Area Lights in Signed Distance Function Scenes

Róbert Bán , Csaba Bálint , Gábor Valasek

Eötvös Loránd University, Hungary
{bundas,csabix,valasek}@inf.elte.hu

Abstract

This paper presents two algorithms to incorporate spherical and general area lights into scenes defined by signed distance functions. The first algorithm employs an efficient approximation to the contribution of spherical lights to direct illumination and renders them at real-time rates. The second algorithm is of superior quality at a higher computational cost which is better suited for interactive rates. Our results are compared to both real-time soft shadow algorithms and a ground truth obtained by Monte Carlo integration. We show in these comparisons that our real-time solution computes more accurate shadows while the more demanding variant outperforms Monte Carlo integration at the expense of accuracy.

CCS Concepts

• ***Computing methodologies*** → *Ray tracing; Computer graphics;*

1. Introduction and Related Work

Implicit functions are a convenient representation for surfaces in many aspects. For example, they provide simple means to express the result of operations such as union and intersection [RMD11]. Modeling surfaces with signed distance functions (SDFs) offers the same advantages and, additionally, several specialized algorithms have been proposed [Har96, KSK*14, BV18] to render them efficiently. This makes this representation viable even in real-time applications, such as computer games [Aal18]. We aim to demonstrate that the global information stored in distance functions can be also used to speed up the rendering of advanced graphical effects.

This paper shows a novel solution to incorporate area lights and their soft shadows into scenes represented by SDFs. We propose two approximate solutions to compute the exitant radiance at a surface point from the direct illumination of an area light. These approximations posses different accuracy and performance characteristics. The following three algorithms are the contributions of this paper:

First, we present a single ray cone tracing method in Section 3. It approximates the closest point of the cone axis to the geometries of the scene. It is achieved by leveraging the fact that our scenes are composed of signed distance functions. To the best of our knowledge, occlusion in a cone has not been approximated for SDF scenes using a single ray before.

Second, we show a fast soft-shadow algorithm for spherical light sources and constrained occlusion configurations in Section 4.1. This approach is suitable for real-time applications.

Finally, in Section 4.2, we present an algorithm that offers higher quality and also extends to light sources of arbitrary shapes. Although slower than our first method, it can be used to accelerate Monte Carlo integrations e.g. when constructing ground truths, at the expense of accuracy.

Real-time solutions for area lights have been proposed by [ABB18, EL18] based on Monte Carlo integration recently. These rely on a small number of rays per pixel and a denoising step to mitigate the artifacts caused by low sample counts. They also used the hardware accelerated ray tracing capabilities of the new generation of NVIDIA GPUs. As such, they do not restrict the represention of geometries; however, their results are only applied to scenes consisting of triangles. In comparison, our first algorithm uses a single ray and does not require denoising. As a trade-off, it is only applicable to signed distance functions or estimates thereof and spherical light sources.

Quílez's algorithm, communicated on his website [Qui10], is a soft shadow algorithm that uses distance fields. Aaltonen has improved on it in [Aal18]. This algorithm shares the most similarities with ours, but they restrict themselves to unsigned distances, essentially halving the penumbras they generate. We allow signed distances and improve on quality at the expense of a minor performance drop in case of our first algorithm.

We provide an empirical comparison of our proposed approximations with the exact evaluation of the exitant radiance integral, as well as with a ground truth obtained by Monte Carlo integration. We also include Quílez's algorithm in our tests [Qui10, Aal18].

DOI: 10.2312/egs.20191021

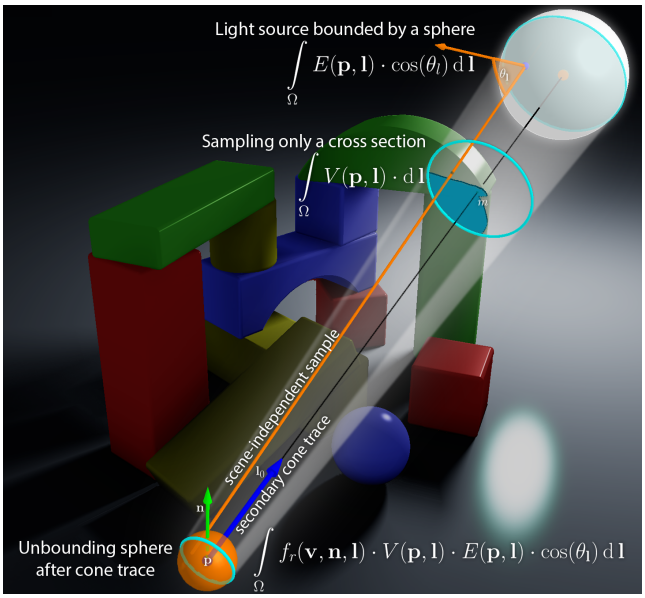


Figure 1: *The orange unbounding sphere is the result of the primary cone trace. From here, a cone is traced towards the white spherical light source. We approximate the ratio of the occluded and total incoming radiance within this by finding the minimum signed distance of the cone to the geometries of the scene, using Algorithm 2. The scene used for this illustration runs at 60 FPS on a GeForce 1080 Ti.*

2. Overview of Tracing Signed Distance Functions

This section reviews the results from the literature on rendering scenes consisting of geometries represented by signed distance functions.

The mapping $f:\mathbb{R}^3\to\mathbb{R}$ is a *signed distance function* (SDF) if $|f(\boldsymbol{p})|$ is the distance of $\boldsymbol{p}\in\mathbb{R}^3$ to the represented surface and $f(\boldsymbol{p})<0$ if $\boldsymbol{p}$ is inside and $f(\boldsymbol{p})>0$ if it is on the outside [Har96].

An *unbounding sphere* at point $\boldsymbol{p}\in\mathbb{R}^3$ is the sphere of radius $|f(\boldsymbol{p})|$ centered at $\boldsymbol{p}$. Since the unbounding sphere is disjoint from the boundary of the object, travelling $|f(\boldsymbol{p})|$ along the ray is guaranteed not to overstep the surface. This observation gave birth to sphere tracing [Har96], shown in Algorithm 1. Generally, surfaces are defined by signed distance lower bounds, but we strive to produce accurate distance functions because both rendering speed and shading accuracy depend on it.

In : $\boldsymbol{p},\boldsymbol{v}\in\mathbb{R}^3, |\boldsymbol{v}|=1$ ray, $f:\mathbb{R}^3\to\mathbb{R}$ SDF estimate
Out: $t\in[0,+\infty)$ distance traveled along the ray
$t:=0;\quad i:=0;$
for $i<i_{max}$ ***and*** $f(\boldsymbol{p}+t\cdot\boldsymbol{v})$ *not too small;* $i:=i+1$; **do**
  $t:=t+f(\boldsymbol{p}+t\cdot\boldsymbol{v})$
end

Algorithm 1: *Basic sphere tracing adapted from [Har96].*

Cone tracing defines a cone around the ray starting at $\boldsymbol{p}$ toward direction $\boldsymbol{v}$ with radius r at $\boldsymbol{p}$ and half opening angle α. An unsafe method for finding the first cone-surface intersection is changing the loop condition in Algorithm 1 to $f(\boldsymbol{p}+t\cdot\boldsymbol{v})-r-t\cdot\tan\alpha$. To improve on this, the authors in [BV18] proposed a method that converges to the cone-surface intersection by dividing the condition by its Lipschitz constant and thus taking $F(t):=\frac{f(\boldsymbol{p}+t\cdot\boldsymbol{v})-r-t\cdot\tan\alpha}{1+|\tan\alpha|}$ sized steps along the ray. Note that r and α may even be negative numbers resulting in an inverted cone that stops inside the surface.

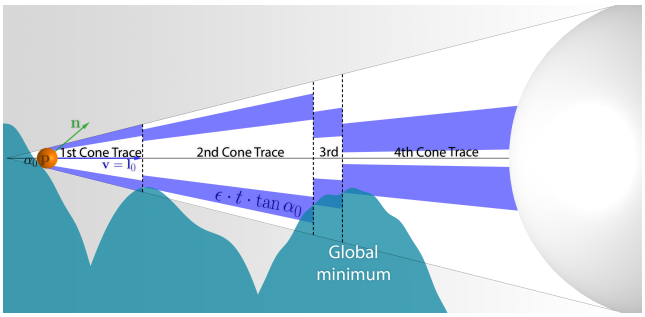


Figure 2: *A cross section of a run of the minimal tangent cone search from Algorithm 2 with* $\varepsilon=0.3$.

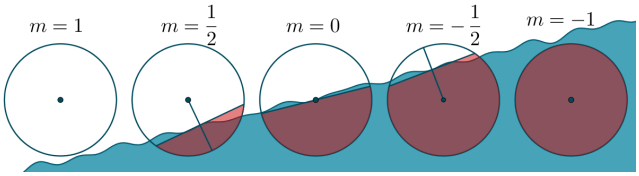


Figure 3: *Cross sections of cones with various occlusion values.*

3. Minimal Tangent Cone Search

This algorithm is used to approximate the ratio of the occluded and total incident light along a cone. This measure is used to establish a proxy occluder. We model the soft shadow after the latter. As such, it is used to estimate the visibility term in the exitant radiance integral.

We construct a single shadow cone to each light from the reported ray-surface intersection $\boldsymbol{p}\in\mathbb{R}^3$. The cone is determined by constraining it to be tangential to the unbounding sphere around $\boldsymbol{p}$ and the bounding sphere of the area light, as illustrated in Figure 1. Sphere tracing along the axis of this cone can be used to find the minimum distance between the axis and the surfaces in the scene.

Our proposed Algorithm 2 is parameterized by an $\varepsilon>0$ scalar that controls the accuracy of the computed shadow. It measures how much the minimizer found by the algorithm deviates from the true global minimum along the cone axis.

More formally, we are minimizing $m(t)=\frac{f(\boldsymbol{p}+t\cdot\boldsymbol{v})}{r_0+t\cdot\tan\alpha_0}$ along the cone. If $m(t)=1$, the surface is touching the cone. At $m(t)=0$, the surface is tangent to the cone axis. If $m(t)=-1$, the surface is touching the inverted cone with radius $-r_0$ and angle $-\alpha_0$. See Figure 3 for a geometric interpretation. For calculating soft shadows, our sole interest is finding the minimum of m. We clamp the result to -1 and 1.

In addition, we also use the cone, defined by its half opening angle α_0 and radius r_0 at $\boldsymbol{p}$, to pocket the currently found minimum. If we find an intersection between the cone and the scene while sphere tracing along the cone axis, we shrink the cone to accommodate this new minimum distance: the new cone will not intersect

the scene geometries at the current point. That is, there are no occluders within the new cone between the starting point $\boldsymbol{p}$ and the current point. Radiance coming within this cone is not blocked. The ratio of the radii of the new and the original cones will be $m(t)-\varepsilon$; therefore we cannot skip a local minimum differing from the current distance value more than by the ε parameter. In practice, $\varepsilon \in [0.05, 0.1]$ proved to be a good trade-off between speed and accuracy. In general, larger light sources required smaller ε values.

In : $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^3, |\boldsymbol{v}| = 1$ ray;
$f : \mathbb{R}^3 \to \mathbb{R}$ SDF estimate;
r_0, α_0: cone radius at $\boldsymbol{p}$ and half opening angle;
$\varepsilon > 0$: maximum error allowed
Out: $m \in [-1, 1]$ minimal relative cone radius
$t := 0 \quad m := 1 \quad r := r_0 \quad \tan\alpha := \tan\alpha_0$
for $t < t_{max}$ ***and*** $m > \varepsilon$ ***and*** $i < i_{max}$ **do**
  $(dist,\ stepSize) := ConeTraceStep(\boldsymbol{p}, \boldsymbol{v}, \alpha, r, t)$
  if $dist < \varepsilon \cdot t \cdot \tan\alpha_0$ **then**
    $m := \frac{dist + r + t \cdot \tan\alpha}{r_0 + t \cdot \tan\alpha_0}$
    $r := (m - \varepsilon) \cdot r_0$
    $\tan\alpha := (m - \varepsilon) \cdot \tan\alpha_0$
  $t := t + stepSize$
  $i := i + 1$
end

Algorithm 2: *The proposed minimal tangent cone algorithm.*

4. Shading Approximation

Let us now focus on approximating the exitant radiance from a $\boldsymbol{p}$ surface point onto the $\boldsymbol{v}$ direction:

$$I(\boldsymbol{p}, \boldsymbol{v}) = \int_\Omega f_r(\boldsymbol{v}, \boldsymbol{n}, \boldsymbol{l}) \cdot V(\boldsymbol{p}, \boldsymbol{l}) \cdot E(\boldsymbol{p}, \boldsymbol{l}) \cdot \cos\theta_{\boldsymbol{l}} \, \mathrm{d}\boldsymbol{l} \qquad (1)$$

where $\boldsymbol{l} \in \Omega$ is an arbitrary direction over the hemisphere Ω, $f_r(\boldsymbol{v}, \boldsymbol{l})$ is the bidirectional reflectance distribution function, $V(\boldsymbol{p}, \boldsymbol{l}) \in \{0, 1\}$ is the visibility function returning 1 or 0 depending on whether the light source is visible from $\boldsymbol{p}$ toward $\boldsymbol{l} \in \Omega$, $E(\boldsymbol{p}, \boldsymbol{l})$ is the incoming radiance at $\boldsymbol{p}$ from the light source along direction $\boldsymbol{l}$, and $\theta_{\boldsymbol{l}}$ is the angle between the surface normal at $\boldsymbol{p}$ and $\boldsymbol{l}$.

The next two sections detail our approximations of Equation 1. Visibility is reduced to only account for the largest occluder returned by the minimal cone search. The occluding surface is assumed to be planar or edge-like at that closest point, so the intersection of the surface and the cross-section of the cone connecting to the light is a straight line, as shown in Figure 3.

4.1. Fast Soft Shadows for Spherical Lights

Let us consider the following crude approximation of Equation 1:

$$I(\boldsymbol{p}, \boldsymbol{v}) \approx \int_\Omega f_r(\boldsymbol{v}, \boldsymbol{n}, \boldsymbol{l}) \, \mathrm{d}\boldsymbol{l} \cdot \int_\Omega V(\boldsymbol{p}, \boldsymbol{l}) \, \mathrm{d}\boldsymbol{l} \cdot \int_\Omega E(\boldsymbol{p}, \boldsymbol{l}) \cdot \cos\theta_l \, \mathrm{d}\boldsymbol{l} \qquad (2)$$

This estimation is visually close to the precise result for spherical lights. To simplify computation, let us approximate the visibility term in two steps. First, we use Algorithm 2 to obtain $m \in [-1, 1]$. Then distance ratio m is used to estimate the ratio of the area that the surface occludes from the cone segment. The occlusion ratio is equivalent to a parametric integral on the unit circle assuming the surface has an edge-like geometry. This gives us

$$\int_\Omega V(\boldsymbol{p}, \boldsymbol{l}) \, \mathrm{d}\boldsymbol{l} \approx \frac{2}{\pi} \int_{-1}^{m} \sqrt{1 - x^2} \, \mathrm{d}x = \frac{m}{\pi} \sqrt{1 - m^2} + \frac{1}{\pi} \arcsin m + \frac{1}{2} \, .$$

For computational efficiency, we use the following approximation:

$$\left(\frac{1}{2} m + \frac{1}{2}\right)^2 \cdot (2 - m) = \mathrm{smoothstep}\left(\frac{1}{2} m + \frac{1}{2}, 0, 1\right) \, .$$

4.2. Numerical Integration for Arbitrarily Shaped Lights

Instead of the crude approximation presented above, we can integrate Equation 1 directly. Sampling the visibility term is the most computationally expensive part, so we approximate V with the largest occluder as before. However, the orientation of the occluding surface is also needed because of the other terms in I, hence the normal vector needs to be computed where the minimum distance is measured. Thus, we can quickly calculate whether a sample is occluded independently of scene geometry.

Moreover, this method is not limited to spherical light sources because it is relatively cheap to calculate if a sample ray hits an arbitrary shaped but simple source of light. This means we can use any ray traceable light source which has a bounding sphere. We used Monte Carlo integration on the $\boldsymbol{l}$ directions inside the cone toward the bounding sphere of a rectangle light.

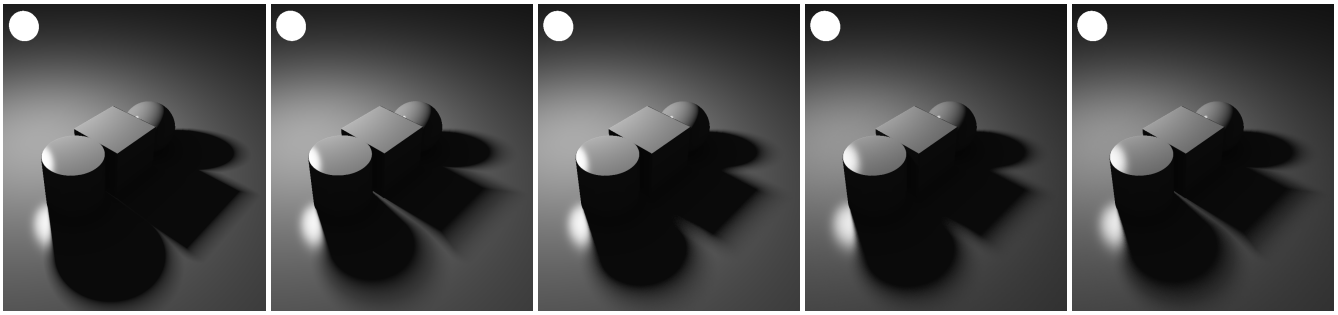
5. Results

The main advantage of the above approximations is that the time overhead of shading depends on the integration method and not the scene or SDF complexity.

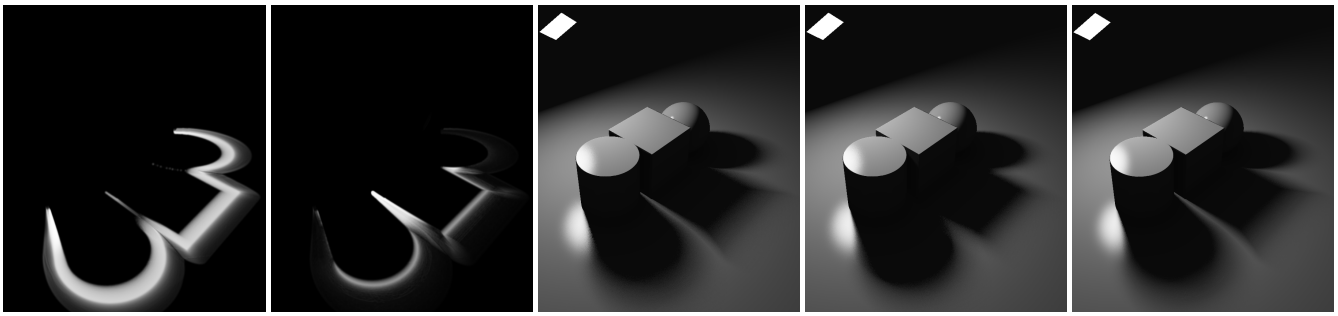
We ran our tests using procedural representations where the scenes are defined as CSG trees stored as shader code. We used NVIDIA's Falcor rendering framework [BYF*18] and implemented the proposed algorithms in HLSL. All performance tests were done on a Windows 10 machine equipped with a GeForce 1070Ti GPU, AMD Ryzen 5 CPU, 16 GB RAM at 1080p.

Spherical lights The algorithms were validated by comparing the Monte Carlo integration of the visibility term to our output as displayed in Figure 4c. The edge-like assumption causes inaccurate shadows at corners and concave surface parts. This approximation difference is visible in Figure 4. To quantify the quality difference between our first and Quílez's improved algorithm, we computed the structural similarity index [WBSS04] with a ground truth image of the same scene obtained with Monte Carlo integration. Our fast method achieved a mean of 0.9968 SSIM similarity score with a standard deviation of 0.0375, whereas Quílez's improved algorithm had an average of 0.9862 with a standard deviation of 0.0891. Both of our proposed algorithms outperform the corresponding Monte Carlo method; that is, on the same configuration (spherical or rectangular light source) and sample count, see Figure 4.

Rectangular lights We used rectangular lights to demonstrate the direct integration of Equation 1, as shown in the second row of Figure 4. On complex scenes, and in both algorithms, we measured third of the render time when we used our approximations.



(a) *[Qui10, Aal18]:* ***1.32 ms*** **(b)** *Our fast method:* ***1.57 ms*** **(c)** *MC of V:* ***36.4 ms*** *@100 spp* **(d)** *Full MC:* ***52.7ms*** *@100spp* **(e)** *Ours:* ***24.4 ms*** *@100 spp*



(f) *SSIM of (a) and (c): 0.985* **(g)** *SSIM of (b) and (c): 0.997* **(h)** *Ours:* ***9.41 ms@35 spp*** **(i)** *Full MC:* ***24.3 ms@35 spp*** **(j)** *Ours:* ***24.3 ms@100 spp***

Figure 4: *Quality and performance of soft shadow algorithms. First row compares methods approximating the shading caused by a spherical light. Figures (a) – (c) approximate the integral in Equation 2. All renders apart from (a) and (b) use* ***M****onte* ***C****arlo integration; samples per pixel (spp) values are displayed. Figure (f) and (g) compare the structural similarity indices of [Qui10, Aal18] and our real time method to an MC ground truth shown on (c). Figures (h) – (j) compares our algorithm to the full* ***M****onte* ***C****arlo method with a rectangular light for various spp counts. Note that our second algorithm traces* 2.86 *times more rays per unit time than full Monte Carlo integration.*

6. Conclusion

This paper presented two approximate solutions to handle area light sources in scenes defined by signed distance functions.

The first algorithm provides a quick approximation for the amount of light occluded while rendering realistic shadows in real-time, as seen in Figure 4b. Empirically, our method proved to be more accurate than that of [Qui10, Aal18] at a comparable speed.

The second algorithm uses the same occlusion approximation, but we evaluate the integral in Equation 1 directly. The measured speed benefit over standard Monte Carlo visible in Figure 4h and 4i is because the integration is now independent of scene geometry. Since we use the same pseudo-random Monte Carlo integration technique, we expect our algorithm to double the performance of other integrating and sampling techniques, such as [DHB17] too.

Acknowledgement The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

References

[Aal18] AALTONEN S.: GPU-based clay simulation and ray-tracing tech in Claybook. Game Developers Conference. 1, 4

[ABB18] ANDERSSON J., BARRÉ-BRISEBOIS C.: Shiny pixels and beyond: real-time raytracing at SEED. Game Developers Conference. 1

[BV18] BÁLINT C., VALASEK G.: Accelerating Sphere Tracing. In *EG 2018 - Short Papers* (2018), Diamanti O., Vaxman A., (Eds.), The Eurographics Association. doi:10.2312/egs.20181037. 1, 2

[BYF*18] BENTY N., YAO K.-H., FOLEY T., OAKES M., LAVELLE C., WYMAN C.: The Falcor rendering framework, 05 2018. URL: https://github.com/NVIDIAGameWorks/Falcor. 3

[DHB17] DUPUY J., HEITZ E., BELCOUR L.: A spherical cap preserving parameterization for spherical distributions. *ACM Trans. Graph. 36*, 4 (2017), 139:1–139:12. doi:10.1145/3072959.3073694. 4

[EL18] EDWARD LIU I. L.: Ray tracing in games with NVIDIA RTX. Game Developers Conference. 1

[Har96] HART J. C.: Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (Dec 1996), 527–545. doi:10.1007/s003710050084. 1, 2

[KSK*14] KEINERT B., SCHÄFER H., KORNDÖRFER J., GANSE U., STAMMINGER M.: Enhanced Sphere Tracing. In *Smart Tools and Apps for Graphics* (2014), The Eurographics Association. doi:10.2312/stag.20141233. 1

[Qui10] QUILEZ I.: Penumbra shadows. URL: https://www.iquilezles.org/www/index.htm. 1, 4

[RMD11] REINER T., MÜCKL G., DACHSBACHER C.: Interactive modeling of implicit surfaces using a direct visualization approach with signed distance functions. *Computers & Graphics* (2011), 596–603. 1

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612. 3