FAST PENUMBRA CALCULATION IN RAY TRACING

Arno Formella
Universität des Saarlandes, FB 14 Informatik, 66041 Saarbrücken, Germany
e-mail: formella@cs.uni-sb.de

Andrzej Łukaszewski
University of Wrocław, Institute of Computer Science,
ul. Przesmyckiego 20, 51-151 Wrocław, Poland
e-mail: anl@ii.uni.wroc.pl

ABSTRACT

Penumbras, or soft shadows, are an important means to enhance the realistic appearance of computer generated images. We present a fast method based on Minkowski operators to reduce the run time for penumbra calculation with stochastic ray tracing. Detailed run time analysis on some examples shows that the new method is significantly faster than the conventional approach. Moreover, it adapts to the environment so that small penumbras are calculated faster than larger ones. The algorithm needs at most twice as much memory as the underlying ray tracing algorithm.

Keywords: shadow calculation, stochastic ray tracing, bounding volumes, Minkowski operators, offsets.

1 INTRODUCTION

Realistic shadow generation plays an important role when producing computer generated images. The human observer is accustomed to seeing shadows in an illuminated scenario in the real world, so they should be present in a computer generated image, too. Moreover, shadows enhance the perception of the third dimension in the two-dimensional image [Wange92].

The computation of shadows is a very expensive task for every rendering algorithm. Adding penumbras or so-called soft shadows makes the problem even more complex. A survey of different techniques is gathered e. g. in [Woo90]. Ray tracing is a powerful rendering algorithm well suited for shadow generation when multiple light sources and reflective surfaces are present.

Stochastic ray tracing [Cook84] is a method where more than one ray is traced per shadow calculation. For each point to be shaded, a certain number of rays is "fired" towards each of the light sources. The target points on the surface of a light source are distributed randomly implementing a kind of Monte Carlo integration method to estimate the correct size of the visible solid angle.

We present a fast method to generate penumbras which is based on stochastic ray tracing. There are no severe restrictions on the shape of the objects or the light sources. However, certain types of objects and light sources will allow for faster rendering times. The main idea is to detect possible regions where penumbra occurs and to confine the expensive process of stochastic ray tracing to those regions.

The next section reviews penumbra calculation from the point of view of ray tracing. Section 3 gives a brief overview of our algorithm. Section 4 introduces Minkowski operators and gives a more formal background of the method for non-spherical shaped objects and light sources. Section 5 presents example images and summarizes some performance results.

2 PENUMBRA CALCULATION

Many simple rendering programs model light sources as mathematical points without any three-dimensional extension. Such light sources cause sharp shadows, because the shadow calculation reflects a step function: a point to be displayed is either in shade or in light.

In real environments, however, the transition from illuminated to non-illuminated regions is smooth. A point is in shade respective to a certain light source when an obstacle totally occludes the light source. In other words, if any ray starting at the point and going towards the light source intersects an opaque surface before it reaches the surface of the light source. Conversely, a point is in light respective to a certain light source when the light source is entirely visible from the point. Penumbra occurs when an obstacle partially occludes the light source, allowing only a subset of the rays to reach the light source.

If we employ stochastic ray tracing to sample the visible solid angle of a light source with d rays per point without any enhancement, the run time to trace the shadow rays is roughly d times larger than the run time in ordinary ray tracing that generates sharp shadows. To achieve good image quality, the value of d should depend on the size of the light source; values of $d \geq 50$ may become necessary.

Two ideas to speedup tracing rays towards linear or area light sources, e. g., [Pouli90, Bao93], have been described in [Woo93]. To decrease the amount of work to be done, the candidate list of objects possibly intersected by the shadow rays is confined to the objects actually intersecting the cone from the point toward the area light source. The candidate list is generated dynamically. The approach can be seen as a special form of cone tracing [Amana84] with shadow caching.

The shadow buffer, as introduced in [Haine86] and extended in [Pearc91], becomes very important in stochastic ray tracing for penumbra. Because we send several rays from the same point to the same light source—clearly, to different positions on the surface of the light—a cached object likely serves as occluding object for many rays. However, the shadow buffer does not exhibit such a large improvement as one might expect at first sight. Many rays for penumbra calculation pass close to the surfaces of the objects but they do not hit the objects. The shadow buffer is best to speedup tracing rays that actually hit objects. Thus, roughly speaking, only half of the rays sent out in a penumbra can profit from the shadow buffer. For points outside of any shadow region, no advantage can be taken of the shadow buffer.

3 OVERVIEW OF THE ALGORITHM

Figure 1 depicts a scenario where a light source L cast a shadow on the surface S because the light is occluded by the object Q. The umbra and the penumbra compose the entire shadow. The basic idea of the algorithm to speedup penumbra calculation is very simple. We detect penumbra regions and employ expensive calculations only where it is necessary. The detection is based on the following observation. If we shrink the light source L to a point and, at the same time, increase the occluding object Q by the radius of L, then the true shadow volume is a subset of the approximate shadow volume. This depends neither on the radius of L nor on the distance between L and Q or L and S. Section 4 includes a more formal and general explanation of this fact.

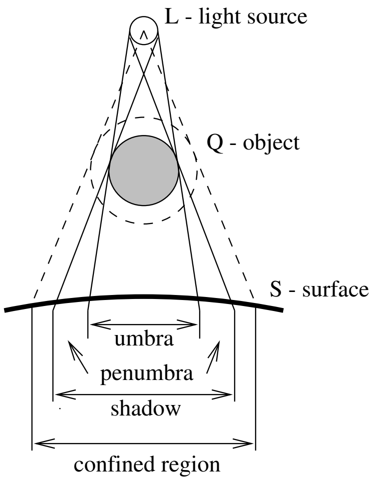

Figure 1: Shadow classification.

Once we have confined the shadow, we can employ stochastic ray tracing to sample the solid angle under which the light source is seen. Outside of the confined region we can skip this step.

If more than one light source is present, we increase all objects by the maximum amount required by all the light sources. We perform ray tracing in two different data sets. The “geometric” data set contains the environment as usual; the “shadow” data set contains the shrunken light sources and the increased objects. We determine in the shadow data set whether a given point belongs to a shadow region or not, i. e., shadow rays are initially traced in the shadow data set. If the point is found to be in shade, we start stochastic ray tracing in the geometric data set. As a further optimization, we detect umbra regions with a similar approach confining further the penumbra region.

4 PENUMBRA DETECTION USING MINKOWSKI OPERATORS

We define both the objects and the light sources as compact and connected sets of points in three-dimensional Euclidean space. We write the distance between two points p and q as $d(p, q)$. We denote by $r(p,q)$ the set of points that defines the ray segment which starts at p and ends at q. Let us define Minkowski operators (e. g., as in [Latom91]) which provide a convenient way to express set operations.

Definition 1 (Minkowski Operators) *For two subsets A and B of Euclidean vector space,* Minkowski sum *and* difference *are defined as:*

$$A \oplus B = \{a + b \mid a \in A, b \in B\} \ , \quad (1)$$
$$A \ominus B = \{a - b \mid a \in A, b \in B\} \ . \quad (2)$$

We will use the Minkowski operators for expanding original objects casting shadows. In the special case of spherical light sources this will simplify to the operation of solid offsetting as defined in [Farou85, Rossi86]. Let $B(p,d)$ denote the ball located at p with radius d. If the light source is such a ball then the solid offset is the required expansion of the obstacle Q

$$\mathcal{O}_d(Q) = Q \ominus B(0,d) \ . \quad (3)$$

The more frequently used equivalent definition of solid offset is as follows.

Definition 2 (Solid Offset) *For an object Q and a distance d, a* solid d-offset $\mathcal{O}_d(Q)$ *is defined as the set of points that are not farther than d from Q, i. e.,*

$$\mathcal{O}_d(Q) = \{p \mid \exists\, q \in Q : d(p,q) \leq d\} \ . \quad (4)$$

We will also need the following property which is weaker than convexity. We say a subset A of the Euclidean space is *star-convex* respective to a point $c \in A$ if for any point $p \in A$ the segment $r(c,p)$ is totally included in A, i. e., we can reach with a straight line from the so-called center c any other point in the set without leaving it.

4.1 Basic Lemma

For a star-convex light source L with center c. We can derive the following simple lemma which is illustrated in Figures 2 and 3.

Lemma 1 *Let L be a star-convex set respective to a point $c \in L$. If the ray $r(c,p)$ does not intersect $Q \ominus (L \ominus \{c\})$ then the point p is not in shade of Q respective to the light source L.*

Proof (by contradiction): Let $C = \bigcup_{x \in L} r(x,p)$ denote the visibility cone of the light source L as seen from point p. The point p is in shade of Q if and only if $C \cap Q \neq \emptyset$ but since the set L is star-convex respective to the point c we have $C \subset r(c,p) \oplus (L \ominus \{c\})$. Thus, we have $(r(c,p) \oplus (L \ominus \{c\})) \cap Q \neq \emptyset$ which means that there are points $r \in r(c,p)$, $l \in L$ and $q \in Q$ such that $r + (l - c) = q$. But this is equivalent to $r = q - (l - c)$ which means that $r(c,p) \cap (Q \ominus (L \ominus \{c\})) \neq \emptyset$. Hence, the ray does intersects the expanded object.

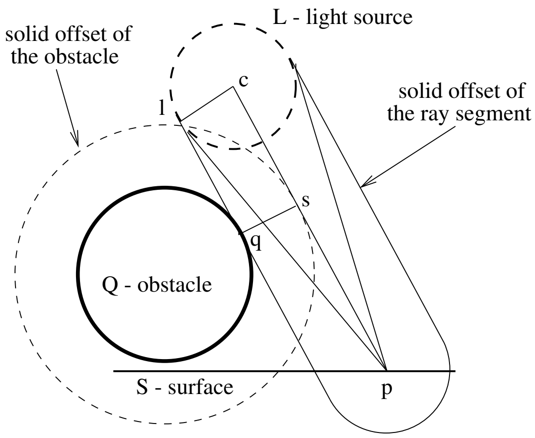

Figure 2: Not-in-shade condition for spheres.

With the notion of solid offsets, we obtain: a point p is not in shade of an object Q respective to a light source $B(c,d)$ if the ray $r(c,p)$ does not intersect the solid offset $\mathcal{O}_d(Q)$.

Minkowski operators and offsets are hard or costly to evaluate in the general case. However, the not-in-shade condition is the base for many ideas to calculate bounding boxes or approximations of expanded objects.

Solid offsets are useful for several reasons. First, they are easy to evaluate for spheres (offsets of spheres are spheres with bigger radius). Second, for other simple geometric objects like cylinders and cones they can be easily bounded within a grown object by changing few parameters. Third, there are specialized algorithms for calculating offsets of parametric surfaces which can be used.

The usage of Minkowski operators is more effective for arbitrary shaped light sources than the usage of simple offsets. If we enclose a linear or planar light source in a bounding sphere, we can handle the light with solid offsetting as described above. However, the approximate shadow volume is unnecessary large. Figure 3 depicts the scenario for a linear light source. If we used just solid offsets, we would have to put the light source into a ball and expand all the bounding boxes of the objects equally in all directions.

4.2 Optimizations

Lemma 1 is a good criterion to detect possible shadow regions. However, there are points in full light which we do not detect since the visibility cone is smaller than the set we are checking (expressed in terms of

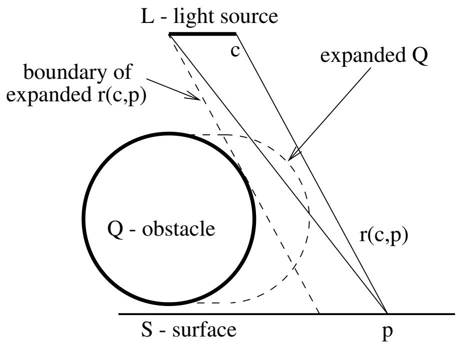

Figure 3: Linear light source causes smaller extended object.

Minkowski operators). We can further confine the possible shadow regions if we construct the shadow data set with smaller offsets (or with scaled sets and Minkowski operators). The idea is depicted in Fig. 4. We start with spherical light sources and offsets and will present obvious generalizations.

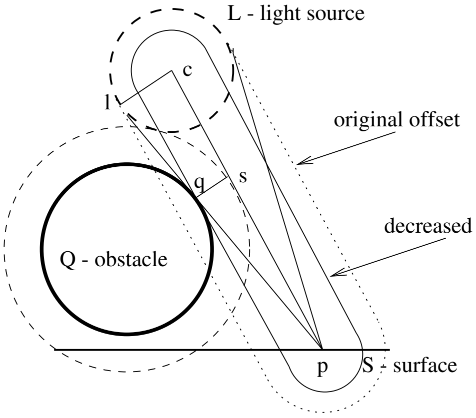

Figure 4: Radius optimization.

As we see in Fig. 4, instead of taking the solid offset of object Q at distance $d(c,l)$ for a shadow intersection test, we can use the offset at distance $d(s,q)$ for these points. We can calculate the ratio t in which we can shrink the offset distance with simple geometry:

$$t = \frac{d(s,q)}{d(c,l)} = \frac{d(p,q)}{d(p,l)} = \frac{d(p,q)}{d(p,q)+d(q,l)} \tag{5}$$

If we choose the maximum value of t for all points $p \in S, l \in L$, and $q \in Q$, with $r(c,p) \cap (Q \ominus (L \ominus \{c\})) \neq \emptyset$, we can shrink the offset distance still detecting the region containing both umbra and penumbra:

$$\begin{aligned} t_{max} &= \max\left\{\frac{d(p,q)}{d(p,q)+d(q,l)}\right\} = \\ &= \frac{\max\{d(p,q)\}}{\max\{d(p,q)\}+\min\{d(q,l)\}} \end{aligned} \tag{6}$$

If we denote the scaling of the set A by a real number t as $t \cdot A = \{t \cdot a \mid a \in A\}$, then we can formulate the following result.

Lemma 2 *Let L be a star-convex set respective to a point $c \in L$ and let t_{max} be defined as above (for S, L and Q). If the ray $r(c,p)$ does not intersect the set*

$$Q \ominus t_{max} \cdot (L \ominus \{c\}) \tag{7}$$

then a point p is not in shade of Q respective to L.

The improvement is significant if the distance to the light source is not too small compared to the diameter of the scene. Since it depends on the placement of the object and the light source, it is much more effective to calculate the ratio t_{max} for each object and light source than to do it globally.

In the case of several light sources we take for a given object the maximum shrinking ratio t_{max} for all light sources and calculate only one expanded object. t_{max} is a lower bound for the shrinking ratio and may be difficult to be calculated. However, any value t with $1 > t \geq t_{max}$ can be used.

4.3 Umbra detection

The previous discussion allowed us to distinguish between the region in full light and the region in shadow. However, there might exist an umbra where we need not to use stochastic ray tracing, either. In the following, we will denote set completion as $\overline{A} = \{x | x \notin A\}$. The following lemma can be used to detect umbra.

Lemma 3 *Let L be a star-convex set respective to a point $c \in L$. If the ray $r(c,p)$ intersects $\overline{\overline{Q} \ominus (L \ominus \{c\})}$ then a point p is in the umbra of Q respective to the light source L.*

Proof: We can state the in-umbra condition as well in the following form (see Fig. 5 for an illustration). There exists a point $r \in r(c,p)$ such that $r \notin \overline{Q} \ominus (L \ominus \{c\})$ which is equivalent to $\forall \bar{q} \notin Q \ \forall l \in L \ : \ r \neq \bar{q} - (l - c)$. With simple transformations, we obtain $\forall \bar{q} \notin Q \ \forall l \in L \ : \ \bar{q} \neq r + (l - c)$. This means that the set $\{r\} \oplus (L \ominus \{c\})$ is totally included in the set Q. But this is nothing else but the set L translated by $r - c$ which is star-convex respective to point r. Hence, for an arbitrary point $l \in L$ the ray $r(p,l)$ intersects the set $\{r\} \oplus (L \ominus \{c\})$ and so the ray intersects object Q what means that the point p is in the umbra.

We can re-formulate the in-umbra condition especially for spherical light sources using negative solid offsets (see [Rossi86] for their definition). The same optimization considerations for the distance as stated above apply, so we may use the bigger set as well:

$$\overline{\overline{Q} \ominus t_{max} \cdot (L \ominus \{c\})} \ . \tag{8}$$

		Cylinders	Balls4	Rings	Molecule	Bust
a)	simple ray tracing	0.91	5.42	1.83	1.34	3.15
b)	stochastic ray tracing	12.05	64.21	32.35	8.25	37.47
	fast penumbra					
c)	standard	6.10	43.48	15.11	5.66	32.46
d)	optimized detection	5.72	26.81	10.54	5.02	24.02
e)	with umbra detection	5.22	25.39	11.04	4.95	

Table 1: Run times in seconds for different images and algorithms.

Remarks: *a)* Traditional ray tracing which produces sharp shadows. *b)* Penumbra with classical stochastic ray tracing. *c)* Our method with shadow data sets. *d)* Optimized method with reduced extended objects according to the t_{max} ratio. *e)* Additional umbra detection. (We still did not implement inner offsets for meshes.)

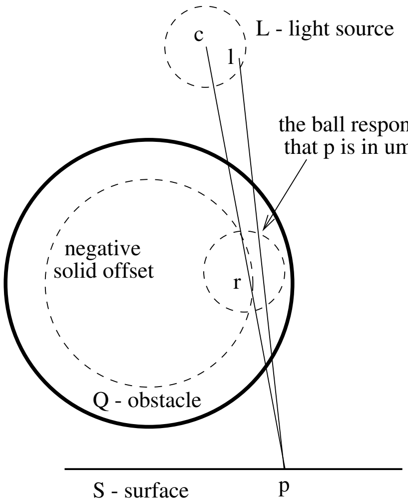

Figure 5: Negative offset.

The umbra detection is done after having detected an object possibly casting shadow which is a good candidate to pass the test. Making the full test in a whole third data set holding only the shrunken objects requires more memory and more time. The gain is not sufficient since most of the tests fail. The experiments have shown that in almost all cases testing all the objects for umbra is slower than testing only one object which passed the penumbra detection test already.

5 PERFORMANCE RESULTS

We have incorporated our algorithm into a ray tracer implemented by one of the authors [Forme95]. Basically, it does not matter which classical method to enhance the intersection finding process is used. It can be incorporated into any ray tracing kernel.

We compare the run times of the new method to the run times of traditional ray tracing without penumbra and classical stochastic ray tracing. More details are added as remarks in Table 1. We present several examples of geometric data sets: a simple scene with *cylinders* and spheres (Fig. 9), a complex *molecule* (Fig. 10) transformed from the Brookhaven Protein Data Bank, a *bust* (Fig. 11) modeled as a mesh of triangles, and the fractal *balls* (Fig. 12) and the *rings* (Fig. 13) from the SPD-benchmark [Haine87]. All measurements are done on a Sun SparcEnterprise 4000, 168 MHz, 1.125 GByte RAM and reflect real time of the ray tracing loop without preprocessing. The test have been done being a single user so that the real time has been essentially equal to the user time.

Table 2 characterizes the different images. The resolution for the tests has been set to 128×128 pixel. For larger images, the run times scale almost linearly with the resolution. The number d of distributed rays was always set to 32. We enhanced the tracing of shadow rays with a shadow buffer. For each node of the ray tree a queue of up to two objects is buffered. A miss in the buffer enforces a delete of the last element in the queue, a hit of an object initiates an insertion as the first element of the queue.

The following trade-off can be observed (Table 2): Our fast method significantly reduces the number of shadow rays to be traced, especially when decreased offsets and umbra region detection are employed. On the other side, the number of intersection tests per ray is increased. The increment is larger using decreased offsets. The umbra detection may not always pay-off, e. g., in the rings example, because the inner offset objects become too small. Additionally, we observed that the light cache hit rate has slightly improved for the more complex data sets.

Table 3 shows the memory requirements of the different implementations. As long as the scene description is small (simple scene) the additional memory required for the fast penumbra calculation is negligible. For the larger scenes at most twice as much memory is required.

The speedup which is obtained with the new method depends on the geometry, especially on the size of the light sources and on the size of the visible penumbras. Table 4 summarizes the run times for the balls3 data

		Cylinders	Balls4	Rings	Molecule	Bust
	# objects	11	7383	62	1685	98506
	# spherical light sources	2	3	3	2	1
	# reflected rays	20167	10976	2962	7303	0
a)	# shadow rays	31592	51719	47415	4260	32136
	# I_{geom}	0.85	1.74	1.40	0.99	0.94
b)	# shadow rays	1027757	1397529	1459345	105153	496704
	# I_{geom}	1.30	2.36	1.22	1.80	2.13
c)	# shadow rays	324911	430300	367355	34615	196667
	# I_{geom}	1.87	2.43	3.31	3.94	2.87
	# I_{shadow}	1.13	2.32	1.72	2.27	4.62
d)	# shadow rays	289054	215151	215155	26800	135324
	# I_{geom}	2.01	4.03	4.39	4.53	4.00
	# I_{shadow}	1.08	0.85	1.39	1.79	2.05
e)	# shadow rays	287481	213120	215155	26691	
	# I_{geom}	1.77	3.16	4.39	4.29	
	# I_{shadow}	1.08	0.85	1.39	1.79	

Table 2: Characteristics of the example scenes.

Remarks: In all scenes the number of primary rays was equal to 16384. The numbers I_{geom} and I_{shadow} denote the number of intersection tests per ray in the geometry data set and in the shadow data set, respectively. For the description of the different methods see remarks in Table 1.

		Cylinders	Balls4	Rings	Molecule	Bust
b)	simple ray tracing	44	3064	82	670	58181
c)	stochastic ray tracing	45	3070	83	693	58181
	fast penumbra					
d)	standard	48	5535	114	1265	110176
d)	optimized detection	48	5991	117	1296	110197
e)	with umbra detection	49	5990	117	1296	

Table 3: Memory requirements in KByte for different images and algorithms.

Remarks: The numbers reflect dynamically allocated memory. In additional to the given values, a frame buffer of 50 KByte was allocated. For the description of the different methods see remarks in Table 1.

set with 822 objects. The size of the light sources was increased for the benchmark according to the sizes of the spheres being present in the data set. Stochastic ray tracing was performed with $d = 32$. The speedup for the fast method ranges from 1.76 to 7.83 depending on the size of the light sources: the smaller the light sources, the better the improvement in run time.

	fast	class.	speedup
Balls3		3.29	
a)	5.63	44.13	7.83
b)	9.88	44.33	4.48
c)	17.10	45.30	2.64
d)	27.21	48.10	1.76

Table 4: Run times for the Balls3 data set for different sizes of the light sources.

Remarks: The sizes of the light sources were set to the sizes of the spheres (*a)* smallest sphere, *d)* largest sphere). The first line shows the run time for simple ray tracing with no penumbra.

In Fig. 6–8 we compare the run time of classical stochastic ray tracing with the run time of the version of the improved penumbra calculation (decreased offsets included). The number d of stochastic rays per point is increased (x-axis). The function plots can be approximated with linear equations. Comparing the slopes of the lines we obtain for large d that the speedup for the simple scene is 2.1, for the molecule scene 2.7 and for the bust scene 1.8, respectively.

6 CONCLUSION

We presented a new algorithm to speedup the calculation of penumbra. The main idea is to detect the shadow regions such that stochastic ray tracing is confined to the penumbra. We used the notion of Minkowski operators and solid offsets to provide the means to handle a variety of differently shaped light sources and objects. We proved formally that the method works correctly and that it essentially renders the same images as stochastic ray tracing.

We described and implemented an improvement of the

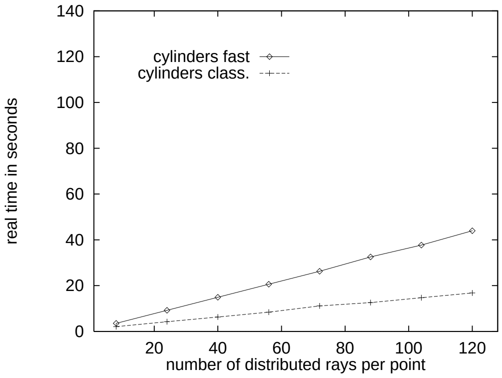

Figure 6: Run time comparison for an increasing number of distributed rays per point in the simple scene with cylinders.

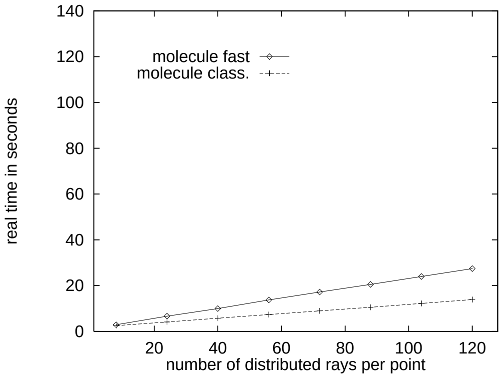

Figure 7: Run time comparison for an increasing number of distributed rays per point in the molecule scene.

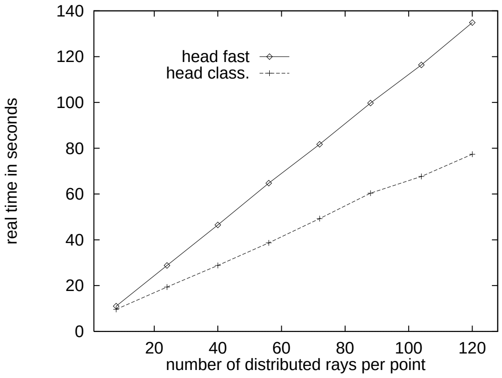

Figure 8: Run time comparison for an increasing number of distributed rays per point in the bust scene.

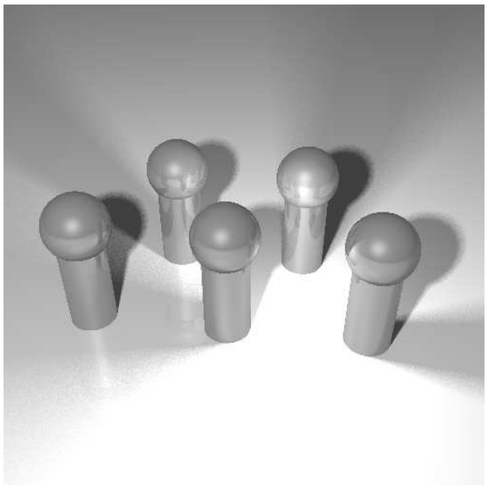

Figure 9: Penumbra in a simple scene.

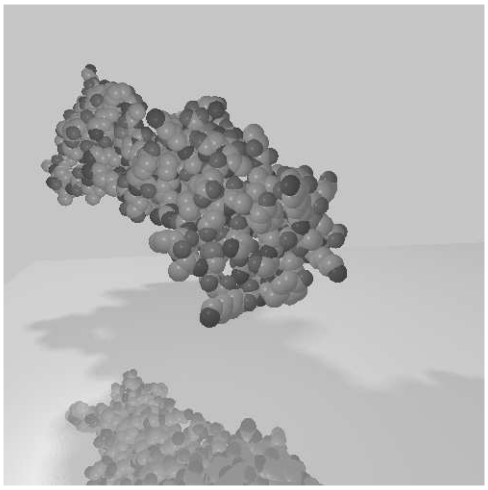

Figure 10: Penumbra for a compact molecule.

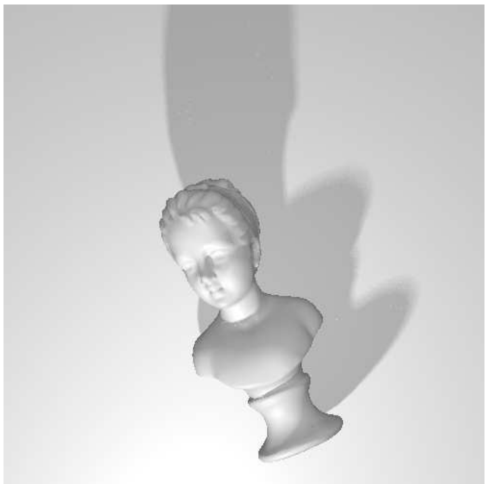

Figure 11: Penumbra in the bust scene.

basic algorithm which used decreased offsets. The effectiveness of the algorithm depends on the particular geometric data set. On average, the presented sample scenes could be rendered two times faster compared to the run time of classical stochastic ray tracing. However, if the penumbra regions are small respective to the visible regions in the scene, much higher speedups can be obtained. The additional memory requirements never exceeded in our experiments a factor of two.

The method seems to be very appropriate to be incorporated into the radiance system [Ward94] to improve further the run time of the penumbra calculation. Further research might investigate the calculation of combined convex *bounded* volumes that would allow to detect the umbra more precisely (for instance for CSG-models). A single bounded volume might be calculated for a number of joined objects (meshes).

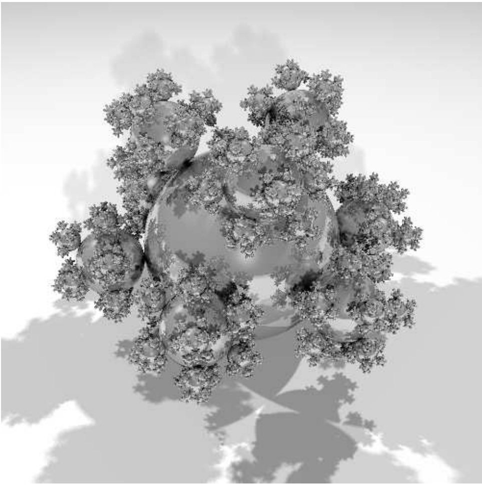

Figure 12: Penumbra in the balls scene.

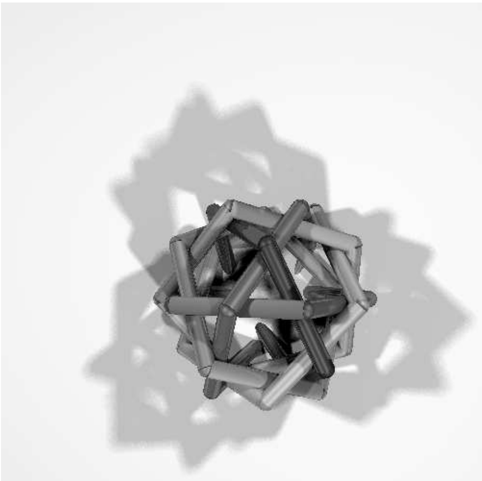

Figure 13: Penumbra in the rings scene.

Acknowledgment

We want to thank Stefan Karbacher, University of Erlangen, (Courtesy by RSI GmbH, Germany) for providing the mesh data of the bust.

References

[Amana84] Amanatides, J.: Ray Tracing with Cones, *Computer Graphics (SIGGRAPH '84 Proceedings)*, ACM Press, Vol.18, No.3, pp.129–135, 1984

[Bao93] Bao, H., Peng, Q.: Shading Models for Linear and Area Light Sources, *Computers and Graphics*, North-Holland, Vol.17, No.2, pp.137–145, 1993

[Cook84] Cook, R.L., Porter, Th., Carpenter, L.: Distributed Ray Tracing, *Computer Graphics (SIGGRAPH '84 Proceedings)*, ACM Press, Vol.18, No.3, pp.137–145, 1984

[Farou85] Farouki, R.: Exact Offset Procedures for Simple Solids, *Computer Aided Geometric Design*, North-Holland, Vol.2, No.4, pp.257–279, 1985

[Forme95] Formella, A., Gill, Ch.: Ray Tracing: A Quantitative Analysis and a New Practical Algorithm, *The Visual Computer*, Springer Verlag, Vol.11, No.9, pp.465–476, 1995

[Haine86] Haines, E.A., Greenberg, D.P.: The light buffer: A Ray Tracer Shadow Testing Accelerator, *IEEE Computer Graphics and Applications*, IEEE Computer Society, Vol.6, No.9, pp.6–16, 1986

[Haine87] Haines, E.A.: A Proposal for Standard Graphics Environments, *IEEE Computer Graphics and Applications*, IEEE Computer Society, Vol.7, No.11, pp.3–5, 1987

[Latom91] Latombe, J.C.: *Robot Motion Planning*, Kluwer Academic Publishers, 1991

[Pearc91] Pearce, A., Jevans D.: Exploiting shadow coherence in ray tracing, *Proceedings of Graphics Interface '91*, Morgan Kaufmann, pp.109–116, 1991

[Pouli90] Poulin, P., Amanatides, J.: Shading and Shadowing with Linear Light Sources, *Eurographics '90*, North-Holland, pp.377–386, 1990

[Rossi86] Rossignac, J.R., Requicha, A.A.G.: Offsetting Operations in Solid Modelling, *Computer Aided Geometric Design*, North-Holland, Vol.3, pp.129–148, 1986

[Wange92] Wanger, L.: The Effect of Shadow Quality on the Perception of Spatial Relationships in Computer Generated Imagery, *Computer Graphics ((1992 Symposium on Interactive 3D Graphics)*, ACM Press, Vol.25, No.2, pp.39–42, 1992

[Ward94] Ward, G.J.: The RADIANCE Lighting Simulation and Rendering System, *Proceedings of SIGGRAPH '94*, ACM Press, pp.459–472, 1994

[Woo90] Woo, A., Poulin, P., Fournier, A.: A Survey of Shadow Algorithms, *IEEE Computer Graphics and Applications*, IEEE Computer Society, Vol.10, No.6, pp.13–32, 1990

[Woo93] Woo, A.: Efficient Shadow Computations in Ray Tracing, *IEEE Computer Graphics and Applications*, IEEE Computer Society, Vol.13, No.5, pp.78–83, 1993