Supplemental Material II: Integration algorithms

Zdravko Velinov, and Kenny Mitchell

Abstract—The main text compares many algorithms that are well known in the graphics literature. This supplemental material provides the algorithms used for ray marching, Monte Carlo integration with distance sampling and different variants of ratio estimators. Additionally, it provides algorithms for PDF and sampling that can be used in traditional path tracing frameworks.

Index Terms—raytracing, color, shading, shadowing, texture.

1 Introduction

The aim of this document is to give additional details regarding implementation of well-known algorithm that we compared against in the main text. Ray marching and distance sampling are the two algorithms implemented for comparison purposes. Additionally, ratio estimator is further discussed in ray marching and distance sampling scenarios. Those algorithms combine known techniques with our analytic unoccluded integral computation.

Finally, in path tracing frameworks it is crucial to implement both functions for generating new samples and probability density functions, required to combine different sampling strategies. Those are derived for a given starting point in space and a set of incident and outgoing light vectors.

2 Single-scattering integration using ray marching

Algorithm 1 Single-scattering integrator using ray marching algorithm

```
procedure L(ω̃_i, ω̃_o, t_S)
    t_{C,B} ← min(t_S, t_{C,E})
    𝒯_total ← 0
    for k ← 1 to N_sample do
        t ← ξ(k/N_sample) t_{C,B}
        p̃ ← p̃_o − t ω̃_o
        d ← IntersectBox(p̃, ω̃_i)
        𝒯_k ← V(p̃) e^{−σ_t (t+d)}
        𝒯_total ← 𝒯_total + 𝒯_k
    end for
    L ← (t_{C,B} / N_sample) σ_s f(ω_i, ω_o) 𝒯_total L_i
    return L
end procedure
```

Ray marching spreads the samples equidistantly along the camera ray ($t = \xi(k/N_{\text{sample}})\, t_{C,B}$). The main details of the algorithm are shown in Alg. 1. Samples are generated within the range of the end of the ray $t_{C,E}$ and distance to the closest surface visible along the camera ray t_S. At the start of the loop the integral over transmittance is initialized to zero ($\mathscr{T}_{\text{total}} = 0$). The integrator is executed for a finite number of samples N_{sample}. Samples are generated by drawing from uniform distribution in the range $[0,1]$ and spreading them according to the distance to the constrained end of the ray ($t = \xi(k/N_{\text{sample}})\, t_{C,B}$). The position is computed by multiplying the generated distance by the view vector ($-\tilde{\omega}_o$) and offsetting the camera origin $\tilde{\mathbf{p}}_o$ ($\tilde{\mathbf{p}} = \tilde{\mathbf{p}}_o - t\,\tilde{\omega}_o$). The exit point from the position following the incident light ray $\tilde{\omega}_i$ is computed using a slab intersection test as outlined in the main text ($\text{IntersectBox}(\tilde{\mathbf{p}}, \tilde{\omega}_i)$). The transmittance of each sample is weighted by the visibility ($\mathscr{T}_k = V(\tilde{\mathbf{p}})\, e^{-\sigma_t (t+d)}$). Each iteration ends with accumulation of the integral over transmittance ($\mathscr{T}_{\text{total}} \leftarrow \mathscr{T}_{\text{total}} + \mathscr{T}_k$). In this integrator the probability of selecting a segment is one over the total distance which results in the first weight ($t_{C,B}/N_{\text{sample}}$). The second weight is the phase function ($f(\omega_i, \omega_o)$) multiplied by the scattering coefficient σ_s. The final result is assembled by multiplying by the uniform incident light radiance L_i ($L = (t_{C,B}/N_{\text{sample}}) f(\omega_i, \omega_o)\, \mathscr{T}_{\text{total}}\, L_i$).

3 Multiple-scattering integration using ray marching

In multiple-scattering scenarios, the algorithm must accumulate the throughput along multiple path segments and connect path segments to the light source and surrounding medium. The throughput at the end of each path segment is characterized by the transmittance, phase function and scattering coefficient,

$$L_{\text{path}} = \sigma_s\, f(\omega_i, \omega_o)\, \exp(-\sigma_t\, t)\, L_{\text{next}}. \tag{1}$$

The next step is to apply the PDF to avoid bias when estimating the final value. The PDFs of both the distance (ray marching) and phase sampling functions must be applied separately as two distinct uncorrelated events to derive an unbiased estimate,

$$\begin{aligned} L'_{\text{path}} &= \frac{L_{\text{path}}}{\text{PDF}_{\text{distance}}(\mathbf{p}, \omega_o)\, \text{PDF}_{\text{phase}}(\omega_i, \omega_o)} \\ &= \frac{\sigma_s\, f(\omega_i, \omega_o)\, \exp(-\sigma_t\, t)}{\frac{1}{t_{C,B}} f(\omega_i, \omega_o)}\, L_{\text{next}} \\ &= t_{C,B}\, \sigma_s\, \exp(-\sigma_t\, t)\, L_{\text{next}} \end{aligned} \tag{2}$$

- *Zdravko Velinov was with Disney Research Los Angeles, Glendale, CA. ORCiD: 0000-0002-1258-1223*
- *Kenny Mitchell was with Disney Research Los Angeles, Edinburgh Napier University and Roblox Corporation, San Mateo, CA. ORCiD: 0000-0003-2420-7447*

Manuscript received December 14, 2020; revised December, 2021.

Algorithm 2 Single-scattering integrator using *distance sampling*

procedure $L(\tilde{\omega}_i, \tilde{\omega}_o, t_S)$
  $t_{C,B} \leftarrow \min(t_S, t_{C,E})$
  $\mathscr{T}_{\text{total}} \leftarrow 0$
  for $k \leftarrow 1$ to N_{sample} **do**
    $r \leftarrow \xi(k/N_{\text{sample}})$
    $t \leftarrow -\frac{1}{\sigma_t}\log\left(1-(1-e^{-\sigma_t t_{C,B}})\, r\right)$
    $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}}_o - t\,\tilde{\omega}_o$
    $d \leftarrow \text{IntersectBox}\left(\tilde{\mathbf{p}}, \tilde{\omega}_i\right)$
    $\mathscr{T}_k \leftarrow V(\tilde{\mathbf{p}})\, e^{-\sigma_t d}$
    $\mathscr{T}_{\text{total}} \leftarrow \mathscr{T}_{\text{total}} + \mathscr{T}_k$
  end for
  $L \leftarrow \frac{1-e^{-\sigma_t t_{C,B}}}{N_{\text{sample}}}\,\sigma_s\, f(\omega_i,\omega_o)\,\frac{\mathscr{T}_{\text{total}}}{\sigma_t}\, L_i$
  return L
end procedure

4 Single-scattering integration using distance sampling

Distance sampling is derived to generate samples only within the medium by normalizing against the highest value of the integral over transmittance along the camera ray segment,

$$
\begin{aligned}
r &= \frac{\int_0^t e^{-\sigma_t t}\mathrm{d}t}{\int_0^{t_{C,B}} e^{-\sigma_t t}\mathrm{d}t} \\
&= \frac{\frac{1}{\sigma_t}(1-e^{-\sigma_t t})}{\frac{1}{\sigma_t}(1-e^{-\sigma_t t_{C,B}})} \\
&= \frac{1-e^{-\sigma_t t}}{1-e^{-\sigma_t t_{C,B}}}.
\end{aligned} \tag{3}
$$

Inversion is performed in a similar way as our importance sampling strategy, but it ends up not having components dependent on the distance from the camera ray to the medium boundary,

$$
\begin{aligned}
1-e^{-\sigma_t t} &= \left(1-e^{-\sigma_t t_{C,B}}\right) r \\
e^{-\sigma_t t} &= 1-\left(1-e^{-\sigma_t t_{C,B}}\right) r \\
t &= -\frac{1}{\sigma_t}\log\left(1-\left(1-e^{-\sigma_t t_{C,B}}\right) r\right).
\end{aligned} \tag{4}
$$

This is a well known result that can be found in books and courses regarding volume rendering [1], [2]. However, it can be easily shown that distance sampling can be derived as a subset of our sampling algorithm when the distance to the edge is constant ($d_{\text{E,S}} = d_{\text{E,E}}$) and the starting distance is 0 ($t_{C,S} = 0$),

$$
\begin{aligned}
t &= t_{C,S} - \frac{1}{\sigma_t c}\log(1-(1-E_u)\, r) \\
t &= t_{C,S} - \frac{1}{\sigma_t\left(1+\frac{\cancel{d_{\text{E,E}}-d_{\text{E,S}}}}{t_{C,E}-t_{C,S}}\right)}. \\
&\log\left(1-\left(1-e^{-\sigma_t\left(1+\frac{\cancel{d_{\text{E,E}}-d_{\text{E,S}}}}{t_{C,E}-t_{C,S}}\right)(t_{C,B}-t_{C,S})}\right) r\right) \\
t &= \cancel{t_{C,S}} - \frac{1}{\sigma_t}\log\left(1-\left(1-e^{-\sigma_t(t_{C,B}-\cancel{t_{C,S}})}\right) r\right) \\
t &= -\frac{1}{\sigma_t}\log\left(1-\left(1-e^{-\sigma_t t_{C,B}}\right) r\right)
\end{aligned}
$$

The integration algorithm incorporating distance sampling is shown in Alg. 2. It partially cancels the transmittance term ($\mathscr{T}_k = V(\tilde{\mathbf{p}})\, e^{-\sigma_t d}$) and results in a constant corrective term proportional to the integral over transmittance along the camera ray line segments ($1-e^{-\sigma_t t_{C,B}}$). Everything else is similar to the ray marching algorithm.

5 Multiple-scattering integration using Monte Carlo with distance sampling

Similarly to multiple-scattering with ray marching the individual PDFs must be applied to compute an unbiased estimate when combining contribution for each path segment,

$$
\begin{aligned}
L'_{\text{path}} &= \frac{L_{\text{path}}}{\text{PDF}_{\text{distance}}(\mathbf{p},\omega_o)\,\text{PDF}_{\text{phase}}(\omega_i,\omega_o)} \\
&= \frac{\sigma_s f(\omega_i,\omega_o)\, e^{-\sigma_t t}}{\sigma_t \frac{e^{-\sigma_t t}}{1-e^{-\sigma_t t_{C,B}}} f(\omega_i,\omega_o)} L_{\text{next}} \\
&= \frac{\sigma_s}{\sigma_t}(1-e^{-\sigma_t t_{C,B}})\, L_{\text{next}}
\end{aligned} \tag{5}
$$

6 Single-scattering integration using ratio estimator with equidistant Sampling

Algorithm 3 Ratio estimator with equidistant sampling single-scattering integration algorithm

procedure $L(\tilde{\omega}_i, \tilde{\omega}_o, t_S)$
  $t_{C,B} \leftarrow \min(t_S, t_{C,E})$
  $\mathscr{T}_{\text{total}} \leftarrow 0$
  $\mathscr{T}_{\text{u}} \leftarrow 0$
  for $k \leftarrow 1$ to N_{sample} **do**
    $t \leftarrow \xi(k/N_{\text{sample}})\, t_{C,B}$
    $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}}_o - t\,\tilde{\omega}_o$
    $d \leftarrow \text{IntersectBox}\left(\tilde{\mathbf{p}}, \tilde{\omega}_i\right)$
    $\tau \leftarrow e^{-\sigma_t (t+d)}$
    $\mathscr{T}_k \leftarrow V(\tilde{\mathbf{p}})\,\tau$
    $\mathscr{T}_{\text{total}} \leftarrow \mathscr{T}_{\text{total}} + \mathscr{T}_k$
    $\mathscr{T}_{\text{u}} \leftarrow \mathscr{T}_{\text{u}} + \tau$
  end for
  $L \leftarrow \sigma_s\, f(\omega_i,\omega_o)\,\mathscr{T}_{\text{box}}(\tilde{\omega}_i,\tilde{\omega}_o,t_S)\,\frac{\mathscr{T}_{\text{total}}}{\mathscr{T}_{\text{u}}}\, L_i$
  return L
end procedure

The main difference compared to previous algorithms is that it accumulates both an occluded $\mathscr{T}_{\text{total}}$ and unoccluded term $\mathscr{T}_{\text{u}}$ (Alg. 3) and at the end of the computation, it multiplies the ratio term by our analytic unoccluded radiance computation algorithm ($\mathscr{T}_{\text{box}}(\tilde{\omega}_i,\tilde{\omega}_o,t_S)\, L_i$)). In the limit, the unoccluded term will approach the analytically computed result ($\mathscr{T}_{\text{u}}\, t_{C,B}/N_{\text{sample}} \rightarrow \mathscr{T}_{\text{box}}(\tilde{\omega}_i,\tilde{\omega}_o,t_S)\, L_i)$) and completely cancels it out, leading to asymptotically converging to unbiased solution single-scattering estimate. The concept extends to multi-scattering integrators with Russian Roulette.

7 Single-scattering integration using ratio estimator with distance Sampling

The main premise of this ratio estimator algorithm (Alg. 4) is the same as the previously explained ratio estimator algorithm (Alg. 3). However, it generates samples according to a *distance sampling* function.

Algorithm 4 Single-scattering ratio estimator using *distance sampling*

procedure $L(\tilde{\omega}_i, \tilde{\omega}_o, t_S)$
$t_{C,B} \leftarrow \min(t_S, t_{C,E})$
$\mathscr{T}_{\text{total}} \leftarrow 0$
$\mathscr{T}_{\text{u}} \leftarrow 0$
for $k \leftarrow 1$ to N_{sample} **do**
$r \leftarrow \xi(k/N_{\text{sample}})$
$t \leftarrow -\frac{1}{\sigma_t}\log\left(1-\left(1-e^{-\sigma_t t_{C,B}}\right) r\right)$
$\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}}_o - t\,\tilde{\omega}_o$
$d \leftarrow \text{IntersectBox}\left(\tilde{\mathbf{p}}, \tilde{\omega}_i\right)$
$\tau \leftarrow e^{-\sigma_t d}$
$\mathscr{T}_k \leftarrow V(\tilde{\mathbf{p}})\,\tau$
$\mathscr{T}_{\text{total}} \leftarrow \mathscr{T}_{\text{total}} + \mathscr{T}_k$
$\mathscr{T}_{\text{u}} \leftarrow \mathscr{T}_{\text{u}} + \tau$
end for
$L \leftarrow \sigma_s f(\omega_i, \omega_o)\,\mathscr{T}_{\text{box}}(\tilde{\omega}_i, \tilde{\omega}_o, t_S)\,\frac{\mathscr{T}_{\text{total}}}{\mathscr{T}_{\text{u}}}\,L_i$
return L
end procedure

8 Multiple-scattering using whole volume distance sampling

Following the general principle of applying the PDF of the distance and phase sampling strategies, the contribution for each segment can be derived,

$$\begin{aligned} L'_{\text{path}} &= \frac{L_{\text{path}}}{\text{PDF}_{\text{distance}}(\mathbf{p}, \omega_o)\,\text{PDF}_{\text{phase}}(\omega_i, \omega_o)} \\ &= \frac{\sigma_s f(\omega_i, \omega_o)\, e^{-\sigma_t t}}{\frac{e^{-\sigma_t (t+d)}}{\mathscr{T}_{\text{total}}} f(\omega_i, \omega_o)} L_{\text{next}} \\ &= \sigma_s \frac{\mathscr{T}_{\text{total}}}{e^{-\sigma_t d}} L_{\text{next}} \end{aligned} \tag{6}$$

9 Alternative sampling algorithms

Algorithm 5 Algorithm for sampling proportional to unoccluded radiance in a box section illuminated uniformly by a light source

procedure $S_{\text{hist}}(\tilde{\omega}_i, \tilde{\omega}_o, t_S, r)$
$\begin{pmatrix}\mathscr{T}_{\text{total}} & \mathscr{T} & \mathbf{K}\end{pmatrix} = \mathscr{T}_{\text{box}}(\tilde{\omega}_i, \tilde{\omega}_o, t_S)$
$H_0 \leftarrow 0$
for $k \leftarrow 1$ to 3 **do**
$G_k \leftarrow \mathscr{T}_k / \mathscr{T}_{\text{total}}$
$H_k \leftarrow \sum_{i=1}^{k} G_i$
$D_k \leftarrow \begin{cases} 1/G_k & G_k > 0 \\ 1 & \text{otherwise} \end{cases}$
$M_k \leftarrow -H_{k-1} D_k$
end for
$r \leftarrow \xi(k/N_{\text{sample}})$
$s \leftarrow \text{step}(H_1, r) + \text{step}(H_2, r)$
$r_s \leftarrow r D_s + M_s$
$t \leftarrow \begin{cases} -\frac{\log(1-\mathbf{K}_{s,1} r)}{\mathbf{K}_{s,2}} & |\mathbf{K}_{s,2}| > \varepsilon \wedge \mathbf{K}_{s,3} > \varepsilon \\ \mathbf{K}_{s,3} r & \text{otherwise} \end{cases}$
$t \leftarrow t + \mathbf{K}_{s,4}$
return $\begin{pmatrix} t & \mathscr{T}_{\text{total}} \end{pmatrix}$
end procedure

Traditional path tracing renderers require a separate sampling and PDF function. The sampling function is derived similarly to the final optimized Monte Carlo algorithm, but with a single sample taken by the function (Alg. 5). The sampling function provides both the distance and pre-multiplied transmittance by the PDF ($\mathscr{T}_{\text{total}}$) which is a common way of optimizing the integration framework.

Algorithm 6 Computing PDF of a sample proportional to unoccluded radiance in a box section illuminated uniformly by a light source

procedure $\text{PDF}(\tilde{\omega}_i, \tilde{\omega}_o, t_S, t)$
$d_{\text{E,S}} \leftarrow d_L$
$t_{C,\text{S}} \leftarrow 0$
$\tilde{\mathbf{p}}_{\text{edge}} \leftarrow \tilde{\mathbf{p}}_L$
for $k \leftarrow 1$ to 3 **do**
$\begin{pmatrix}\tilde{\mathbf{p}}_k & \tilde{\mathbf{w}}_k\end{pmatrix} \leftarrow \text{ComputeCorner}(\tilde{\mathbf{p}}_{\text{prev}})$
$\begin{pmatrix}d_{\text{E,E}} & t_{C,E}\end{pmatrix} \leftarrow \text{DistanceToCorner}(\tilde{\mathbf{p}}_k, \tilde{\mathbf{w}}_k)$
$\begin{pmatrix}\mathscr{T}_k & \mathbf{K}_k\end{pmatrix} \leftarrow \mathscr{T}_{\text{trapezoid}}(t_{C,S}, t_{C,E}, d_{\text{E,S}}, d_{\text{E,E}})$
$t_{C,\text{S}} \leftarrow t_{C,\text{E}}$
if $t_{C,S} \leq t \leq t_{C,E}$ **then**
$\tau \leftarrow e^{-\sigma_t \left(t + d_{\text{E,S}} + (t - t_{C,S})\frac{d_{\text{E,E}} - d_{\text{E,S}}}{t_{C,E} - t_{C,S}}\right)}$
end if
$d_{\text{E,S}} \leftarrow d_{\text{E,E}}$
end for
$\mathscr{T}_{\text{total}} \leftarrow \sum_{k=1}^{3} \mathscr{T}_k$
return $\tau / \mathscr{T}_{\text{total}}$
end procedure

The PDF (Alg. 6) is computed as the transmittance τ at a given distance t divided by the integral over the unoccluded transmittance in the box section $\mathscr{T}_{\text{total}}$. The algorithm executes a loop over all segments until the distance falls within a given segment ($t_{C,S} \leq t \leq t_{C,E}$). At that segment the transmittance is evaluated following Beer's law,

$$\tau = e^{-\sigma_t \left(t + d_{\text{E,S}} + (t - t_{C,S})\frac{d_{\text{E,E}} - d_{\text{E,S}}}{t_{C,E} - t_{C,S}}\right)} \tag{7}$$

Everything else is computed as explained in the main text [3].

10 Sampling multiple light sources

Shirley and Wang [4] express the integral over radiance contributed by all light sources in a Monte Carlo sense as a sum of individual terms weighted by their respective PDF,

$$L = \sum_{i=0}^{N_{\text{light}}} L_i \approx \sum_{i=0}^{N_{\text{light}}} \frac{L_i(X)}{\text{PDF}_i(X)}. \tag{8}$$

They further outline that each PDF is actually a product of the PDF of selecting each light source and given light sample. Their suggestion is that in the simplest case, it can be as simple as selecting one source at random from the set, which corresponds to a probability density function $1/N_{\text{light}}$, however, they instead advice to sample according to the light source size, intensity, distance or any other possible parameters that might affect the computation. However, in our case, we can sample proportional to the unoccluded radiance which is a more concrete example of a good estimator which covers participating media which is a step over what is shown in the original paper. We support that claim with example in our main text [3]. The weights are expressed as the integral over transmittance of each light source divided by the total transmittance over all segments,

$$\text{PDF}_i(X) = \frac{\mathscr{T}_{\text{box}\,i}}{\sum_{i=0}^{N_{\text{light}}} \mathscr{T}_{\text{box}\,i}}. \tag{9}$$

As it is expected, if the contribution is equal for all light sources, it reduces to $1/N_{\text{light}}$, however in most cases that won't be true and sampling will be proportional to the expected maximum contribution of each light source, resulting in variance reduction.

11 Extending past rectilinear boxes

Since constructing most of the integral solutions requires projection operations and each segment is relatively independent from its neighbors. It is possible to construct the integral equation by directly intersecting the medium, however, more care will be required to keep the computation numerically stable in corner cases. Alternatively, Kuijper *et al.* [5] proposed an algorithm which is guaranteed to terminate on convex shapes. Intersecting triangles can be performed individually or against a bounding volume hierarchy. The intersection between the light-view plane and a bounding box can be expressed by constraining it between the starting and ending light plane and checking whether the box lies on both sides of the light-view plane. Then the bounding box is definitely intersected by the plane and its content can potentially contribute to the final radiance. Refer to Alg. 7 for more details. The bounding box is defined by its three axes $\mathbf{a}_x$, $\mathbf{a}_y$ and $\mathbf{a}_z$. Operations follow as already explained. Having a triangle selected for intersection. the objective is to find the projected segment of the camera ray on the triangle that falls within the triangle. First the triangle edge vectors are computed $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$. Then the triangle normal $\mathbf{n}$ and distance vector of the projected line segment is computed $\mathbf{d}$, which allow to project the beginning and end of the camera ray on the triangle plane. The next step is to truncate the projection to the triangle or discard the segment altogether if it is outside of the triangle. The basic procedure involves first constructing each normal vector perpendicular to the triangle plane and pointing inside the space enclosed by the triangle and perpendicular to each edge. We will call them *edge plane vectors*. Then we project the offset between the vertex of each triangle edge on that normal and the direction between the starting and end point of the segment. If the sign of the projection of the direction between the two points and the edge plane vector is negative and they need to move in positive direction to enter the triangle, then movement will make the line segment longer, and we consider that case as a failure to contain the projected line segment within the triangle. We terminate the process in that case. Otherwise, we advance the point by the ratio of the projected offset and the projection of the direction vector. We truncate it, so that they don't result in increasing the line. We repeat the same procedure with the inverse direction between the ends of the line segments. After successfully completing the procedure for all edges, we then truncate the point to not overshoot the length of the line segment, thus producing the two ends of the projected line segment $\mathbf{s}'_0$ and $\mathbf{s}'_1$. We can now project back on the view vector and construct a *trapezoid segment*. The basic step involve projecting to find the new beginning and end distances along the view vector $t_{C,S}$ and $t_{C,E}$. From them, the distance to the projected line segment can be found by projecting their corresponding points on the view vector and then on the incident light vector. The computation afterwards continues in the same manner as the box using the algorithms for computing the integral over transmittance enclosed in a trapezoid section $\mathscr{T}_{\text{trapezoid}}$ and sampling according to our algorithm S_{s}.

Algorithm 7 Intersection between the light-view plane and Oriented Bounding Box

```
procedure LightViewOBBIntersectionTest(ω_i, ω_o, p_o, t_S, t_C)
    t_OBB ← |a_x · ñ_l| + |a_y · ñ_l| + |a_y · ñ_l|
    t_org ← t_OBB − p_o · ñ_l
    if t_org < 0 then
        return False
    end if
    t_end ← (p_o − min(t_C, t_S) · ω_o) · ñ_l + t_OBB
    if t_end > 0 then
        return False
    end if
    t_org,lv ← p_o · n_lv
    t_OBB,lv ← |a_x · n_lv| + |a_y · n_lv| + |a_y · n_lv|
    if t_org,lv − t_OBB,lv < 0 ∧ t_org,lv + t_OBB,lv ≥ 0 then
        return True
    end if
    return False
end procedure
```

Algorithm 8 Intersection between the light-view plane and Oriented Bounding Box

```
procedure ComputeTrapezoidTriangleRay(ω_i, ω_o, p_o, t_S)
    e_0 ← v_1 − v_0
    e_1 ← v_2 − v_1
    e_2 ← v_0 − v_2
    d ← −min(t_C, t_S) ω_o
    n ← e_2 × e_0
    s_0 ← p_o + ω_o ((v_0 − p_o)·n) / (ω_o·n)
    s_1 = p_o + d + ω_o ((v_0 − p_o − d)·n) / (ω_o·n)
    for i ← 1 to 3 do
        τ_i ← n × e_i
        h ← d · τ_i
        g ← (s_0 − v_i) · τ_i
        if h < ε ∧ g > ε then
            return False
        end if
        s'_0 ← s_0 + max(0, max(0, g)/h) d
        h ← −h
        g ← (s_1 − v_i) · n
        if h < ε ∧ g > ε then
            return False
        end if
        s'_1 ← s_1 + max(0, max(0, g)/h) d
    end for
    s'_0 ← s_0 + d min(1, (s'_0 − s_0) · d/|d|^2)
    s'_1 ← s_1 + d min(1, (s'_1 − s_1) · d/|d|^2)
    t_C,S ← ((p_o − s'_0)·ñ_l) / (ω_o·ñ_l)
    t_C,E ← ((p_o − s'_1)·ñ_l) / (ω_o·ñ_l)
    d_E,S ← (s'_0 − (p_o − ω_o t_C,S)) · ω_i
    d_E,E ← (s'_1 − (p_o − ω_o t_C,E)) · ω_i
    return (True  t_C,S  t_C,E  d_E,S  d_E,E)
end procedure
```

12 Avoiding unbounded memory requirements

One possible issue is that memory requirements can grow potentially unbounded. In practice, it is completely avoidable as long as there is access to a pseudorandom number generator. Given a set of variables that we want to sample proportional to their total contribution, we can take two variables at each step - one from a previous step and another from the current step. If we compare the variable in the current step against the total accumulated contribution so far and it is below that threshold, we can replace the variable from the previous step, otherwise, we keep the variable from the previous step. We can thus express this relation as inequality

$$\begin{aligned} P(C_n) &< \xi(x) \\ \frac{C_n}{\sum_{i=1}^{n} C_i} &< \xi(x), \end{aligned} \tag{10}$$

where we outline the contribution at the current step as C_i. The relationship above works because for the variables from a previous step we have

$$P_n(C_k|k < n) = \frac{C_k}{\sum_{g=1}^{k} C_g} \prod_{j=k+1}^{n} \frac{\sum_{m=1}^{j-1} C_j}{\sum_{h=1}^{j} C_j}. \tag{11}$$

Basically, it cancels the denominator at each step from the previous one. Note, that this probability approaches 1 which will result in worse floating-point precision. Therefore, the probability of the current variable is to be preferred. The same framework can be used both for sampling segments and for selecting luminaires.

REFERENCES

[1] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[2] J. Fong, M. Wrenninge, C. Kulla, and R. Habel, “Production volume rendering: Siggraph 2017 course,” in *ACM SIGGRAPH 2017 Courses*, ser. SIGGRAPH ’17. New York, NY, USA: ACM, 2017.

[3] Z. Velinov and K. Mitchell, “Collimated whole volume light scattering in homogeneous finite media,” 2020.

[4] P. Shirley and C. Wang, “Direct lighting calculation by monte carlo integration,” in *Photorealistic Rendering in Computer Graphics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994.

[5] W. Kuijper, V. Ermolaev, and O. Devillers, “Celestial walk: A terminating, memoryless walk for convex subdivisions,” *Journal of Computer Graphics Techniques (JCGT)*, vol. 7, no. 3, pp. 29–49, September 2018. [Online]. Available: http://jcgt.org/published/0007/03/02/