Per-Vertex Defocus Blur for Stochastic Rasterization

Jacob Munkberg[1] Robert Toth[1] Tomas Akenine-Möller[1,2]

[1]Intel Corporation [2]Lund University

Abstract

We present user-controllable and plausible defocus blur for a stochastic rasterizer. We modify circle of confusion coefficients per vertex to express more general defocus blur, and show how the method can be applied to limit the foreground blur, extend the in-focus range, simulate tilt-shift photography, and specify per-object defocus blur. Furthermore, with two simplifying assumptions, we show that existing triangle coverage tests and tile culling tests can be used with very modest modifications. Our solution is temporally stable and handles simultaneous motion blur and depth of field.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction

Digital artists and game developers often want non-physically-based control over depth of field (DOF) parameters. Examples include the possibility to limit foreground blur or to extend the in-focus range while preserving the foreground and background blur. With post-processed DOF approaches [PC81, Dem04, EH07], this is straightforward since the blur filter is expressed as a user-provided function of the scene depth. Kosara et al. [KMH01] introduce the term *semantic depth of field*, where the amount of DOF is controlled on a per-object basis. Their implementation use a set of depth-sorted layers or billboards, which are individually blurred and composited into a final image. However, visibility is never resolved correctly using the above methods, and consequently, their use is limited.

A physically-based camera model can be simulated with distributed ray tracing through a set of lenses and apertures [KMH95]. Kosloff and Barsky [KB09] use non-linear ray tracing (by tracing bent rays) to simulate non-physical effects such as multiple focus planes. Each pixel may have a unique ray distribution. In general, *each point in 3D space* can have a different blur value. They also present two other approaches for generalized DOF: a heat diffusion solver on a layered depth image and a light field filtering technique.

Lee et al.'s work [LES10] on flexible defocus blur allows for varying lens parameters *for each pixel* to emulate effects like tilt-shift photography, curvature of confusion, and lens aberrations. Visibility is resolved by ray tracing through a layered depth image. User-provided camera space DOF constraints are interpolated using a least-squares fit to derive varying per-pixel lens parameters. We refer to Lee et al.'s paper for an overview of lens effects and a more complete survey of controllable defocus techniques. It is far from straightforward to extend these ray tracing approaches to handle simultaneous motion blur and depth of field.

State-of-the-art stochastic rasterizers, e.g., RenderMan [AG00], have programmable surface and displacement shading, and extensive *global* camera control, where the aperture shape and density can be adjusted. To the best of our knowledge, however, there is no publicly known flexible DOF approach for stochastic rasterizers with *local* control, such that the blur amount can be controlled per-vertex.

Our goal is flexible DOF that works directly in a stochastic rasterizer without any layer-generation passes, iterative solvers, and with minimal impact on pipeline design and rendering optimizations. Specifically, we want to explore how user-defined DOF control can be integrated with efficient tile-based traversal for higher-dimensional rasterization [LAKL11, AMTMH12, MAM12].

We let the user specify circle of confusion parameters for every vertex in a mesh. This enables artistic control of defocus blur, while still supporting the standard thin lens model as a subset. In addition, we introduce two simplifying assumptions that makes this approach compatible with state-of-the-art per-tile tests and coverage tests in 4D and 5D rasterizers, and handles *simultaneous* motion blur and DOF. The streaming nature of rasterization also enables interesting features, such as per-triangle DOF control and reduced DOF for specific depth ranges, which are not easily supported by ray tracing-based approaches. Finally, we show that our model is also a useful tool to reduce performance variations for scenes rendered with defocus blur.

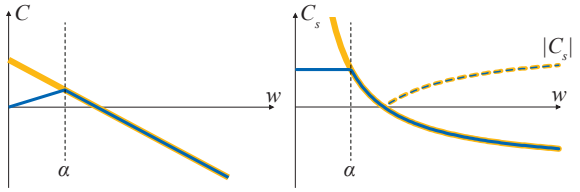


Figure 1: *Circle of confusion radius in clip space (left) and screen space (right). The bright yellow curves show the functions obtained by the thin lens model. The dark blue curves show the functions obtained after limiting the foreground blur.*

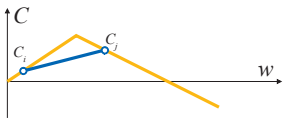


Figure 2: *Linearization errors are introduced for triangles that span discontinuities in C.*

2. Algorithm

Let us first revisit the standard assumptions of a stochastic rasterizer. A clip-space vertex of a triangle is denoted $\mathbf{p} = (x, y, w)$. In the thin lens model, the signed clip-space circle of confusion radius, C, for $\mathbf{p}$ is a linear function of the vertex depth, w, i.e., $C(w) = a + wb$, where a and b are constants derived from the camera's aperture size and focal distance. The clip space vertex position for a specific lens position, (u, v), is given by: $\mathbf{p}'(u, v) = \mathbf{p} + C(w)(u, k\,v, 0)$, which is a shear in clip space x and y. The scalar, k, adjusts for non-square aspect ratios, and the focus plane is located at $w = F$.

The screen space blur radius, $C_s = C/w$, goes towards infinity for $w \to 0$, so a small out-of-focus object very close to the camera can easily cover the entire screen. This is in contrast to most post-processing DOF methods, where the screen space defocus blur is limited to a certain max radius, which is physically incorrect, but has more predictable performance.

Example As an illustrative example, we limit the screen space blur, $C_s(w) = (a + wb)/w$, to a certain radius, R, in the foreground region ($w < F$). We see that $C_s(w) = R$ for the depth $w = \frac{a}{R-b}$. The threshold is denoted $\alpha = \frac{a}{R-b}$. This leads to the following definitions:

$$C_s = \begin{cases} \frac{a+wb}{w}, & w > \alpha, \\ R & w \leq \alpha, \end{cases} \tag{1}$$

which is equivalent to the following formula in clip space:

$$C = \begin{cases} a + wb, & w > \alpha, \\ wR & w \leq \alpha. \end{cases} \tag{2}$$

Equations 1 and 2 are illustrated in Figure 1.

C is a piecewise linear function of the vertex depth, and the thin lens property that C can be linearly interpolated over the triangle does not hold. Therefore, we introduce the assumption that *the clip space circle of confusion radius is linear in the interior of a triangle*. With this assumption, we can compute C in the vertex shader using Equation 2 and provide these coefficients instead of the global camera parameters when positioning and coverage-testing the triangle for a certain lens position. The stochastic rasterizer is essentially unmodified, and the amount of defocus blur is now an attribute that can be controlled on vertex granularity for added artistic control.

Any triangle straddling the discontinuity in C will have a somewhat different circle of confusion in the interior than what is described by the piecewise linear C function, as illustrated in Figure 2. This effect is most visible on very large triangles, and is negligible in a micropolygon pipeline. Also, geometry with T-junctions may show cracks with this assumption. We see this as yet another reason to completely avoid T-junctions.

2.1. Controllable Defocus Blur in 5D Rasterization

In this section, we generalize the user-controllable DOF to 5D stochastic rasterization, where the fifth dimension is shutter time. We assume linear vertex motion in clip space, i.e., $\mathbf{p}(t) = (1-t)\mathbf{q} + t\mathbf{r}$. With the thin lens model, the signed clip space circle of confusion radius, C, of a moving vertex, $\mathbf{p}(t)$, is a linear function in t: $C(t) = C(w(t)) = a + w(t)b$. The clip space vertex position for a time, t, and lens position, uv, is given by:

$$\mathbf{p}'(u, v, t) = \mathbf{p}(t) + C(t)(u, k\,v, 0). \tag{3}$$

If C is replaced by a non-linear function in t, efficient triangle coverage tests (per tile and per sample) in 5D no longer work. We therefore introduce another simplification, namely that *the clip space circle of confusion radius of a vertex is a linear function in time within the frame*. The user provides two circle of confusion coefficients for each vertex per frame, namely c_0 at $t = 0$ and c_1 at $t = 1$. During rasterization, we use $C(t) = (1-t)c_0 + tc_1$ to determine the triangle's position for a time, t, and lens position, uv, in Equation 3. If the user provides the standard thin lens coefficients, e.g., $c_0 = a + b\,w(0)$ and $c_1 = a + b\,w(1)$, the rendered image is identical to that of a standard 5D rasterizer.

3. Pipeline Modifications

In this section, we outline pipeline differences between a rasterizer where the clip space circle of confusion, C, is a user-provided vertex attribute compared to a rasterizer that assumes a thin lens model.

The sample coverage test of a stochastic rasterizer can be slightly optimized if C is a linear function of w. Compared to a test that handles arbitrary C-coefficients, the difference

in arithmetic cost is 16 vs. 18 FMA (fused-multiply add) operations for a DOF coverage test. For a 5D coverage test, the difference is 25 vs. 30 FMA operations [LK11a].

Some stochastic rasterizers also perform coverage tests in 5D per tile to quickly discard samples that cannot hit the triangle. It is easy to verify that the tile culling tests from previous work [LAKL11,AMTMH12,MAM12] still work as long as C varies linearly over the triangle and varies linearly in t within the frame, which are exactly the two assumptions we introduced in Section 2. The same holds for view-frustum culling and screen space bounding. For example, Laine and Karras' screen space bound algorithm [LK11b] works out of the box with user-provided C-coefficients.

A backface culling test for 5D rasterization [MAM11] can be optimized for the thin lens model. We show the expression for a generalized backface test that supports user-provided C-coefficients in Appendix A and compare it to the thin-lens version.

4. Results

For all images in this section, we have simply averaged all samples within a pixel. No motion or defocus-aware reconstruction filters were used. Also, we deliberately use high-contrast shading to more easily see the effect of the blur.

With arbitrary circle of confusion parameters, it is possible to accomplish a wide variety of defocus effects. Figure 3 shows frames from a sequence where the foreground blur is reduced (Section 2). Figure 4 shows reduced foreground blur in a scene with simultaneous motion blur and DOF (Section 2.1). In our experience, the transition from large blur to no foreground blur looks plausible. The in-focus range can be extended by another piecewise linear $C(w)$ function, as shown in Figure 5.

In general, C is replaced with an arbitrary user-provided function, expressed as a scalar vertex attribute, which gives artistic freedom. We achieve a set of highly non-physical defocus blurs by computing custom per-vertex defocus radii, C. For example, we can modify the orientation and shape of the focus surface to approximate effects such as tilt-shift photography & curvature of field [LES10]. Given a vertex $\mathbf{p} = (x, y, w)$ we can increase the blur amount radially from the center with $C = (a + bw)(1 + x^2 + y^2)$. Figure 6 shows an example of a tilted focus surface on a scene with very complex visibility. In this example, we set $C = a + b(w + x)$.

It is important to highlight that per-vertex DOF control as presented here is orthogonal to the sample distribution in *xyuvt* space. In Figure 8, we show two different sample distributions over the lens, combined with controlled DOF where we have disabled DOF (C is set to zero) for some objects in the scene. Please refer to the accompanying video for more examples with animated focus parameters, moving objects and camera, and simultaneous depth of field and motion blur.

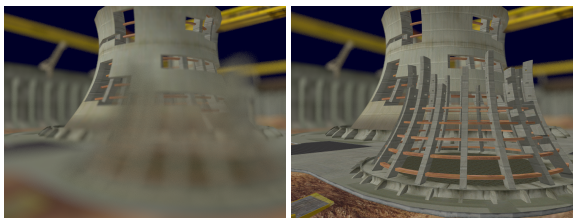


Figure 3: *Removed foreground blur on a scene with depth of field, while leaving the background blur intact. Example scene from the Microsoft DirectX SDK.*

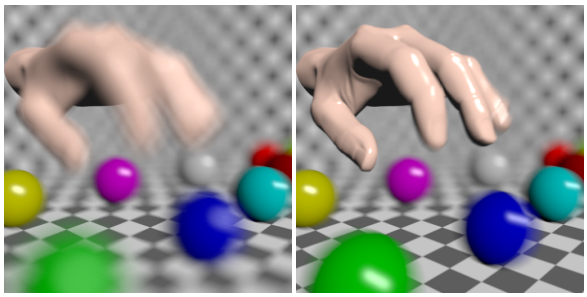


Figure 4: *Removed foreground defocus blur on a scene with both motion blur and DOF, while leaving the background blur intact. In our software implementation, the right image renders 3× faster. Note that the motion blur is still present. The animated hand model is from the Utah 3D Animation Repository.*

In real-time applications, a moving camera may (accidentally) move very close to an object. As illustrated in Figure 1, the screen space blur is very large for objects close to the camera, and the performance impact of rasterizing these heavily defocused objects may be drastic. By bounding the foreground blur, the performance variations can be reduced. Figure 7 illustrates this use case, where a bounded foreground blur results in a $4-9\times$ performance increase.

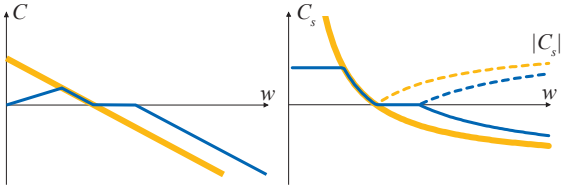
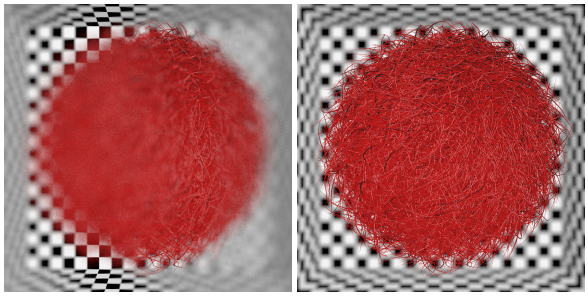
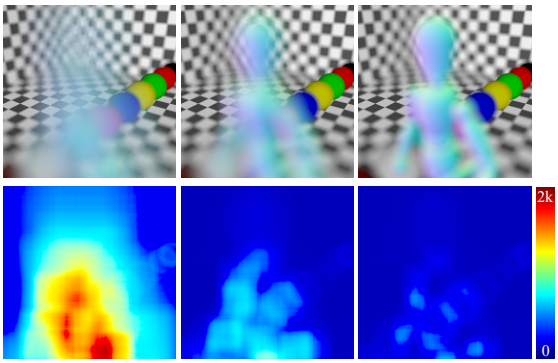


Figure 5: *Circle of confusion radius in clip space (left) and screen space (right). The bright yellow curves show the functions obtained by the thin lens model. The dark blue curves show the functions obtained after limiting the foreground blur and extending the depth of field to a larger depth range.*



tilted focal surface no foreground blur

Figure 6: *Examples with complex visibility. Hairball model from Samuli Laine.*

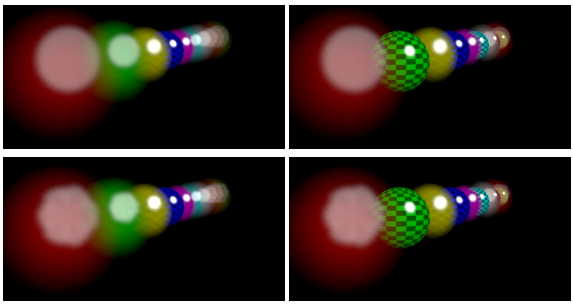


Reference (1.0) Large bound (0.24) Small bound (0.11)

Figure 7: *Defocused foreground triangles cover large screen space regions. We visualize the number of triangles processed per pixel as heat maps. The numbers in parentheses indicate relative render times compared to the reference in a software stochastic rasterizer. Bounding the foreground blur reduces the work substantially. Animated wood doll from the Utah 3D Animation Repository.*

4.1. Discussion

In a rasterizer, the per-vertex circle of confusion parameter, C, simply indicates how much an object should be sheared for a specific lens coordinate, and is an object-specific parameter. In a ray tracer, however, the geometry is expressed in a global coordinate system, and individual shears per object mean that the distribution and angles of rays from the camera lens are unique for each triangle. This would require sending "bent rays" through the scene, which makes the ray traversal step much more complicated. See Kosloff and Barsky's work [KB09] for more details of non-linear distributed ray tracing. Some special cases, like limiting foreground blur can be handled as shown in Figure 9, using a unique lens per screen space sample, that is inserted in the scene. This is cumbersome, and is very likely to completely disable certain traversal optimizations for primary rays. A stochastic rasterizer, on the other hand, handles these cases easily with our algorithm, with none or very modest performance implications.



reference flexible

Figure 8: *Our flexible DOF approach is orthogonal to the lens sample distribution. Here is an example with circular (top) & hexagonal (bottom) lens shape combined with per-vertex control of the blur amount.*

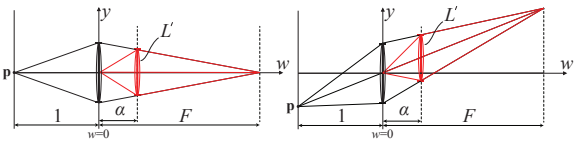
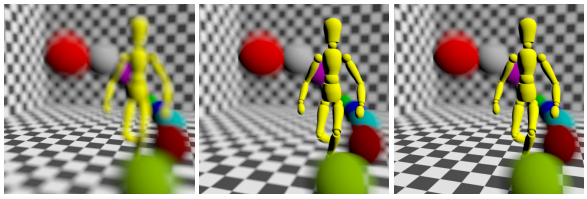


Figure 9: *Limiting foreground blur in a ray tracer. The global lens is replaced by a per-pixel lens, L' (red), inserted at depth $w = \alpha$, centered around the ray through the pixel and the pinhole camera. Its focal length, $f = \frac{F}{\alpha(F-\alpha)}$, is chosen such that the blur in the region $w > \alpha$ is unaffected. Rays are now traced from a pinhole camera at $w = 0$ towards the lens and each live ray is refracted in the lens. This is shown for two different points, $\mathbf{p}$, on the image plane.*

On the other hand, in a rasterizer designed around the thin-lens model, it is hard to simulate effects that vary as functions of the position on the lens, such as general lens aberrations. Our extension with user-provided C-coefficients per vertex only modifies the amount the vertex is sheared in x and y in clip space, and is assumed to be a linear function in u and v, where the shears in x and y are independent. That said, the rasterizer could be extended with more defocus parameters per vertex, defining the coefficients of a function $C = f(u,v,t)$ to support more elaborate lens models. This would, however, complicate the coverage and cull tests significantly. Note also that with arbitrary C parameters, the integral of the view-dependent shading contribution from all lens samples may differ from the thin-lens equivalent in out-of-focus regions.

A difficult scenario for user-defined blur per-vertex is a large triangle spanning the focus plane. When gradually reducing the foreground blur, the focus distance appears to move. This is an effect of the assumption that the defocus blur varies linearly over the triangle (e.g., Figure 2). With higher tessellation rates, this effect is hardly noticeable, as shown in Figure 10.

It is important to note that it is hard to guarantee an upper



reference coarse floor tessellated floor

Figure 10: *With coarse tessellation, the linearization assumption gives visible artifacts when the CoCs deviated from the thin-lens model. The focus distance moves due to the coarse tessellation (2 triangles) of the ground floor. With a higher tessellation rate, this effect is hardly visible.*

bound, R, for defocus in all cases. For triangles straddling *both* $w = 0$ and $w = F$, the screen space blur radius is not bounded by R as $C(w = 0) \neq 0$. However, with reasonably tessellated scenes, our approach has the potential to greatly improve the average rasterization performance.

Requirements for flexible defocus blur may also influence future API designs for stochastic rasterizers. We would like to see an output variable from the vertex shader (OpenGL/Direct3D) or surface shader (RenderMan) representing the defocus coefficients. The user then has the option to override the global camera parameters when the primitive is bounded, positioned, and sampled.

We leave a detailed study of the combination of flexible defocus and reconstruction filters for future work.

Acknowledgements The authors thank Aaron Coday, Tom Piazza, Charles Lingle, and Aaron Lefohn for supporting this research. The Advanced Rendering Technology team at Intel provided valuable feedback. Tomas Akenine-Möller is a Royal Swedish Academy of Sciences Research Fellow supported by a grant from the Knut and Alice Wallenberg Foundation. In addition, we acknowledge support from the Swedish Foundation for Strategic Research.

References

[AG00] APODACA A. A., GRITZ L.: *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 2000. 1

[AMTMH12] AKENINE-MÖLLER T., TOTH R., MUNKBERG J., HASSELGREN J.: Efficient Depth of Field Rasterization using a Tile Test based on Half-Space Culling. *Computer Graphics Forum, 31*, 1 (2012), 3–18. 1, 3

[Dem04] DEMERS J.: Depth of Field: A Survey of Techniques. In *GPU Gems*, Fernando R., (Ed.). Addison-Wesley, 2004, pp. 375–390. 1

[EH07] EARL HAMMON J.: Practical Post-Process Depth of Field. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2007, chapter 28, pp. 583–606. 1

[KB09] KOSLOFF T. J., BARSKY B. A.: Three Techniques for Rendering Generalized Depth of Field Effects. In *Proceedings of the Fourth SIAM Conference on Mathematics for Industry: Challenges and Frontiers (MI09)* (October 2009), pp. 42–48. 1, 4

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A Realistic Camera Model for Computer Graphics. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (1995), pp. 317–324. 1

[KMH01] KOSARA R., MIKSCH S., HAUSER H.: Semantic Depth of Field. In *Proc. IEEE Information Visualization* (2001), pp. 97–104. 1

[LAKL11] LAINE S., AILA T., KARRAS T., LEHTINEN J.: Clipless Dual-Space Bounds for Faster Stochastic Rasterization. *ACM Transactions on Graphics, 30*, 4 (2011), 106:1–106:6. 1, 3

[LES10] LEE S., EISEMANN E., SEIDEL H.-P.: Real-Time Lens Blur Effects and Focus Control. *ACM Transactions on Graphics, 29*, 4 (2010), 65:1–7. 1, 3

[LK11a] LAINE S., KARRAS T.: *Efficient Triangle Coverage Tests for Stochastic Rasterization*. Technical Report NVR-2011-003, NVIDIA, Sept. 2011. 3

[LK11b] LAINE S., KARRAS T.: *Improved Dual-Space Bounds for Simultaneous Motion and Defocus Blur*. Technical Report NVR-2011-004, NVIDIA, Nov. 2011. 3

[MAM11] MUNKBERG J., AKENINE-MÖLLER T.: Backface Culling for Motion Blur and Depth of Field. *journal of graphics, gpu, and game tools 15*, 2 (2011), 123–139. 3, 5

[MAM12] MUNKBERG J., AKENINE-MÖLLER T.: Hyperplane Culling for Stochastic Rasterization. In *Eurographics 2012 - Short Papers* (2012), pp. 105–108. 1, 3

[PC81] POTMESIL M., CHAKRAVARTY I.: A Lens and Aperture Camera Model for Synthetic Image Generation. In *Computer Graphics (Proceedings of SIGGRAPH 81)* (1981), pp. 297–305. 1

Appendix A - 5D Backface Test for Flexible Blur

A backface culling test for DOF [MAM11] is expressed as a determinant test:

$$\mathbf{p}'_0(u,v) \cdot (\mathbf{p}'_1(u,v) \times \mathbf{p}'_2(u,v)) > 0, \tag{4}$$

where $\mathbf{p}'_i(u,v) = \mathbf{p}_i + C_i(u, kv, 0)$. This can be expanded to:

$$\mathbf{p}_0 \cdot (\mathbf{p}_1 \times \mathbf{p}_2) + (u, k\,v, 0) \cdot (C_0\mathbf{p}_1 \times \mathbf{p}_2 + C_1\mathbf{p}_2 \times \mathbf{p}_0 + C_2\mathbf{p}_0 \times \mathbf{p}_1) > 0. \tag{5}$$

If C follows the thin lens model, e.g., $C_i = a + bw_i$, the expression can be simplified to:

$$\mathbf{p}_0 \cdot (\mathbf{p}_1 \times \mathbf{p}_2) + a(u, kv, 0) \cdot (\mathbf{p}_0 \times \mathbf{p}_1 + \mathbf{p}_1 \times \mathbf{p}_2 + \mathbf{p}_2 \times \mathbf{p}_0) > 0. \tag{6}$$

For user-provided C-coefficients, this optimization must of course be disabled.

A similar argument holds for the 5D backface test. The 5D version of Equation 5 contains terms on the form:

$$\mathbf{p}_0(t) \cdot (\mathbf{p}_1(t) \times \mathbf{p}_2(t)) \quad \text{and} \quad (u, k\,v, 0) \cdot (C_m(t)\mathbf{p}_n(t) \times \mathbf{p}_o(t)), \tag{7}$$

where m, n, o are indices in $\{0, 1, 2\}$ and each $C_m = (1-t)c_0 + tc_1$ is a linear function in t determined by user-provided vertex attributes c_0 and c_1. Thus, Equation 7 contains t^3, ut^3, and vt^3 terms, which must be bounded over the lens and shutter interval to conservatively cull a triangle.

The thin lens version of the 5D backface test has simplified terms (c.f., Equation 6):

$$\mathbf{p}_0(t) \cdot (\mathbf{p}_1(t) \times \mathbf{p}_2(t)) \quad \text{and} \quad a(u, k\,v, 0) \cdot (\mathbf{p}_n(t) \times \mathbf{p}_o(t)). \tag{8}$$

which have lower order polynomial coefficients t^3, ut^2 and vt^2. Thus, a backface test with user-provided C-coefficients is slightly harder to bound.