of rendering state in *Maya* versus the procedural approach of *Katana*.

6.3 USD and Glimpse

At Animal Logic, we have replaced most uses of GSS with Pixar's USD. USD matches or exceeds the capabilities of GSS in almost all areas and has been enthusiastically embraced by production facilities and commercial vendors in the industry.

Important aspects of USD are:

- full time sequence description (not just a single frame)
- editable to allow interactivity without further duplication
- state tracking for dependent updates
- high performance for scene load and processing

Our transition approach has been to create a USD front end for *Glimpse*. The *Hydra* rendering system of USD is appealing but at the time of writing, this interface is not feature complete for our needs.

There are a number of areas where USD does not yet match our previous functionality directly, but is flexible enough to allow us to take hybrid approaches:

- material binding: *Glimpse* supports layered materials described with a custom USD schema
- procedural geometry: references to our renderer's procedurals are embedded with a custom USD schema

For editing and manipulation, we have replaced the use of our *Glimpse* to *Maya* bridge with the plugin *AL_USDMaya* [Animal Logic, 2019]. *AL_USDMaya* bridges USD into *Maya* in similar but more extensive ways than our previous approach with GSS. This offers direct manipulation of USD in *Maya* with interactive rendering through *Glimpse*.

Acknowledgements

The systems described here were developed by many at Animal Logic (past and present) especially the *Glimpse* and *AL_USDMaya* teams and open-source contributors. This work also leans heavily on Pixar's *Universal Scene Description*.

References

Animal Logic. 2019. AL_USDMaya Github Repository. https://github.com/AnimalLogic/AL_USDMaya.

Luca Fascione, Johannes Hanika, Rob Pieké, Christophe Hery, Ryusuke Villemin, Thorsten-Walther Schmidt, Christopher Kulla, Daniel Heckenberg, and André Mazzone. 2017. Path Tracing in Production - Part 2: Making Movies. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. Article 15, 32 pages. https://doi.org/10.1145/3084873.3084906

Pixar. 2019. Universal Scene Description. https://graphics.pixar.com/usd/docs/index.html.