Supplemental Document on BRDF Importance Sampling for Polygonal Lights

CHRISTOPH PETERS, Karlsruhe Institute of Technology, Germany

ACM Reference Format:
Christoph Peters. 2021. Supplemental Document on BRDF Importance Sampling for Polygonal Lights. *ACM Trans. Graph.* 40, 4, Article 140 (July 2021), 12 pages. https://doi.org/10.1145/3450626.3459672

In this supplemental document, we provide a few more lengthy proofs and derivations, discuss technical details of our implementation and provide complete descriptions of all of our algorithms.

A PROJECTED SOLID ANGLE SAMPLING OF POLYGONS

This section revisits different aspects of our sampling procedure in the order in which they are mentioned in the paper. It provides many of the more technical details, provides mathematical proof for several claims, describes the initialization in detail and summarizes the complete algorithms.

A.1 Clipping Polygons

In principle, clipping of convex polygons is a simple problem. We may iterate over the vertices in order. Vertices above the horizon get copied to the output sequence. If the previous vertex is above the horizon but the current vertex is below the horizon, or vice versa, a new vertex at the intersection of the edge and the horizon is output. Otherwise, the vertex is discarded.

The problem with this approach is that the location at which we write to the output array is dynamically computed. Thus, we incur the cost of register spilling on GPUs. We devised an alternative method for polygons with up to seven vertices.

First, we encode the geometric configuration into a 32-bit integer. The three least significant bits store the vertex count of the input. Each of the next seven bits is set to one, if and only if the corresponding vertex is above the horizon. If none of these bits is set, the clipped polygon is empty. If all are set, the input is the output.

Otherwise, the integer encodes which edges should be clipped to create two new vertices and which vertices need to be part of the output. We use a single large switch statement to jump to optimal code for every possible value of the integer. This branching control flow may seem like an unorthodox solution for GPUs. However, execution of branches is coherent for smooth surfaces and we only need a single jump with this design.

We generate this code through a brute force search. The search space includes different cyclic shifts of the output polygon and different orders of assignment. Our method works in situ, overwriting the input with the output. If there is enough space in the output sequence, the first vertex is repeated after the last vertex to simplify iteration over edges. If vertices that are still needed get overwritten, the candidate is discarded. Preprocessor directives specialize the code to a maximal vertex count. The generated code and the code generator are part of the supplementary materials.

In most cases, our method overwrites only three vertices. In the worst case it overwrites six vertices. We have compared our method to prior work for clipping of triangles and quads on GPUs [McGuire, 2011]. We were unable to measure a significant difference in run times. However, our method supports polygons with up to seven vertices.

A.2 Great Circles through the Zenith

The algorithms described in the paper assume that the normal vector of an edge n_j has non-zero z-coordinate $n_{j,\mathrm{z}}$. If it gets close to zero, our methods remain stable but once the coordinate underflows, they fail. This section explains how we handle the case $n_{j,\mathrm{z}} = 0$ correctly. This effort is worthwhile because the case is fairly common in man-made environments with lots of right angles.

Our approach assumes that no three vertices of the polygon are collinear. Otherwise, it is possible to remove the offending vertices without changing the polygon. If the edge normal satisfies $n_{j,\mathrm{z}} = 0$, its plane contains the zenith $(0, 0, 1)^\mathsf{T}$. Thus, the edge is an arc of a great circle through the zenith. We call such edges and the corresponding ellipses degenerate.

We make three small adjustments to our algorithm:

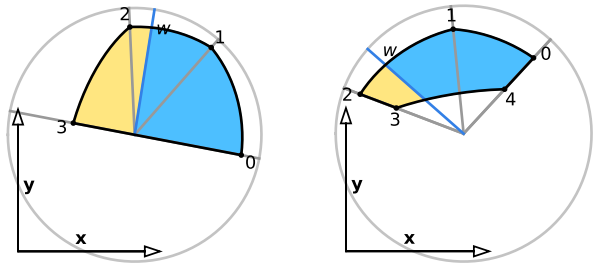
(1) For degenerate ellipses, we set $u_{j,\mathrm{x}} = \infty$ (which is representable in floating point numbers). Thus, degenerate ellipses are marked as such and additionally count as outer ellipses,
(2) If a degenerate ellipse is involved, any attempt to compute the area within the sector according to Equation (1) yields zero area,
(3) If two vertices compare equal during sorting, vertices associated with a degenerate ellipse come first.

We now explain why these rules give the correct behavior in all cases. Once again, there are two major cases to distinguish (Fig. A.1). Either a degenerate edge passes through the zenith or the edge itself does not pass through the zenith but its great circle does.

If an edge passes through the zenith, this edge counts as outer edge, just like all other edges of this polygon (Fig. A.1a). Thus, our algorithm treats the situation as central case. The sector for the degenerate edge is large but its intersection with the polygon has zero area. Due to rule 2, the area for this sector is computed correctly. Since it is zero, there will never be an attempt to sample from this

Author's address: Christoph Peters, christoph.peters@kit.edu, Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131, Karlsruhe, Germany.

0730-0301/2021/7-ART140 $15.00
https://doi.org/10.1145/3450626.3459672



(a) Central case (b) Decentral case

Fig. A.1. The great circles of edges may pass through the zenith. If the edge itself passes through the zenith, the central case is present. The edge gives rise to a 180° sector but the projected solid angle of the polygon within this sector is zero. Otherwise, the decentral case is present and these edges give rise to empty sectors.

sector. The other sectors do not involve degenerate edges and get sampled correctly.

A related case occurs when all ellipses are degenerate. It arises when the shading point lies in the plane of the polygon. In that case, rule 2 correctly computes zero projected solid angle and the polygon does not get sampled at all.

Otherwise, there are at most two degenerate edges: One at the clockwise end of the polygon with index $j_{\mathrm{cw}} \in \{0, \dots, m-1\}$ and one at its counterclockwise end with index $j_{\mathrm{ccw}} \in \{0, \dots, m-1\}$. For example, in Fig. A.1b $j_{\mathrm{cw}} = 4$ and $j_{\mathrm{ccw}} = 2$. More degenerate edges would require a non-convex polygon or collinear vertices, which we forbid. Since the degenerate ellipses count as outer ellipses, they get associated with vertices j_{cw}, j_{ccw}.

Vertices j_{cw} and $j_{\mathrm{cw}} + 1$ compare equal. Then due to rule 3, the degenerate edge comes first in the sorted list. The first sector is empty and associated with the degenerate ellipse j_{cw}. By rule 2, its area evaluates to zero and it will not be sampled. The outer ellipse for the next sector is the one associated with vertex $j_{\mathrm{cw}} + 1$, which is the correct outcome. Inner ellipses are never degenerate by rule 1.

What happens in the sector at the counterclockwise end barely matters. If one of the ellipses is degenerate, its area evaluates to zero by rule 2. Otherwise, it evaluates to zero because the sector itself is empty. Either way, it is not sampled.

Note that these arguments still imply correctness if either one of the degenerate edges does not exist. The degenerate sectors turn into common sectors and are handled correctly as usual. Overall, all cases with or without degenerate ellipses are handled properly.

A.3 Iterative Procedure for the Decentral Case

In each step, our iterative procedure finds a direction $w \in \mathbb{R}^2$ such that the area enclosed between two tangent lines within the sector w_n, w is exactly $A_\delta(w_n)$. The direction is characterized by the homogeneous quadratic equation $w^{\mathsf{T}} T w = 0$. This quadratic commonly has two different solutions, i.e. the solution set consists of two lines through the origin. In this section, we address the rather technical question how our algorithm picks the correct solution.

The ambiguity of solutions stems from the fact that our areas are signed. The ray through w_n first intersects the tangent line for the inner ellipse and then the one for the outer ellipse. As we move w counterclockwise starting at w_n, the enclosed area grows for some time. However, if the ray through w reaches the intersection of the two tangent lines, the tangent line for the outer ellipse becomes the inner tangent line. At this point, contributions to the area become negative and the enclosed area starts shrinking. As w becomes parallel to the inner tangent, the negative contribution becomes infinite. Thus, we certainly find another root somewhere between the global maximum at the intersection and the direction of the inner tangent.

If the intersection is located clockwise of w_n, it works analogically. We know that it is not exactly at w_n because ellipses only intersect at boundaries of the domain but there we disable the iteration. If the solution to the homogeneous quadratic is exactly at the intersection, we have a double root because the root is simultaneously a critical point. In that case the discriminant vanishes.

We could compute both roots and discard the one on the wrong side of the intersection. However, it is more efficient to compute only the relevant root. To this end, we ensure that results of our quadratic solver depend continuously on the input.

Lemma 1. *Let $T \in \mathbb{R}^{2\times 2}$ such that $\Delta := -\frac{1}{4}|T + T^{\mathsf{T}}| > 0$ and let*

$$w_{n+1} := \begin{cases} \left(\frac{|T_{\mathrm{x,y}}+T_{\mathrm{y,x}}|}{2} + \sqrt{\Delta},\ -T_{\mathrm{x,x}}\right)^{\mathsf{T}} & \text{if } T_{\mathrm{x,y}} + T_{\mathrm{y,x}} \geq 0, \\ \left(T_{\mathrm{y,y}},\ \frac{|T_{\mathrm{x,y}}+T_{\mathrm{y,x}}|}{2} + \sqrt{\Delta}\right)^{\mathsf{T}} & \text{otherwise.} \end{cases}$$

If we interpret $w_{n+1} \in \mathbb{R}^2$ as point in projective space, it depends on T continuously.

Proof. Both of the possible outputs depend on T continuously. We only need to prove that the transition at the case $T_{\mathrm{x,y}} + T_{\mathrm{y,x}} = 0$ is continuous. In that case, the possible outputs simplify to

$$(\sqrt{\Delta},\ -T_{\mathrm{x,x}})^{\mathsf{T}}, \qquad (T_{\mathrm{y,y}},\ \sqrt{\Delta})^{\mathsf{T}}, \qquad \text{where } \Delta = -T_{\mathrm{x,x}} T_{\mathrm{y,y}}.$$

To prove that these vectors are identical in the projective sense, we make a case distinction based on the sign of $T_{\mathrm{y,y}}$. Note that $T_{\mathrm{x,x}}$ has opposite sign since $\Delta > 0$.

Case 1. If $T_{\mathrm{y,y}} > 0$

$$\begin{pmatrix} \sqrt{\Delta} \\ -T_{\mathrm{x,x}} \end{pmatrix} = \begin{pmatrix} \sqrt{-T_{\mathrm{x,x}} T_{\mathrm{y,y}}} \\ \sqrt{-T_{\mathrm{x,x}}}\sqrt{-T_{\mathrm{x,x}}} \end{pmatrix} = \sqrt{\frac{-T_{\mathrm{x,x}}}{T_{\mathrm{y,y}}}} \begin{pmatrix} T_{\mathrm{y,y}} \\ \sqrt{-T_{\mathrm{x,x}} T_{\mathrm{y,y}}} \end{pmatrix} = \sqrt{-\frac{T_{\mathrm{x,x}}}{T_{\mathrm{y,y}}}} \begin{pmatrix} T_{\mathrm{y,y}} \\ \sqrt{\Delta} \end{pmatrix}.$$

Case 2. If $T_{\mathrm{y,y}} < 0$

$$\begin{pmatrix} \sqrt{\Delta} \\ -T_{\mathrm{x,x}} \end{pmatrix} = \begin{pmatrix} \sqrt{T_{\mathrm{x,x}}(-T_{\mathrm{y,y}})} \\ -\sqrt{T_{\mathrm{x,x}}}\sqrt{T_{\mathrm{x,x}}} \end{pmatrix} = \sqrt{\frac{T_{\mathrm{x,x}}}{-T_{\mathrm{y,y}}}} \begin{pmatrix} -T_{\mathrm{y,y}} \\ -\sqrt{T_{\mathrm{x,x}}(-T_{\mathrm{y,y}})} \end{pmatrix} = -\sqrt{-\frac{T_{\mathrm{x,x}}}{T_{\mathrm{y,y}}}} \begin{pmatrix} T_{\mathrm{y,y}} \\ \sqrt{\Delta} \end{pmatrix}.$$

□

Now we prove that this strategy gives us the correct root if the current direction w_n is close enough to the converged result of the iteration, i.e. if $|A_\delta(w_n)|$ is small enough. Such a local result is enough to retain local cubic convergence. Our extensive search for failure cases shows that the error with our initialization strategy is indeed small enough.

Proposition 2. *Let $w_n \in \mathbb{R}^2$ with $w_n \neq 0$ and consider the homogeneous quadratic for our iterative procedure*

$$T := T(w_n) := Rw_n w_n^\mathsf{T}(C_i - C_o) - 2A_\delta(w_n)C_i w_n w_n^\mathsf{T} C_o.$$

Suppose $A_\delta(w_n) = 0$, i.e. w_n already solves our inverse problem, and

$$w_n^\mathsf{T} C_i w_n > w_n^\mathsf{T} C_o w_n,$$

i.e. it does not point towards the intersection of the two ellipses. Then the discriminant Δ is positive and the homogeneous quadratic solver in Lemma 1 applied to $T(w_n)$ produces a non-zero multiple of w_n.

Additionally, sufficiently small perturbations to A (or equivalently $A_\delta(w_n)$) do not lead to a vanishing discriminant and induce continuous changes in the solution of the homogeneous quadratic.

Proof. We prove that the solution is a non-zero multiple of w_n constructively. We begin with a reformulation of the discriminant:

$$\begin{aligned}
4\Delta &= -|T(w_n) + T^\mathsf{T}(w_n)| \\
&= -|Rw_n w_n^\mathsf{T}(C_i - C_o) + (C_i - C_o)w_n w_n^\mathsf{T} R| \\
&= -\left|(Rw_n, (C_i - C_o)w_n)\begin{pmatrix} w_n^\mathsf{T}(C_i - C_o) \\ w_n^\mathsf{T} R\end{pmatrix}\right| \\
&= |(Rw_n, (C_i - C_o)w_n)|^2 \\
&= (w_n^\mathsf{T}(C_i - C_o)RRw_n)^2 \\
&= (w_n^\mathsf{T}(C_i - C_o)w_n)^2
\end{aligned}$$

Then due to our assumption $\Delta > 0$. Since the discriminant is positive, the solution to the homogeneous quadratic in Lemma 1 cannot be zero either.

We also formulate new expressions for the entries of T. Using $e_0 := (1,0)^\mathsf{T}$ and $e_1 := (0,1)^\mathsf{T}$

$$\begin{aligned}
T_{\mathrm{x,x}} &= e_0^\mathsf{T} T e_0 = (e_0^\mathsf{T} R w_n)(w_n^\mathsf{T}(C_i - C_o)e_0) \\
&= -w_n^\mathsf{T}(C_i - C_o)e_0 w_{n,\mathrm{y}}, \\
T_{\mathrm{y,x}} + T_{\mathrm{x,y}} &= e_1^\mathsf{T} R w_n w_n^\mathsf{T}(C_i - C_o)e_0 + e_0^\mathsf{T} R w_n w_n^\mathsf{T}(C_i - C_o)e_1 \\
&= w_n^\mathsf{T}(C_i - C_o)(e_0 w_{n,\mathrm{x}} - e_1 w_{n,\mathrm{y}}), \\
T_{\mathrm{y,y}} &= (e_1^\mathsf{T} R w_n)(w_n^\mathsf{T}(C_i - C_o)e_1) \\
&= w_n^\mathsf{T}(C_i - C_o)e_1 w_{n,\mathrm{x}}.
\end{aligned}$$

To prove that we get some multiple of w_n, we make the same case distinction as in the quadratic solver.

Case 1. If $T_{\mathrm{x,y}} + T_{\mathrm{y,x}} \geq 0$, the solution is

$$\begin{aligned}
\begin{pmatrix} \frac{|T_{\mathrm{x,y}} + T_{\mathrm{y,x}}|}{2} + \sqrt{\Delta} \\ -T_{\mathrm{x,x}} \end{pmatrix} &= \frac{1}{2}\begin{pmatrix} w_n^\mathsf{T}(C_i - C_o)(e_0 w_{n,\mathrm{x}} - e_1 w_{n,\mathrm{y}} + w_n) \\ 2w_n^\mathsf{T}(C_i - C_o)e_0 w_{n,\mathrm{y}} \end{pmatrix} \\
&= w_n^\mathsf{T}(C_i - C_o)e_0 w_n.
\end{aligned}$$

Case 2. If $T_{\mathrm{x,y}} + T_{\mathrm{y,x}} < 0$, the solution is

$$\begin{aligned}
\begin{pmatrix} T_{\mathrm{y,y}} \\ \frac{|T_{\mathrm{x,y}} + T_{\mathrm{y,x}}|}{2} + \sqrt{\Delta} \end{pmatrix} &= \frac{1}{2}\begin{pmatrix} 2w_n^\mathsf{T}(C_i - C_o)e_1 w_{n,\mathrm{x}} \\ w_n^\mathsf{T}(C_i - C_o)(-e_0 w_{n,\mathrm{x}} + e_1 w_{n,\mathrm{y}} + w_n) \end{pmatrix} \\
&= w_n^\mathsf{T}(C_i - C_o)e_1 w_n.
\end{aligned}$$

$T(w_n)$ depends continuously (even linearly) on A and so does Δ. Thus, a neighborhood where the discriminant is still positive exists. Applying Lemma 1, we then find that the solution changes continuously as well. □

In other words, if w_n is already the correct solution, we compute $w_{n+1} = w_n$ and do not move away from it. Within a neighborhood of $A_\delta(w_n) = 0$ there are no double roots and the next direction w_{n+1} depends continuously on A as it should. Thus, we still pick the correct root there. From the proof and our previous considerations it is clear that these arguments may collapse at the point where the two tangent lines intersect. However, our search for failure cases shows that this point is never too close, at least if we disable the iteration when ξ_0 is too close to the boundary.

A.4 Initialization for the Decentral Case

Recall that the sector boundary $j \in \{0, h, 1\}$ intersects ellipse $l \in \{i, o\}$ at $\lambda_{l,j} s_j$. Disregarding the sign, the area of the quad in sector s_h, s_k where $k \in \{0, 1\}$ is

$$\begin{aligned}
&\frac{1}{2}|(\lambda_{o,h}s_h, \lambda_{o,k}s_k)| - \frac{1}{2}|(\lambda_{i,h}s_h, \lambda_{i,k}s_k)| \\
=&\frac{1}{2}(\lambda_{o,h}\lambda_{o,k} - \lambda_{i,h}\lambda_{i,k})|(s_h, s_k)|.
\end{aligned}$$

Note that $-|(s_h, s_0)| = |(s_h, s_1)|$ since s_h is a half-vector. Thus, we drop this factor and the factor $\frac{1}{2}$ during randomized selection of a quad and incorporate it later.

Our algorithm selects a quad $k \in \{0, 1\}$. For all ellipses $l \in \{i, o\}$, the paper defines the edge normal

$$r_l = C_l(\lambda_{l,h}s_h + \lambda_{l,k}s_k).$$

Indeed, this vector is orthogonal to the edge connecting $\lambda_{l,h}s_h$ to $\lambda_{l,k}s_k$ because

$$\begin{aligned}
&(\lambda_{l,h}s_h - \lambda_{l,k}s_k)^\mathsf{T} r_l \\
=&\lambda_{l,h}^2 s_h^\mathsf{T} C_l s_h + \lambda_{l,h}\lambda_{l,k} s_h^\mathsf{T} C_l s_k - \lambda_{l,k}\lambda_{l,h} s_k^\mathsf{T} C_l s_h - \lambda_{l,k}^2 s_k^\mathsf{T} C_l s_k \\
=&\lambda_{l,h}^2 s_h^\mathsf{T} C_l s_h - \lambda_{l,k}^2 s_k^\mathsf{T} C_l s_k \\
=&1 - 1 = 0.
\end{aligned}$$

A point $q \in \mathbb{R}^2$ lies on this edge if $r_l^\mathsf{T} q = D_l$.

The intersection of a ray through $w \in \mathbb{R}^2$ with this edge is at

$$\frac{D_l}{r_l^\mathsf{T} w} w.$$

Then the signed area in the sector s_k, w between the two edges is

$$\begin{aligned}
&\frac{1}{2}\left|\left(\lambda_{o,k}s_k, \frac{D_o}{r_o^\mathsf{T} w} w\right)\right| - \frac{1}{2}\left|\left(\lambda_{i,k}s_k, \frac{D_i}{r_i^\mathsf{T} w} w\right)\right| \\
=&\frac{1}{2}\left(\frac{\lambda_{o,k}D_o}{r_o^\mathsf{T} w} - \frac{\lambda_{i,k}D_i}{r_i^\mathsf{T} w}\right)|(s_k, w)|.
\end{aligned}$$

For sampling, we have to select w such that this area matches a prescribed value $A_q \in \mathbb{R}$:

$$\begin{aligned}
&\frac{1}{2}\left(\frac{\lambda_{o,k}D_o}{r_o^\mathsf{T} w} - \frac{\lambda_{i,k}D_i}{r_i^\mathsf{T} w}\right)|(s_k, w)| = A_q \\
\Leftrightarrow\ &(\lambda_{o,k}D_o r_i^\mathsf{T} w - \lambda_{i,k}D_i r_o^\mathsf{T} w) w^\mathsf{T} R s_k = 2A_q w^\mathsf{T} r_o r_i^\mathsf{T} w \\
\Leftrightarrow\ &w^\mathsf{T}(R s_k(\lambda_{o,k}D_o r_i^\mathsf{T} - \lambda_{i,k}D_i r_o^\mathsf{T}) - 2A_q r_o r_i^\mathsf{T})w = 0 \\
\Leftrightarrow\ &w^\mathsf{T}\underbrace{(\lambda_{o,k}D_o R s_k r_i^\mathsf{T} - (\lambda_{i,k}D_i R s_k + 2A_q r_i) r_o^\mathsf{T})}_{=:Q} w = 0
\end{aligned}$$

As for the iteration, we have reduced the problem to solving a homogeneous quadratic equation $w^\mathsf{T} Q w = 0$. This time, the homogeneous quadratic solver in Lemma 1 is guaranteed to compute the correct root globally:

Proposition 3. *Let A_q lie between 0 and the signed area of quad k*

$$\frac{1}{2}(\lambda_{o,h}\lambda_{o,k} - \lambda_{i,h}\lambda_{i,k})|(s_h, s_k)|.$$

Suppose that $\lambda_{i,k} < \lambda_{o,k}$. Then the homogeneous quadratic Q has positive discriminant and the solver in Lemma 1 computes a root within the quad.

Proof. We begin with the case $A_q = 0$. Using $D_l = \lambda_{l,k} r_l^\mathsf{T} s_k$, the quadratic simplifies to

$$\begin{aligned}
Q &= R s_k(\lambda_{o,k}D_o r_i^\mathsf{T} - \lambda_{i,k}D_i r_o^\mathsf{T}) \\
&= R s_k(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}).
\end{aligned}$$

Clearly, $s_k^\mathsf{T} Q s_k = 0$ and our goal is to prove that the solver in Lemma 1 produces this solution. We begin by reformulating the discriminant as in the proof of Proposition 2:

$$\begin{aligned}
4\Delta &= |(R s_k, \lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o)|^2 \\
&= (\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} R R s_k - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T} R R s_k)^2 \\
&= ((\lambda_{o,k}^2 - \lambda_{i,k}^2) r_i^\mathsf{T} s_k r_o^\mathsf{T} s_k)^2.
\end{aligned}$$

We know $\lambda_{o,k}^2 - \lambda_{i,k}^2 > 0$. Note that $R u_l \in \mathbb{R}^2$ is the direction where the great circle intersects the horizon. Thus, this vector cannot be located between two sector boundaries. Therefore,

$$\begin{aligned}
r_l^\mathsf{T} s_k &= s_k^\mathsf{T} C_l(\lambda_{l,h} s_h + \lambda_{l,k} s_k) \\
&= \lambda_{l,h} s_k^\mathsf{T} s_h + s_k^\mathsf{T} u_l u_l^\mathsf{T} s_h + \lambda_{l,k} s_k^\mathsf{T} C_l s_k \\
&> \lambda_{l,h} s_k^\mathsf{T} s_h \geq 0.
\end{aligned}$$

The last step exploits that sectors are at most 90° large once we have split them in half. Therefore, the discriminant is positive and the solver returns a non-zero vector.

As before, we also reformulate the polynomial coefficients.

$$\begin{aligned}
Q_{\mathrm{x,x}} &= e_0^\mathsf{T} Q e_0 = -(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) e_0 s_{k,\mathrm{y}} \\
Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}} &= e_1^\mathsf{T} Q e_0 + e_0^\mathsf{T} Q e_1 \\
&= (\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T})(e_0 s_{k,\mathrm{x}} - e_1 s_{k,\mathrm{y}}) \\
Q_{\mathrm{y,y}} &= e_1^\mathsf{T} Q e_1 = (\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) e_1 s_{k,\mathrm{x}}
\end{aligned}$$

Case 1. If $Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}} \geq 0$,

$$\begin{aligned}
&|Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}}| + 2\sqrt{\Delta} \\
=&(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T})(2e_0 s_{k,\mathrm{x}} - s_k) + (\lambda_{o,k}^2 - \lambda_{i,k}^2) r_i^\mathsf{T} s_k r_o^\mathsf{T} s_k \\
=&(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) 2e_0 s_{k,\mathrm{x}}.
\end{aligned}$$

Then the solver returns the root

$$\begin{pmatrix} \frac{|Q_{\mathrm{x,y}}+Q_{\mathrm{y,x}}|}{2} + \sqrt{\Delta} \\ -Q_{\mathrm{x,x}} \end{pmatrix} = (\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) e_0 s_k.$$

Case 2. If $Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}} < 0$,

$$\begin{aligned}
&|Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}}| + 2\sqrt{\Delta} \\
=&(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T})(2e_1 s_{k,\mathrm{y}} - s_k) + (\lambda_{o,k}^2 - \lambda_{i,k}^2) r_i^\mathsf{T} s_k r_o^\mathsf{T} s_k \\
=&(\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) 2e_1 s_{k,\mathrm{y}}.
\end{aligned}$$

Then the solver returns the root

$$\begin{pmatrix} Q_{\mathrm{y,y}} \\ \frac{|Q_{\mathrm{x,y}}+Q_{\mathrm{y,x}}|}{2} + \sqrt{\Delta} \end{pmatrix} = (\lambda_{o,k}^2 r_o^\mathsf{T} s_k r_i^\mathsf{T} - \lambda_{i,k}^2 r_i^\mathsf{T} s_k r_o^\mathsf{T}) e_1 s_k.$$

Thus, we get the correct result if $A_q = 0$. We know that the edges do not intersect within the quad and therefore the discriminant remains positive for relevant values of A_q. Then by Lemma 1, the computed root as function of A_q changes continuously. Since double roots cannot occur, we must get the correct root throughout the quad, which is the one within the quad. □

The only case excluded by Proposition 3 is the one where the ray through s_k intersects both quad edges in the same point. This case commonly occurs for the first and last sector. Since our method behaves continuously, it still picks the correct solution in this case. We carefully designed the implementation to ensure that this remains true in presence of rounding errors. However, in some degenerate variants of this case, it is possible that the homogeneous quadratic solver returns a zero vector. This problem could be avoided with two additional branches for the case $Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}} = 0$ in the quadratic solver [Blinn, 2006]. Though, we never encountered issues in practice and omit the additional branches for the sake of increased efficiency.

Algorithm 3 presents our complete initialization procedure among with the iterative procedure.

A.5 Theoretical Error Analysis

Next we prove the claims on local cubic convergence. The core of this proof is to show that the mapping from w_n to w_{n+1} has two vanishing derivatives when w_n is already converged. Thus, we first take two derivatives of our algorithm and show that they vanish under the right circumstances. Once we have proven that, we prove local cubic convergence.

We use the following conventions for these calculations:

Definition 4. We are given inner and outer ellipses $C_i, C_o \in \mathbb{R}^{2\times 2}$. As before, we consider the residual error and the homogeneous quadratic as functions of $w_n \in \mathbb{R}^2$ with $w_n \neq 0$, i.e.

$$A_\delta(w_n) := A - \frac{1}{2\sqrt{|C_o|}}\operatorname{atan2}\left(\frac{-s_0^\mathsf{T} R w_n}{\frac{1}{\sqrt{|C_o|}} s_0^\mathsf{T} C_o w_n}\right) + \frac{1}{2\sqrt{|C_i|}}\operatorname{atan2}\left(\frac{-s_0^\mathsf{T} R w_n}{\frac{1}{\sqrt{|C_i|}} s_0^\mathsf{T} C_i w_n}\right),$$

$$T(w_n) := R w_n w_n^\mathsf{T}(C_i - C_o) - 2A_\delta(w_n) C_i w_n w_n^\mathsf{T} C_o.$$

The function $w_{n+1}(w_n)$ implements our iteration, i.e. $w_{n+1}(w_n) \neq 0$,

$$w_{n+1}^\mathsf{T}(w_n) T(w_n) w_{n+1}(w_n) = 0,$$

and this root is selected by the solver in Lemma 1. For a direction $v \in \mathbb{R}^2$, we write

$$\frac{\partial}{\partial v} A_\delta(w_n) \in \mathbb{R}$$

to denote the derivative of A_δ with respect to variable w_n in direction v.

The laborious part of the proof is concerned with the following lemma:

Lemma 5. *Let $w_n \in \mathbb{R}^2$ with*

$$w_n \neq 0, \qquad A_\delta(w_n) = 0, \qquad w_n^\mathsf{T} C_i w_n > w_n^\mathsf{T} C_o w_n.$$

Let $w_{n+1} := w_{n+1}(w_n)$. Then the first two directional derivatives of $w_{n+1}(w_n)$ in direction $v := R w_n$ satisfy

$$v^\mathsf{T} \frac{\partial}{\partial v} w_{n+1}(w_n) = 0, \tag{A.1}$$

$$v^\mathsf{T} \frac{\partial^2}{\partial v^2} w_{n+1}(w_n) = 0. \tag{A.2}$$

Proof. First, we note that Proposition 2 is applicable and thus w_{n+1} is a non-zero multiple of w_n. Let $l \in \{i, o\}$. We begin by computing first-order derivatives:

$$\frac{\partial}{\partial v} T(w_n) = R(v w_n^\mathsf{T} + w_n v^\mathsf{T})(C_i - C_o) - 2\frac{\partial A_\delta(w_n)}{\partial v} C_i w_n w_n^\mathsf{T} C_o - 2A_\delta(w_n) C_i (v w_n^\mathsf{T} + w_n v^\mathsf{T}) C_o$$

Let $LL^\mathsf{T} := C_l$ be a Cholesky decomposition and recall from the proof of Proposition 1 that

$$\operatorname{atan2}\left(\frac{-s_0^\mathsf{T} R w_n}{\frac{1}{\sqrt{|C_o|}} s_0^\mathsf{T} C_o w_n}\right) = \operatorname{atan2}\left(\frac{(RL^\mathsf{T} s_0)^\mathsf{T} L^\mathsf{T} w_n}{(L^\mathsf{T} s_0)^\mathsf{T} L^\mathsf{T} w_n}\right).$$

For the purpose of derivative computation, a constant offset is irrelevant. Thus, it is safe to replace $L^\mathsf{T} s_0$ by $e_0 := (1,0)^\mathsf{T}$ and accordingly $RL^\mathsf{T} s_0$ by $e_1 := (0,1)^\mathsf{T}$. Then

$$\frac{\partial}{\partial v}\operatorname{atan2}\left(\frac{-s_0^\mathsf{T} R w_n}{\frac{1}{\sqrt{|C_o|}} s_0^\mathsf{T} C_o w_n}\right) = \frac{\partial}{\partial v}\operatorname{atan2}\begin{pmatrix} e_1^\mathsf{T} L^\mathsf{T} w_n \\ e_0^\mathsf{T} L^\mathsf{T} w_n\end{pmatrix} = \frac{v^\mathsf{T} L R L^\mathsf{T} w_n}{w_n^\mathsf{T} L L^\mathsf{T} w_n} = \frac{|L| v^\mathsf{T} R w_n}{w_n^\mathsf{T} L L^\mathsf{T} w_n} = \sqrt{|C_l|}\frac{v^\mathsf{T} R w_n}{w_n^\mathsf{T} C_l w_n}.$$

Thus, the derivative of $A_\delta(w_n)$ is

$$\frac{\partial A_\delta(w_n)}{\partial v} = -\frac{1}{2}\frac{v^\mathsf{T} R w_n}{w_n^\mathsf{T} C_o w_n} + \frac{1}{2}\frac{v^\mathsf{T} R w_n}{w_n^\mathsf{T} C_i w_n}.$$

Coincidentally, this is the derivation for the derivative in Equation 9, which would be used by Newton's method.

We know for all $w_n \in \mathbb{R}^2$ with $w_n \neq 0$

$$\begin{aligned} & w_{n+1}^\mathsf{T}(w_n) T(w_n) w_{n+1}(w_n) = 0 \\ \Rightarrow\ & w_{n+1}^\mathsf{T}(T(w_n) + T^\mathsf{T}(w_n))\frac{\partial w_{n+1}(w_n)}{\partial v} = -w_{n+1}^\mathsf{T}\frac{\partial T(w_n)}{\partial v} w_{n+1} \end{aligned} \tag{A.3}$$

By definition

$$w_{n+1}^\mathsf{T}(T(w_n) + T^\mathsf{T}(w_n)) w_{n+1} = 0$$

but by our assumption

$$w_{n+1}^\mathsf{T}(T(w_n) + T^\mathsf{T}(w_n)) = w_{n+1}^\mathsf{T}(C_i - C_o) w_n w_n^\mathsf{T} R^\mathsf{T} \neq 0.$$

Thus, we have a non-zero vector orthogonal to w_n. Since we are in 2D, it must be a multiple of $v = R w_n$. Then to prove Equation (A.1), it suffices to prove that the right-hand side in Equation (A.3) vanishes:

$$\begin{aligned} & w_n^\mathsf{T}\frac{\partial}{\partial v} T(w_n) w_n \\ =& w_n^\mathsf{T} R(v w_n^\mathsf{T} + w_n v^\mathsf{T})(C_i - C_o) w_n - 2\frac{\partial A_\delta(w_n)}{\partial v} w_n^\mathsf{T} C_i w_n w_n^\mathsf{T} C_o w_n \\ =& w_n^\mathsf{T} R v w_n^\mathsf{T}(C_i - C_o) w_n + (w_n^\mathsf{T} C_i w_n - w_n^\mathsf{T} C_o w_n) v^\mathsf{T} R w_n \\ =& 0. \end{aligned}$$

Note that we have exploited $v^\mathsf{T} R w_n = -w_n^\mathsf{T} R v$.

For second-order derivatives, we exploit $A_\delta(w_n) = 0$ immediately since we do not need further derivatives:

$$\begin{aligned} \frac{\partial^2}{\partial v^2} T(w_n) &= 2R v v^\mathsf{T}(C_i - C_o) - 2\frac{\partial^2 A_\delta(w_n)}{\partial v^2} C_i w_n w_n^\mathsf{T} C_o \\ &\quad - 4\frac{\partial A_\delta(w_n)}{\partial v} C_i (v w_n^\mathsf{T} + w_n v^\mathsf{T}) C_o, \\ \frac{\partial}{\partial v}\frac{1}{2}\frac{v^\mathsf{T} R w_n}{w_n^\mathsf{T} C_l w_n} &= \frac{1}{2}\frac{v^\mathsf{T} R v w_n^\mathsf{T} C_l w_n - 2 v^\mathsf{T} R w_n w_n^\mathsf{T} C_l v}{(w_n^\mathsf{T} C_l w_n)^2} \\ &= -\frac{v^\mathsf{T} R w_n w_n^\mathsf{T} C_l v}{(w_n^\mathsf{T} C_l w_n)^2}, \\ \frac{\partial^2 A_\delta(w_n)}{\partial v^2} &= \frac{v^\mathsf{T} R w_n w_n^\mathsf{T} C_o v}{(w_n^\mathsf{T} C_o w_n)^2} - \frac{v^\mathsf{T} R w_n w_n^\mathsf{T} C_i v}{(w_n^\mathsf{T} C_i w_n)^2}. \end{aligned}$$

Since we are in 2D, Equation (A.1) implies that $\frac{\partial w_{n+1}(w_n)}{\partial v}$ is a multiple of w_n. In particular,

$$\begin{aligned} 0 =& w_n^\mathsf{T} T(w_n) w_n = \frac{\partial w_{n+1}^\mathsf{T}(w_n)}{\partial v} T(w_n) w_n \\ =& \frac{\partial w_{n+1}^\mathsf{T}(w_n)}{\partial v} T(w_n)\frac{\partial w_{n+1}(w_n)}{\partial v}, \\ 0 =& w_{n+1}^\mathsf{T}(w_n)\frac{\partial T(w_n)}{\partial v} w_{n+1}(w_n) \\ =& \frac{\partial w_{n+1}^\mathsf{T}(w_n)}{\partial v}\frac{\partial T(w_n)}{\partial v} w_{n+1}(w_n). \end{aligned}$$

As we take a second directional derivative of Equation (A.3), only terms with second derivatives of $T(w_n)$ or $w_{n+1}(w_n)$ remain, i.e.

$$w_{n+1}^\mathsf{T}(T(w_n) + T^\mathsf{T}(w_n))\frac{\partial^2 w_{n+1}(w_n)}{\partial v^2} = -w_{n+1}^\mathsf{T}\frac{\partial^2 T(w_n)}{\partial v^2} w_{n+1}.$$

For the same reasons as before, it now suffices to prove that the right-hand side vanishes:

$$\begin{aligned}
&w_n^\mathsf{T} \frac{\partial^2 T(w_n)}{\partial v^2} w_n \\
=&2 w_n^\mathsf{T} R v v^\mathsf{T} (C_i - C_o) w_n \\
&-2 \frac{\partial^2 A_\delta(w_n)}{\partial v^2} w_n^\mathsf{T} C_i w_n w_n^\mathsf{T} C_o w_n \\
&-4 \frac{\partial A_\delta(w_n)}{\partial v} (w_n^\mathsf{T} C_i v w_n^\mathsf{T} C_o w_n + w_n^\mathsf{T} C_i w_n v^\mathsf{T} C_o w_n) \\
=&2 v^\mathsf{T} R w_n (w_n^\mathsf{T} C_o v - w_n^\mathsf{T} C_i v) \\
&-2 v^\mathsf{T} R w_n \left(w_n^\mathsf{T} C_o v \frac{w_n^\mathsf{T} C_i w_n}{w_n^\mathsf{T} C_o w_n} - w_n^\mathsf{T} C_i v \frac{w_n^\mathsf{T} C_o w_n}{w_n^\mathsf{T} C_i w_n} \right) \\
&-2 v^\mathsf{T} R w_n \left(w_n^\mathsf{T} C_i v \frac{w_n^\mathsf{T} C_o w_n}{w_n^\mathsf{T} C_i w_n} + w_n^\mathsf{T} C_o v - w_n^\mathsf{T} C_i v - w_n^\mathsf{T} C_o v \frac{w_n^\mathsf{T} C_i w_n}{w_n^\mathsf{T} C_o w_n} \right) \\
=&0.
\end{aligned}$$

□

The proof of local cubic convergence is now mostly a matter of transitioning from direction vectors to angles. For $w \in \mathbb{R}^2$ with $w \neq 0$ and $\alpha \in \mathbb{R}$, we introduce the functions

$$\alpha(w) := \operatorname{atan2}(w_y, w_x), \qquad w(\alpha) := (\cos\alpha, \sin\alpha)^\mathsf{T}.$$

If $\alpha(w)$ has its discontinuity at an inconvenient location, we move it somewhere else using e.g.

$$\alpha(w) := \operatorname{atan2}(w_x, -w_y) - \frac{\pi}{2}.$$

Proposition 6. *Let $w_* \in \mathbb{R}^2$ with*

$$w_* \neq 0, \qquad A_\delta(w_*) = 0, \qquad w_n^\mathsf{T} C_i w_n > w_n^\mathsf{T} C_o w_n.$$

Let $w_0 \in \mathbb{R}^2$ be sufficiently close to w_* and for all $n \in \mathbb{N}_0$ consider the sequence $w_{n+1} := w_{n+1}(w_n)$. Then there exists a $\zeta > 0$ such that for all $n \in \mathbb{N}_0$

$$|\alpha(w_{n+1}) - \alpha(w_*)| \leq \zeta |\alpha(w_n) - \alpha(w_*)|^3.$$

Proof. For $\alpha \in \mathbb{R}$, consider the function

$$\beta_{n+1}(\alpha) := \alpha(w_{n+1}(w(\alpha))).$$

The chain rule and Lemma 5 applied to w_* let us compute its derivatives at $\beta_* := \alpha(w_*)$:

$$\begin{aligned}
\beta'_{n+1}(\beta_*) &= \alpha'(w_{n+1}(w(\beta_*))) \frac{\partial}{\partial w'(\beta_*)} w_{n+1}(w(\beta_*)) \\
&= \alpha'(w_{n+1}(w(\beta_*))) \frac{\partial}{\partial R w(\beta_*)} w_{n+1}(w(\beta_*)) = 0, \\
\beta''_{n+1}(\beta_*) &= \alpha''(w_{n+1}(w(\beta_*))) \frac{\partial}{\partial R w(\beta_*)} w_{n+1}(w(\beta_*)) \\
&\quad + \alpha'(w_{n+1}(w(\beta_*))) \frac{\partial^2}{\partial (R w(\beta_*))^2} w_{n+1}(w(\beta_*)) = 0.
\end{aligned}$$

Therefore, a Taylor expansion of $\beta_{n+1}(\alpha)$ around β_* takes the form

$$\beta_{n+1}(\alpha) = \beta_* + \frac{\beta'''_{n+1}(\beta_*)}{3} (\beta_* - \alpha)^3 + \dots.$$

Table A.1. The number of comparisons in our sorting networks for bitonic sequences of different size, compared to the number of comparisons in optimal sorting networks for arbitrary sequences.

Element count	3	4	5	6	7	8
Our network	3	4	8	9	12	12
General network	3	5	9	12	16	19

Then there exists a ζ slightly bigger than $\frac{1}{3}|\beta'''_{n+1}(\beta_*)|$ such that for all α sufficiently close to β_*

$$|\beta_{n+1}(\alpha) - \beta_*| \leq \zeta |\alpha - \beta_*|^3.$$

In particular,

$$|\alpha(w_{n+1}) - \beta_*| = |\beta_{n+1}(\alpha(w_n)) - \beta_*| \leq \zeta |\alpha(w_n) - \beta_*|^3.$$

□

A.6 Sorting Bitonic Sequences

In a way, the input to our algorithm is already sorted, just not with respect to the zenith. Suppose v_0 happens to be the vertex at the clockwise end of the polygon with respect to the zenith. As we move on to vertex v_j by increasing $j \in \{0, \dots, m-1\}$, we first pass along one or more outer edges, i.e. we move counterclockwise. Eventually, we encounter the first inner edge and move clockwise (Fig. 3b). Consequently, the indices indicating the positions of the vertices $v_0, \dots, v_{m-1}$ in a sorted sequence first increase and then decrease. Therefore, it is a bitonic sequence. If the vertex at the clockwise end is in a different position, we have a circular shift of such a sequence. By definition, that is still a bitonic sequence.

We seek optimal sorting networks that sort bitonic sequences with up to eight elements. A sorting network is a sequence of index pairs. In each step, the array entries at the two indices are compared and swapped if their order is wrong. As for the clipping, our search employs brute force. We prescribe a number of allowed comparisons and enumerate all sequences of index pairs of this length. We apply each of them to a complete list of bitonic sequences with entries $\{0, \dots, m-1\}$. If all sequences get sorted correctly, the sorting network is a candidate. We retain the candidate of minimal depth.

This approach scales poorly but is trivial to parallelize. We used CUDA and packed complete sequences into 32-bit integers. For up to six elements, it is no problem to try all possibilities. For seven elements, we invested 5.5 hours of GPU time to find that none of the $21^{11} \approx 3.5 \cdot 10^{14}$ sorting networks with eleven comparisons is successful. An educated guess for the last few comparisons allowed us to find sorting networks with twelve comparisons, which must be optimal. For eight elements, we found sorting networks with twelve comparisons in the same manner. Such a sorting network is also used in bitonic merge sort.

To eliminate some conditional code execution, we designed networks for different element counts that share the last few comparisons. Fig. A.2 visualizes our optimal sorting networks. Table A.1 gives the savings that we achieve by specializing our networks to bitonic sequences.

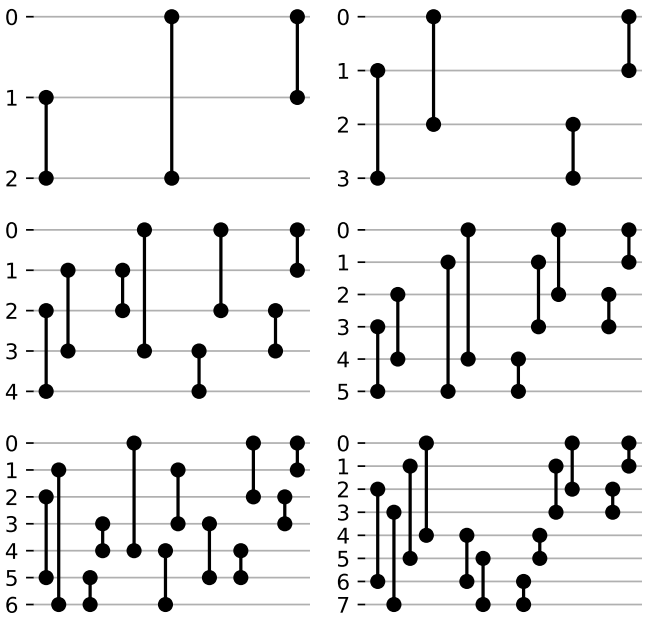


Fig. A.2. Visualizations of our optimal sorting networks for bitonic sequences. Each black line connects two channels. In sequence, the elements in these channels are compared and if the one with the greater index is less, they are swapped.

A.7 The Full Algorithm

In this section, we assemble all the pieces described thus far into the algorithm for projected solid angle sampling. First, we need to establish a few little helper functions. The function `is_inner(`u_j`)`returns true if the sign bit on $u_{j,\mathrm{x}}$ is one (indicating an inner ellipse) and false otherwise. The function `ellipse_area_in_sector(`u_j, s_0, s_1`)` implements Equation (1), i.e. it evaluates to

$$\frac{1}{2\sqrt{1+\|u_j\|^2}}\,\mathrm{atan2}\begin{pmatrix} -s_0^\mathsf{T} R s_1 \\ \frac{1}{\sqrt{1+\|u_j\|^2}} s_0^\mathsf{T}(s_1 + u_j u_j^\mathsf{T} s_1)\end{pmatrix}.$$

For degenerate ellipses, it returns zero according to rule 2. Algorithm 1 produces our representation of ellipses through vectors $u_j \in \mathbb{R}^2$ and Algorithm 2 solves homogeneous quadratics [Blinn, 2006].

The core problem is sampling of the area between two ellipses within a sector. This part is implemented by Algorithm 3. Our implementation of this algorithm eliminates a few more common subexpressions. For example, `ellipse_area_in_sector()` should utilize w_i and w_o. The normalization of the current direction via

$$\frac{w_n}{|w_{n,\mathrm{x}}| + |w_{n,\mathrm{y}}|}$$

also deserves some attention. In theory, it could be omitted but in floating point arithmetic it is crucial to avoid under- or overflow. Though, there is no need for an exact reciprocal. Our implementation instead flips all exponent bits on the floating-point number $|w_{n,\mathrm{x}}| + |w_{n,\mathrm{y}}|$ (i.e. it does an XOR with 0x7F800000). If the original exponent is $E \in \{-127, \ldots, 128\}$, the new exponent is $1 - E$. Consequently,

Algorithm 1 `ellipse_from_edge`
Input: Vertices $v_j, v_{j+1} \in \mathbb{R}^3$.
Output: A vector $u_j \in \mathbb{R}^2$ such that $C_j := I + u_j u_j^\mathsf{T}$ describes the ellipse that arises from projection of the great circle through v_j, v_{j+1} to the xy-plane.

$n_j := v_j \times v_{j+1}$ *// using Kahan's algorithm for each entry*

$$u_j := \begin{cases} -\frac{1}{n_{j,\mathrm{z}}} n_{j,\mathrm{xy}} & \text{if } \texttt{is\_inner\_ellipse}(n_{j,\mathrm{xy}}), \\ \frac{1}{n_{j,\mathrm{z}}} n_{j,\mathrm{xy}} & \text{otherwise.} \end{cases}$$

if $n_{j,\mathrm{z}} = 0$: $\quad u_{j,\mathrm{x}} := \infty$

return u_j

after this normalization $2 \le |w_{n,\mathrm{x}}| + |w_{n,\mathrm{y}}| \le 8$. The approach fails for denormal numbers but GPUs flush these to zero anyway.

Algorithm 4 implements the sampling procedure for complete polygons. It is split into two parts. The first part needs to run once per polygon, the second part once per sample. The primary purpose of the first part is to pair ellipses with sectors and to compute the areas for each sector. In the decentral case, that includes sorting with the sorting networks described in Supplement A.6. Vertex $v_{j,\mathrm{xy}}$ is less than $v_{k,\mathrm{xy}}$ if $|(v_{j,\mathrm{xy}}, v_{k,\mathrm{xy}})| > 0$. This determinant is evaluated using Kahan's algorithm. If the vertices are equal, the one with the degenerate ellipse comes first (rule 3).

In the central case, the sampling procedure sets

$$w := \cos(2\sqrt{|C_j|}A)\sqrt{|C_j|}s_0 + \sin(2\sqrt{|C_j|}A)RC_j s_0.$$

Except for a constant factor, this is a correct solution because

$$\begin{aligned}
&\begin{pmatrix} -s_0^\mathsf{T} R \\ \frac{1}{\sqrt{|C_j|}} s_0^\mathsf{T} C_j \end{pmatrix} w \\
&= \cos(2\sqrt{|C_j|}A)\sqrt{|C_j|}\begin{pmatrix} 0 \\ \frac{1}{\sqrt{|C_j|}} s_0^\mathsf{T} C_j s_0 \end{pmatrix} + \sin(2\sqrt{|C_j|}A)\begin{pmatrix} -s_0^\mathsf{T} R R C_j s_0 \\ 0 \end{pmatrix} \\
&= s_0^\mathsf{T} C_j s_0 \begin{pmatrix} \sin(2\sqrt{|C_j|}A) \\ \cos(2\sqrt{|C_j|}A) \end{pmatrix}.
\end{aligned}$$

And as explained in the paper, constant factors cancel out.

There is one more ingredient to the efficiency of our implementation, which might be specific to Vulkan or the used GPU. We found that use of atan2 is considerably more expensive than use of its single-parameter counterpart atan. Therefore, we designed every invocation of atan2 such that the passed y-coordinate is non-negative. In presence of rounding errors, clamping to zero is needed. Under these circumstances

$$\mathrm{atan2}(y, x) = \mathrm{atan}\frac{y}{x} + \begin{cases} \pi & \text{if } \frac{y}{x} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Since no offset gets added for small outputs, this formula avoids numeric cancellation.

Table A.2 compares timings of this method to competing techniques when using polygons with six or seven vertices.

Table A.2. Timings in milliseconds for rendering a frame at 1920 × 1080 resolution using 128 samples per pixel. The samples are either taken from 128 different polygonal lights or all from the same light. The baseline timings in the last row, which include access to random numbers, BRDF evaluation, etc., have been subtracted from each timing in the rows above.

	128 lights		128 samples	
Polygon vertex count	6	7	6	7
Area, Turk	8.65	11.4	1.29	1.52
Solid angle, Arvo	13.1	16.4	2.57	2.99
Solid angle, ours	9.45	11.0	2.11	2.37
Solid angle, clipped, ours	12.6	15.5	2.75	3.53
Bilinear, Hart et al.	11.9	13.7	3.20	3.89
Biquadratic, Hart et al.	20.6	24.9	8.46	9.38
Proj., central, Arvo	65.2	75.4	13.4	13.6
Proj., central, ours	17.4	22.6	3.24	3.47
Biased, central, ours	19.0	21.3	2.84	3.42
Proj., decentral, Arvo	100	132	22.7	23.8
Proj., decentral, ours	45.6	59.4	12.4	12.9
Biased, decentral, ours	36.4	45.0	6.23	7.00
+ Baseline	6.71	6.60	3.59	3.65

Algorithm 2 `solve_homogeneous_quadratic`
Input: A matrix $Q \in \mathbb{R}^{2\times 2}$.
Output: A direction $w \in \mathbb{R}^2$ such that $w^\mathsf{T} Q w = 0$.

$$b := \frac{Q_{\mathrm{x,y}} + Q_{\mathrm{y,x}}}{2}$$
$$\Delta := b^2 - Q_{\mathrm{x,x}} Q_{\mathrm{y,y}}$$
if $b \geq 0$: return $\left(|b| + \sqrt{\Delta},\ -Q_{\mathrm{x,x}}\right)^\mathsf{T}$
else: return $\left(Q_{\mathrm{y,y}},\ |b| + \sqrt{\Delta}\right)^\mathsf{T}$

B MULTIPLE IMPORTANCE SAMPLING HEURISTICS

In the paper, we advocate use of clamped optimal MIS but only provide an intuitive justification. The way in which it blends two familiar heuristics is rather obvious. However, our starting point for this formulation is optimal MIS [Kondapaneni et al., 2019]. In the following, we first derive a more efficient variant of optimal MIS using strong assumptions that model our particular setting (Supplement B.1). This estimator has an usual structure but is still unbiased (Supplement B.2). Then we arrive at our clamped optimal MIS by clamping negative MIS weights (Supplement B.3).

B.1 Optimal Multiple Importance Sampling

Our weighted balance heuristic is optimized for situations without occlusion. To incorporate occlusion, we adapt optimal MIS [Kondapaneni et al., 2019]. We still base this derivation on the assumption that LTCs provide perfect fits and that the light source is Lambertian. Thus, the integrand is

$$f(\omega_i) := L_e V(\omega_i) f_r(\omega_o, \omega_i) n^\mathsf{T} \omega_i = V(\omega_i) \sum_{j=0}^{N-1} c_j p_j(\omega_i). \quad \text{(B.1)}$$

Algorithm 3 `sample_sector_between_ellipses`
Input: Uniform random numbers $\xi_0, \xi_1 \in [0, 1)$, $A = \xi_0 A_s$ where A_s is the size of the area being sampled, inner and outer ellipses $u_i, u_o \in \mathbb{R}^2$ and a sector s_0, s_1.
Output: A uniform sample $w \in \mathbb{R}^2$ of the area between the ellipses in the sector.

$$s_0 := \frac{s_0}{\|s_0\|}, \quad s_1 := \frac{s_1}{\|s_1\|}, \quad s_h := s_0 + s_1$$
for $l \in \{i, o\}$ and $j \in \{0, h, 1\}$:
$$\lambda_{l,j} := \frac{1}{\sqrt{\|s_j\|^2 + (u_l^\mathsf{T} s_j)^2}}$$
$$A_{q,0} := \lambda_{o,0}\lambda_{o,h} - \lambda_{i,0}\lambda_{i,h}, \quad A_{q,1} := \lambda_{o,h}\lambda_{o,1} - \lambda_{i,h}\lambda_{i,1}$$
// Select a quad and move its attributes to index 1
$$A_q := (1 - \xi_0)(-A_{q,0}) + \xi_0 A_{q,1}$$
if $A_q \leq 0$: $(s_1, \lambda_{i,1}, \lambda_{o,1}) := (s_0, \lambda_{i,0}, \lambda_{o,0})$
$$A_q := A_q + \begin{cases} A_{q,0} & \text{if } A_q \leq 0, \\ -A_{q,1} & \text{otherwise.} \end{cases}$$
$$A_q := \frac{A_q}{2} ||(s_h, s_1)||$$
for $l \in \{i, o\}$:
$$r_l := \lambda_{l,h} s_h + \lambda_{l,1} s_1, \quad r_l := r_l + u_l u_l^\mathsf{T} r_l, \quad D_l := \lambda_{l,h} r_l^\mathsf{T} s_h$$
$$Q := \lambda_{o,1} D_o R s_1 r_i^\mathsf{T} - (\lambda_{i,1} D_i R s_1 + 2 A_q r_i) r_o^\mathsf{T}$$
$w_n :=$ `solve_homogeneous_quadratic(`Q`)`
// Determine the iteration count and iterate
$$N := \begin{cases} 2 & \text{if } 10^{-5} \leq \xi_0 \leq 1 - 10^{-5}, \\ 0 & \text{otherwise.} \end{cases}$$
for $n \in \{0, \ldots, N-1\}$:
$$w_n := \frac{w_n}{|w_{n,\mathrm{x}}| + |w_{n,\mathrm{y}}|}$$
if $s_h^\mathsf{T} w_n < 0$: $w_n := -w_n$
$$w_i := w_n + u_i u_i^\mathsf{T} w_n, \quad w_o := w_n + u_o u_o^\mathsf{T} w_n$$
$A_\delta := A -$ `ellipse_area_in_sector(`u_o, s_0, w_n`)`
$+$ `ellipse_area_in_sector(`u_i, s_0, w_n`)`
$$T := R w_n (w_i - w_o)^\mathsf{T} - (2 A w_i) w_o^\mathsf{T}$$
$w_n :=$ `solve_homogeneous_quadratic(`T`)`
if $s_h^\mathsf{T} w_n < 0$: $w_n := -w_n$
$$w := \sqrt{(1 - \xi_1) \frac{1}{\|w_n\|^2 + (u_i^\mathsf{T} w_n)^2} + \xi_1 \frac{1}{\|w_n\|^2 + (u_o^\mathsf{T} w_n)^2}}\, w_n$$
return w

Optimal MIS [Kondapaneni et al., 2019] uses a technique matrix

$$A := (A_{j,k})_{j,k=0}^{N-1} := \left(\int_{\mathbb{P}} \frac{p_j(\omega) p_k(\omega)}{\sum_{l=0}^{N-1} p_l(\omega)} \, \mathrm{d}\omega \right)_{j,k=0}^{N-1} \in \mathbb{R}^{N \times N}$$

Algorithm 4 `sample_projected_solid_angle_polygon`
Input: A polygon with vertices $v_0, \dots, v_{m-1} \in \mathbb{R}^3$ subject to the requirements explained in Sec. 3.1 and Supplement A.2, $v_m := v_0$ and uniform random numbers $\xi_0, \xi_1 \in [0, 1)$.
Output: The sampled direction $\omega_i \in \mathbb{S}^2$ and its density with respect to solid angle measure.

// Associate vertices with ellipses (next counterclockwise)
$u_0 :=$ `ellipse_from_edge`(v_0, v_1)
$u_{\text{prev}} := u_0, \quad u_{i,0} := (1, 0)^\mathsf{T}$
for $j = 1, \dots, m-1$:
  $u_{\text{cur}} :=$ `ellipse_from_edge`(v_j, v_{j+1})
  $u_j := \begin{cases} u_{\text{prev}} & \text{if } \texttt{is\_inner}(u_{\text{cur}}), \\ u_{\text{cur}} & \text{otherwise.} \end{cases}$
  if `is_inner`(u_{prev}) and not `is_inner`(u_{cur}) : $u_{i,0} := u_{\text{prev}}$
  $u_{\text{prev}} := u_{\text{cur}}$
$u_{\text{cur}} := u_0$ *// Close the edge loop*
$u_0 := \begin{cases} u_{\text{prev}} & \text{if } \texttt{is\_inner}(u_{\text{cur}}), \\ u_{\text{cur}} & \text{otherwise.} \end{cases}$
if `is_inner`(u_{prev}) and not `is_inner`(u_{cur}) : $u_{i,0} := u_{\text{prev}}$
// Compute the projected solid angle sector by sector
$A_\Sigma := 0$
if not `is_inner`$(u_{i,0})$: *// Central case*
  for $j = 0, \dots, m-1$:
    $A_j :=$ `ellipse_area_in_sector`$(u_j, v_{j,\text{xy}}, v_{j+1,\text{xy}})$
    $A_\Sigma := A_\Sigma + A_j$
else: *// Decentral case*
  sort $(v_{0,\text{xy}}, u_0), \dots, (v_{m-1,\text{xy}}, u_{m-1})$ in place
  $u_i := u_{i,0}$
  for $j = 0, \dots, m-2$:
    if $j > 0$ and `is_inner`(u_j) : $u_i := u_j$
    if $j = 0$ or not `is_inner`(u_j) : $u_o := u_j$
    $A_j :=$ `ellipse_area_in_sector`$(u_o, v_{j,\text{xy}}, v_{j+1,\text{xy}})$
      − `ellipse_area_in_sector`$(u_i, v_{j,\text{xy}}, v_{j+1,\text{xy}})$
    $A_\Sigma := A_\Sigma + A_j$
// continued below

and a contribution vector $b \in \mathbb{R}^N$ with entries

$$b_j := \int_{\mathbb{P}} \frac{p_j(\omega) f(\omega)}{\sum_{l=0}^{N-1} p_l(\omega)} \, \mathrm{d}\omega = \sum_{k=0}^{N-1} c_k \int_{\mathbb{P}} V(\omega) \frac{p_j(\omega) p_k(\omega)}{\sum_{l=0}^{N-1} p_l(\omega)} \, \mathrm{d}\omega.$$

If we were to disregard the visibility term, our special setting would turn b_j into a linear combination of entries in the technique matrix. Since visibility cannot exceed one, we certainly have

$$0 \le b_j \le \sum_{k=0}^{N-1} A_{j,k} c_k.$$

Algorithm 4 continued
// Begin the sampling itself
$A := \xi_0 A_\Sigma$
if not `is_inner`$(u_{i,0})$: *// Central case*
  for $j = 0, \dots, m-1$:
    if $j > 0$: $A := A - A_{j-1}$
    $u_o := u_j, \quad s_0 := v_{j,\text{xy}}$
    if $A < A_j$: break
  $\sqrt{|C_j|} := \sqrt{1 + \|u_o\|^2}$
  $w := \cos(2\sqrt{|C_j|}A)\sqrt{|C_j|}s_0 + \sin(2\sqrt{|C_j|}A)R(s_0 - s_0 u_o u_o^\mathsf{T})$
  $w := \sqrt{\xi_1 \frac{1}{w^\mathsf{T} C_j w}} w$
else: *// Decentral case*
  $u_i := u_{i,0}$
  for $j = 0, \dots, m-2$:
    if $j > 0$: $A := A - A_{j-1}$
    if $j > 0$ and `is_inner`(u_j) : $u_i := u_j$
    if $j = 0$ or not `is_inner`(u_j) : $u_o := u_j$
    $s_0 := v_{j,\text{xy}}, \quad s_1 := v_{j+1,\text{xy}}, \quad A_s := A_j$
    if $A < A_s$: break
  $\xi_0 := \frac{A}{A_s}$
  $w :=$ `sample_sector_between_ellipses`$(\xi_0, \xi_1, A, u_i, u_o, s_0, s_1)$
$\omega_i := \left(\frac{w}{\sqrt{1 - \|w\|^2}} \right)$
return $\omega_i, \quad \frac{\omega_{i,\text{z}}}{A_\Sigma}$

We now assume that visibility affects samples from all techniques equally, i.e. there exists a visibility factor $v \in [0, 1]$ such that $b = vAc$, where $c := (c_0, \dots, c_{N-1})^\mathsf{T}$. When the densities p_0, p_1 are similar, e.g. for distant light sources, that is a good approximation. Otherwise, it may be inaccurate but it is key to the efficiency of our approach and works well in practice.

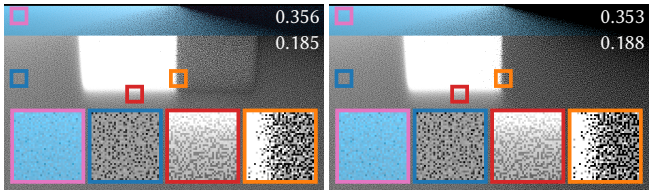
Optimal MIS weights are defined in terms of the coefficient vector $\alpha \in \mathbb{R}^N$ that solves $A\alpha = b = vAc$. Thanks to our assumptions, the solution is simply

$$\alpha = vc.$$

The technique matrix and the contribution vector have dropped out entirely. Thus, we have eliminated the main hurdle to the application of optimal MIS. Normally these coefficients must be learned from a substantial number of samples [Kondapaneni et al., 2019]. Our approach only requires some estimate of average visibility v.

The optimal MIS weights are [Kondapaneni et al., 2019]

$$w_j^o(\omega_i) := \alpha_j \frac{p_j(\omega_i)}{f(\omega_i)} + \frac{p_j(\omega_i)}{\sum_{k=0}^{N-1} p_k(\omega_i)} \left(1 - \frac{\sum_{k=0}^{N-1} \alpha_k p_k(\omega_i)}{f(\omega_i)} \right).$$



(a) Our optimal MIS, $v = 1/2$ (b) Our clamped optimal MIS, $v = 1/2$

Fig. B.1. A blue diffuse plane and a white glossy plane, both lit by a rectangular Lambertian emitter that is partially occluded by a wall. Optimal MIS introduces noise into umbras. We report RMSEs of HDR frames without (top) and with (bottom) clamping of overexposed pixels.

There is an intriguing aspect of optimal MIS that prior work does not discuss. Division by the integrand $f(\omega_i)$ may be division by zero. The proof of optimality [Kondapaneni et al., 2019, Equation (30)] assumes $\frac{f(\omega_i)}{f(\omega_i)} = 1$. Thus, the MIS estimate must be constructed accordingly, even when $f(\omega_i) = 0$. Another way to see this is to replace the value of the integrand by $\varepsilon > 0$ wherever it is zero. For $\varepsilon \to 0$ the integral approaches the intended value and $\frac{f(\omega_i)}{f(\omega_i)} = 1$.

Then the contribution of a sample ω_j from technique p_j is

$$\frac{f(\omega_j)}{p_j(\omega_j)} w_j^o(\omega_j) = v c_j + \frac{1}{\sum_{k=0}^{N-1} p_k(\omega_j)} \left(f(\omega_j) - v \sum_{k=0}^{N-1} c_k p_k(\omega_j) \right).$$

Due to the treatment of division by zero, this contribution may be non-zero when $f(\omega_j) = 0$. The method is unbiased nonetheless because the terms without $f(\omega_j)$ have zero mean (see the next section). In practice, we still discard samples ω_1 below the horizon.

To understand how these zero-mean contributions improve optimality, we consider a case with weak specular shading and a distant light source. We idealize the weak specular contribution by $c_1 = 0$. Since the light source is distant, its solid angle is small and we assume that both sampling densities are practically constant. In particular, $p_0 = p_1$. Then

$$\sum_{j=0}^{1} \frac{f(\omega_j)}{p_j(\omega_j)} w_j^o(\omega_j) = \sum_{j=0}^{1} v c_j + \frac{1}{2}(V(\omega_j) c_0 - v c_0) = \sum_{j=0}^{1} V(\omega_j) \frac{c_0}{2}.$$

Thus, diffuse and specular samples make equal contributions, independent of v, which is the intended outcome.

The visibility factor v, weights down the zero-mean contributions in umbras. In our evaluation, we just set $v = 1/2$ globally. Thus, optimal MIS helps to reduce variance in many spots but introduces zero-mean noise into umbras, which manifests as a sort of light leaking (Fig. B.1a). We recommend use of our optimal MIS only if reasonable estimates of v are available, e.g. from a denoised previous frame.

B.2 Proof of Unbiasedness

Our optimal MIS is a specialization of prior work [Kondapaneni et al., 2019]. In principle, the corresponding proofs apply and show that our method is unbiased as long as the visibility factor v is independent of the current samples. However, we discard samples below the horizon, which could cause problems. Besides, the treatment of division by zero makes it problematic to apply classic proofs [Veach and Guibas, 1995].

Therefore, we now provide a concise proof that our optimal MIS is unbiased. The contribution of a sample ω_j from technique p_j is

$$\frac{f(\omega_j)}{p_j(\omega_j)} w_j^o(\omega_j) = v c_j + \frac{1}{\sum_{k=0}^{N-1} p_k(\omega_j)} \left(f(\omega_j) - v \sum_{k=0}^{N-1} c_k p_k(\omega_j) \right).$$

The term

$$\frac{f(\omega_j)}{\sum_{k=0}^{N-1} p_k(\omega_j)}$$

matches MIS with the balance heuristic. Therefore, it is an unbiased estimate on its own. If the overall estimate is unbiased, the remaining terms

$$v c_j - v \frac{\sum_{k=0}^{N-1} c_k p_k(\omega_j)}{\sum_{k=0}^{N-1} p_k(\omega_j)} \tag{B.2}$$

must have zero mean.

Let $\mathbb{K} \subset \mathbb{S}^2$ be a subset of the unit sphere such that for all $\omega \in \mathbb{K}$

$$\sum_{k=0}^{N-1} p_k(\omega) > 0.$$

If we discard samples ω_j that do not fall into the set $\mathbb{K}$, the expectation of the terms in Equation (B.2) is

$$\begin{aligned}
&\sum_{j=0}^{N-1} \int_{\mathbb{K}} \left(v c_j - v \frac{\sum_{k=0}^{N-1} c_k p_k(\omega)}{\sum_{k=0}^{N-1} p_k(\omega)} \right) p_j(\omega) \, \mathrm{d}\omega \\
=& v \int_{\mathbb{K}} \sum_{j=0}^{N-1} c_j p_j(\omega) - \frac{\sum_{k=0}^{N-1} c_k p_k(\omega)}{\sum_{k=0}^{N-1} p_k(\omega)} \sum_{j=0}^{N-1} p_j(\omega) \, \mathrm{d}\omega \\
=& 0.
\end{aligned}$$

Thus, the expression has zero mean, no matter how we pick the admissible set $\mathbb{K}$ because contributions cancel out in every single point. If our estimate discards samples below the horizon, it is still unbiased.

B.3 Clamped Optimal Multiple Importance Sampling

If no good estimates of the visibility factor v are available, we need a more robust alternative. Through Equation (B.1), we rewrite the contribution of sample ω_j as

$$\frac{f(\omega_j)}{p_j(\omega_j)} w_j^o(\omega_j) = v c_j + \frac{\sum_{k=0}^{N-1} c_k p_k(\omega_j)}{\sum_{k=0}^{N-1} p_k(\omega_j)} (V(\omega_j) - v).$$

Now we separate the contributions for visible samples, where $V(\omega_j) = 1$ and thus $f(\omega_j) \neq 0$, and occluded samples, where $V(\omega_j) = 0$:

$$\frac{f(\omega_j)}{p_j(\omega_j)} w_j^o(\omega_j) = f(\omega_j) \left(\frac{v c_j}{\sum_{k=0}^{N-1} c_k p_k(\omega_j)} + \frac{1 - v}{\sum_{k=0}^{N-1} p_k(\omega_j)} \right) \tag{B.3}$$

$$+ (1 - V(\omega_j)) v \left(c_j - \frac{\sum_{k=0}^{N-1} c_k p_k(\omega_j)}{\sum_{k=0}^{N-1} p_k(\omega_j)} \right). \tag{B.4}$$

The contribution for visible samples in Equation (B.3) matches our clamped optimal MIS. For $v = 0$, we get the standard balance heuristic. For $v = 1$, we use the weighted balance heuristic, which we designed to have zero variance in absence of occluders.

The contribution for occluded samples in Equation (B.4) is the cause of our problems with variance in umbras. We take a pragmatic approach and simply remove this zero mean contribution. Then we are left with our clamped optimal MIS. The MIS weights still sum to one such that the estimate remains unbiased.

Through the clamping, the heuristic is no longer optimal, even under our strong assumptions. However, it blends between two heuristics, which are provably optimal for $v \to 0$ or for fully lit regions. The derivation via the visibility factor v provides a useful interpretation of the blending parameter. In practice, our clamped optimal MIS yields convincing results with $v = 1/2$ (Fig. B.1b). In some spots, variance is slightly greater than with our optimal MIS but variance in umbras is gone.

C SOLID ANGLE SAMPLING OF POLYGONS

In this section, we provide the missing bits to turn the approach for solid angle sampling from the paper into a complete, efficient and robust algorithm.

The paper claims

$$r = \left(G_0 \cos\frac{A}{2} - G_1 \sin\frac{A}{2}\right) v_0 + G_2 \sin\frac{A}{2} v_2,$$

where

$$G_0 = |(v_0, v_1, v_2)|, \qquad G_1 = v_0^\mathsf{T} v_2 + v_1^\mathsf{T} v_2, \qquad G_2 = 1 + v_0^\mathsf{T} v_1.$$

Proof. By definition

$$\begin{aligned} r &= u \times (v_0 \times v_2) \\ &= \cos\frac{A}{2}(v_0 \times v_1) \times (v_0 \times v_2) - \sin\frac{A}{2}(v_0 + v_1) \times (v_0 \times v_2). \end{aligned}$$

The triple product expansion implies

$$\begin{aligned} (v_0 + v_1) \times (v_0 \times v_2) &= (v_0 + v_1)^\mathsf{T} v_2 v_0 - (v_0 + v_1)^\mathsf{T} v_0 v_2 \\ &= G_1 v_0 - G_2 v_2, \\ (v_0 \times v_1) \times (v_0 \times v_2) &= (v_0 \times v_1)^\mathsf{T} v_2 v_0 - (v_0 \times v_1)^\mathsf{T} v_0 v_2 \\ &= G_0 v_0. \end{aligned}$$

Thus,

$$r = \cos\frac{A}{2} G_0 v_0 - \sin\frac{A}{2} G_1 v_0 + \sin\frac{A}{2} G_2 v_2.$$

□

Once we have split the triangle v_0, v_1, v_2 with a new edge connecting v_1, v_2', we need to sample this edge. Our samples are generated radially around v_1. Thus, this problem has a lot in common with solid angle sampling of the unit sphere [Pharr et al., 2016, chapter 13.6.1]. Our method takes the same approach as Arvo's [1995] but we employ minor optimizations.

The dot product of the sampled direction $\omega_i \in \mathbb{S}^2$ with v_1 should be

$$s := (1 - \xi_1) + \xi_1 v_1^\mathsf{T} v_2'.$$

Algorithm 5 `sample_solid_angle_triangle`

Input: Normalized direction vectors to vertices $v_0, v_1, v_2 \in \mathbb{S}^2$ and uniform random numbers $\xi_0, \xi_1 \in [0, 1)$.

Output: The sampled direction $\omega_i \in \mathbb{S}^2$ and its density with respect to solid angle measure.

$$G_0 := ||(v_0, v_1, v_2)||, \quad G_1 := v_0^\mathsf{T} v_2 + v_1^\mathsf{T} v_2, \quad G_2 := 1 + v_0^\mathsf{T} v_1$$

$$A_\Delta := 2\,\mathrm{atan2}\begin{pmatrix} G_0 \\ G_2 + G_1 \end{pmatrix}, \quad A := \xi_0 A_\Delta$$

$$r := \left(G_0 \cos\frac{A}{2} - G_1 \sin\frac{A}{2}\right) v_0 + G_2 \sin\frac{A}{2} v_2$$

$$v_2' := -v_0 + 2\frac{v_0^\mathsf{T} r}{\|r\|^2} r$$

$$s_2' := v_1^\mathsf{T} v_2', \quad s := (1 - \xi_1) + \xi_1 s_2', \quad t' := \sqrt{\frac{1 - s^2}{1 - s_2'^2}}$$

$$\omega_i := (s - t' s_2') v_1 + t' v_2'$$

Return ω_i, A_Δ^{-1}

And of course, it must be located in the plane of v_1 and v_2'. Arvo projects v_2' onto the plane orthogonal to v_1 and normalizes via

$$v_\perp := \frac{v_2' - v_1 v_1^\mathsf{T} v_2'}{\|v_2' - v_1 v_1^\mathsf{T} v_2'\|}.$$

Then the sampled direction is

$$\omega_i = s v_1 + \sqrt{1 - s^2} v_\perp.$$

Our first optimization exploits

$$\begin{aligned} \|v_2' - v_1 v_1^\mathsf{T} v_2'\|^2 &= \|v_2'\|^2 - 2(v_1^\mathsf{T} v_2')^2 + \|v_1 v_1^\mathsf{T} v_2'\|^2 \\ &= 1 - (v_1^\mathsf{T} v_2')^2. \end{aligned}$$

For the second optimization, we defer addition of vectors:

$$\begin{aligned} \omega_i &= s v_1 + \sqrt{\frac{1 - s^2}{1 - (v_1^\mathsf{T} v_2')^2}} (v_2' - v_1 v_1^\mathsf{T} v_2') \\ &= \left(s - v_1^\mathsf{T} v_2' \sqrt{\frac{1 - s^2}{1 - (v_1^\mathsf{T} v_2')^2}}\right) v_1 + \sqrt{\frac{1 - s^2}{1 - (v_1^\mathsf{T} v_2')^2}} v_2'. \end{aligned}$$

Eliminating common subexpressions then gives Algorithm 5.

Note that Algorithm 5 sets G_0 to the absolute value of the determinant, contrary to our previous definitions. van Oosterom and Strackee's formula [1983] for the solid angle of a triangle is designed to give a negative sign when this determinant is negative. By taking the absolute value of the determinant, we effectively also take the absolute value of the solid angle. We observe that

$$-r = \left(-G_0 \cos\left(-\frac{A}{2}\right) - G_1 \sin\left(-\frac{A}{2}\right)\right) v_0 + G_2 \sin\left(-\frac{A}{2}\right) v_2.$$

Thus, the only thing that changes about r is the sign. Then the constructed vertex

$$v_2' = -v_0 + 2\frac{v_0^\mathsf{T} r}{\|r\|^2} r = -v_0 + 2\frac{v_0^\mathsf{T}(-r)}{\|-r\|^2}(-r)$$

does not change at all. We get the same result but do not need to bother with negative signs in several spots, e.g. in the arctangent.

When Algorithm 5 is implemented naively, small triangles will lead to numerical noise. We traced this noise back to cancellations in the 3×3 determinant computation with the Laplace expansion. To overcome it, we construct a single Householder reflection from $v_1 \in \mathbb{S}^2$. Let $\sigma := -1$ if $v_{1,\mathrm{x}} > 0$ and $\sigma := 1$ otherwise. Let $e_0 := (1, 0, 0)^\mathsf{T}$. The Householder reflection is given by the vector

$$h := \frac{v_1 - \sigma e_0}{\|v_1 - \sigma e_0\|} = \frac{v_1 - \sigma e_0}{\sqrt{2(|v_{1,\mathrm{x}}| + 1)}}.$$

It mirrors v_1 onto σe_0. Then

$$\begin{aligned} |G_0| &= ||(v_0, v_1, v_2)|| = ||(I - 2hh^\mathsf{T})(v_0, v_1, v_2)|| \\ &= ||(v_0 - 2h(h^\mathsf{T} v_0), e_0, v_2 - 2h(h^\mathsf{T} v_2))|| \end{aligned}$$

Since one column is e_0, only a 2×2 determinant remains to be computed, which is far more stable. Our code exploits that $v_1^\mathsf{T} v_0$ and $v_2^\mathsf{T} v_0$ are already available to optimize further. In the end, this formulation is only marginally more costly than the Laplace expansion.

In practice, we split Algorithm 5 into a part that is executed once per triangle and another part that is executed once per sample. Besides, we support complete triangle fans.

REFERENCES

James Arvo. 1995. Stratified Sampling of Spherical Triangles. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, 437–438. https://doi.org/10.1145/218380.218500

James Arvo. 2001. Stratified sampling of 2-manifolds. In *State of the Art in Monte Carlo Ray Tracing for Realistic Image Synthesis (SIGGRAPH 2001 Course Notes)*. ACM.

James F. Blinn. 2006. How to solve a quadratic equation. Part 2. *IEEE Computer Graphics and Applications* 26, 2 (2006), 82–87. https://doi.org/10.1109/MCG.2006.35

David Hart, Matt Pharr, Thomas Müller, Ward Lopes, Morgan McGuire, and Peter Shirley. 2020. Practical Product Sampling by Fitting and Composing Warps. *Computer Graphics Forum (proc. EGSR)* 39, 4 (2020). https://doi.org/10.1111/cgf.14060

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Trans. Graph. (proc. SIGGRAPH)* 38, 4, Article 37 (2019). https://doi.org/10.1145/3306346.3323009

Morgan McGuire. 2011. Efficient Triangle and Quadrilateral Clipping Within Shaders. *Journal of Graphics, GPU, and Game Tools* 15, 4 (2011), 216–224. https://doi.org/10.1080/2151237X.2011.619891

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering, 3rd Edition*. Morgan Kaufmann. http://www.pbr-book.org

Greg Turk. 1992. Generating random points in triangles. In *Graphics Gems*, Andrew S. Glassner (Ed.). Academic Press, 24–28.

Adriaan van Oosterom and Jan Strackee. 1983. The Solid Angle of a Plane Triangle. *IEEE Transactions on Biomedical Engineering* 30, 2 (1983), 125–126. https://doi.org/10.1109/TBME.1983.325207

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, 419–428. https://doi.org/10.1145/218380.218498