The Visual Computer

Ray-bound tracing for perfect and efficient anti-aliasing

Masataka Ohta[1]
and Mamoru Maekawa[2]

[1] Computer Center, Tokyo Institute of Technology, 2-12-1 O-okayama, Meguro-ku, Tokyo 152, Japan
[2] Department of Information Science, Faculty of Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

A complete detection of whether aliasing occurs in a given pixel is possible by using the concept of bounded rays and ray-bound tracing. A coherent set of rays can be bounded by bounding both their origins (by a sphere) and directions (by a circle on a unit sphere). By tracing bounds of rays in a pixel, along the direction of reflection, refraction or to light sources, it is possible to obtain an upper bound on the degree of aliasing in the pixel. Ray bound tracing is possible against various shapes and with various shading algorithms.

Key words: Ray-tracing – Anti-aliasing – Coherence – Bound

1 Introduction

Ray tracing (Whitted 1980) is a rendering algorithm which can produce the most realistic computer generated images. But anti-aliasing with ray tracing has been a difficult problem. In ray tracing, anti-aliasing is performed by super sampling or by tracing a set of rays.

The super sampling approach is first found in (Whitted 1980), and statistically satisfactory approaches are found in Cook et al. (1984) and Cook (1986). If all pixels are super-sampled many many times, there will eventually be no aliasing. But such an approach is too costly; pixels which do not cause aliasing should be sampled only once. The problem is that determination of the degree of aliasing in a pixel has been difficult. The approach in Whitted (1980) is imperfect and leaves some aliasing, while the approach in Cook et al. (1984) is inefficient even with the improvements such as are described by Lee et al. (1985) and Cook (1986).

The other approach to anti-aliasing is tracing a set of rays instead of individual rays. But, it is, in general, impossible, because no easy representation of the set exists after reflection or refraction on curved surfaces. Accurate tracing is therefore severely limited, allowing only polygonal objects and requiring a modified law of refraction (Heckbert and Hanrahan 1984). Approximations with enough accuracy is also difficult and can be used only in special cases (Amanatides 1984).

This paper describes a new approach for anti-aliasing, which is a combination of super sampling and tracing of sets of rays. The approach uses super sampling for anti-aliasing, but a set of rays are traced to see whether a pixel needs to be anti-aliased. To determine the necessity of anti-aliasing, the concepts of ray bounds and ray-bound tracing are introduced. Although a ray bound only approximates an actual set of rays, it is traced widely enough to always include it. The approach is perfect, because it can find every pixel requiring anti-aliasing, and is efficient since most pixels not requiring anti-aliasing are detected as such.

2 Coherence, ray bounds and ray-bound tracing

One major problem of ray tracing is its slow speed. But recent research results (Fujimoto et al. 1986; Joy and Bhetanabhotla 1986) show that acceleration is possible by extracting some sort of coher-

The Visual Computer (1990) 6:125–133

ence. Coherence is also useful for anti-aliasing because if rays in a pixel are coherently traced, i.e. have nearly the same origins, nearly the same directions, and intersect with the same objects, no aliasing will occur in the pixel. But extraction of such coherence has been difficult.

The authors have developed a new type of coherence, *ray coherence* (Ohta and Maekawa 1987), to show that a ray can be traced in constant time regardless of the number of objects. In that paper, for formal treatment of ray coherence, the concept of a ray bound was introduced. A similar coherence is also described in Arvo and Kirk (1987). Ray bounds are a useful means of treating a set of similar rays simultaneously and in a simple and numerically stable way.

Figure 1a is an example of similar rays. They have similar origins and similar directions. Therefore, as shown in Fig. 1b, their origins can be bounded by a small sphere. At the same time, as shown in Fig. 1c, their directions can be bounded by a small circle on a sphere (S^2) in direction space using spherical geometry. Though other forms of bounding shapes are possible, e.g., by convex hulls (Ohta and Maekawa 1987), they are too complex and too slow to be practical. Therefore, in this paper, a ray bound is always represented by a circle (in S^2) and a sphere (in R^3).

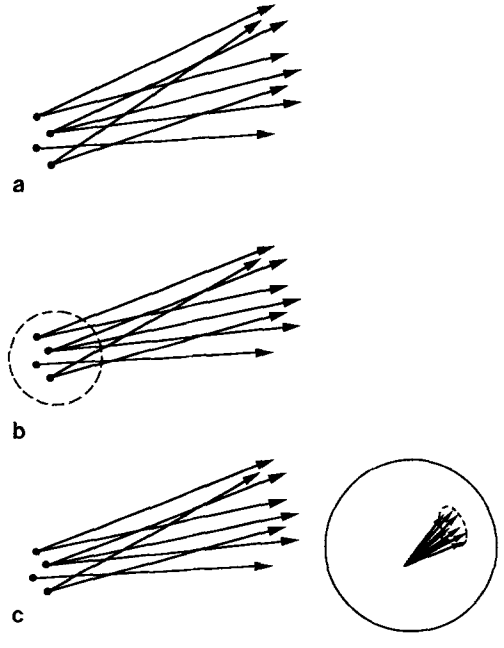

Fig. 1a–c. Concept of ray bound. **a** A coherent set of rays; **b** bounding of origins in R^3; **c** bounding of directions in S^2

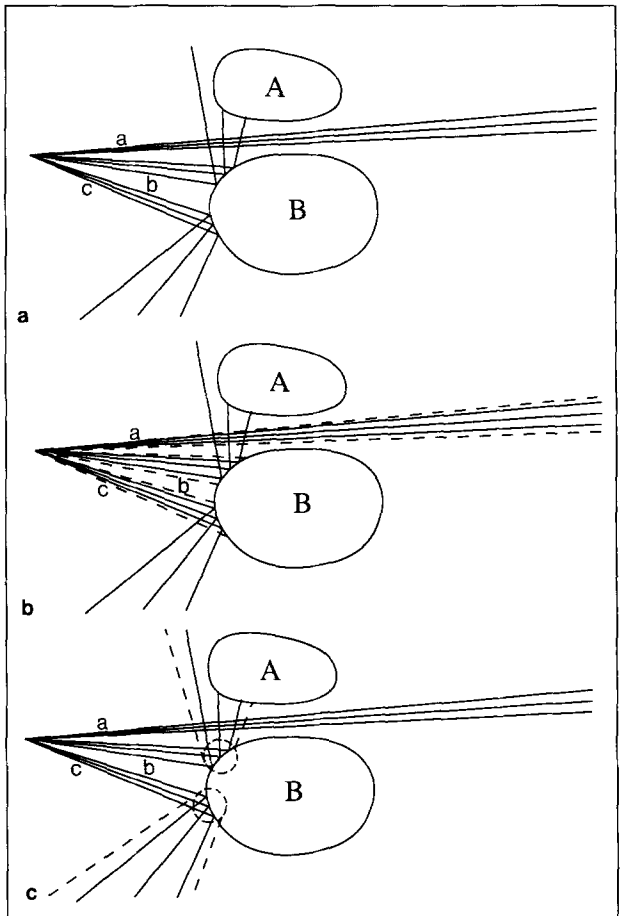

Fig. 2a–c. Concept of ray-bound tracing. **a** Situation; **b** first generation ray bounds; **c** second-generation ray bounds

The important fact is that it is possible to trace a ray bound in a fashion similar to tracing an individual ray. Figure 2a shows tracing of three sets of rays corresponding to pixels *a*, *b* and *c* with two reflective objects A and B. As in Fig. 2b, the first-generation ray bounds are all traced coherently. Because the ray bound of *a* intersects with no objects, there is no aliasing in pixel *a*. For the first-generation ray bounds of pixels *b* and *c*, which coherently intersect with object A, the next-generation ray bounds should be traced and checked for coherence. As in Fig. 2c, the second-generation ray bounds are generated. In this case, because the ray bounds of *b* partially intersect with object B, pixel *b*, will contain aliasing. Pixel *c* is traced coherently.

In more general cases, there are various causes of aliasing, which will be discussed in the next section.

3 External ray coherence, surface ray coherence and anti-aliasing

3.1 Types of coherence

There are several types of relationships between objects and a ray bound. Among them, two coherent cases are useful for anti-aliasing. They are:

(1) All rays in the bound will intersect with no object (Fig. 3a).

(2) All rays in the bound will first intersect with an object in a *similar* manner (Fig. 3b).

Note that the similarity here is measured by the shading algorithm of the object.

We call the former *external coherence*, and the latter *surface coherence*. With the concept of these two types of coherence the detection of aliasing is simple.

3.2 Detection of aliasing

If external ray coherence holds for a ray bound, only aliasing caused by the background is possible and this can be detected easily. If surface coherence holds with a ray bound, coherence is recursively checked for any additional ray bounds generated for shadowing, reflection or refraction. If neither type of coherence is observed the ray bound may contain aliasing, and hence, anti-aliasing should be performed. It should be noted that underestimation of coherence, i.e., overestimation of aliasing, is always safe though it costs the extra effort of anti-aliasing. Consequently, numerically critical cases can be safely handled.

3.3 Detection of external ray coherence

External coherence can be quickly detected even with complex shapes by using bounding volumes (often spheres). If rays in a ray bound intersect with no bounding volume, they intersect with no object, so that external coherence holds. If some ray in a ray bound intersects with some bounding volume of an object, external coherence is not observed

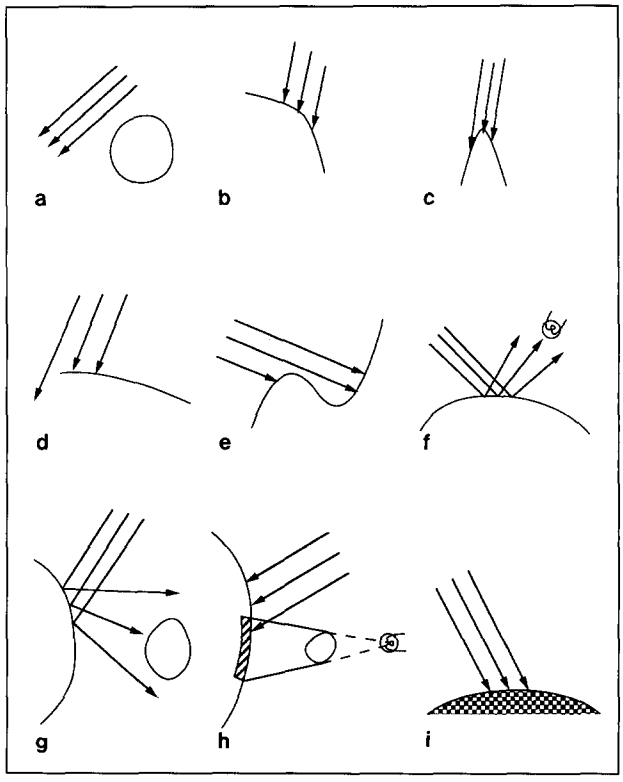

Fig. 3a–i. Relationships between an object and a ray bound. **a** Externally coherent; **b** surface coherent; **c** abrupt change of normal vector; **d** edge of the object; **e** concave object; **f** keen specular; **g** lack of coherence after reflection; **h** edge of shadow; **i** fine mapped texture

by this method. Obviously, the tightness of the bounding volume is important to prevent unnecessary anti-aliasing. But even for complex objects like bi-cubic patches, a tight but still simple bound (such as a union of spheres), can be constructed using recursive subdivision.

As shown in Fig. 3c–i, there are many causes for lack of coherence, so the detection of surface ray coherence is more difficult than that of external coherence.

Surface coherence must be computed according to the shape and shading algorithm of objects. For example, incoherence in Fig. 3c–e is caused by the shape of the object, and incoherence in Fig. 3f–i is caused by shading. To separate the effects of shape and shading algorithm, the concept of an intersection bound is introduced. An intersection bound is a bound of things related to the intersection between a ray bound and an object. It contains:

1) a bound of points of the intersection;

2) almost always, a bound of directions of normal vectors at the intersection points;
3) optionally, a bound of local coordinates on the surface of the object.

An intersection bound can be calculated using only shape information of the object. After that, using the intersection bound only, aliasing effects related to shading can be calculated. An algorithm to compute an intersection bound of a ray bound and a sphere is described in Appendix A. More general cases are treated in Appendices B (quadratic surfaces) and C (relatively flat surface).

Based on the various causes of aliasing shown in Fig. 3, the following detection algorithms are applicable:

1) For the detection of abrupt change of surface normal vector, a bound of normal vectors of the intersection bound can be directly used. Though actual threshold depends on the accuracy requirement, maximum normal vector angle (to the center of the bound) of 15° (corresponds to 26% worst case variation of diffuse component) seems to be a good choice.

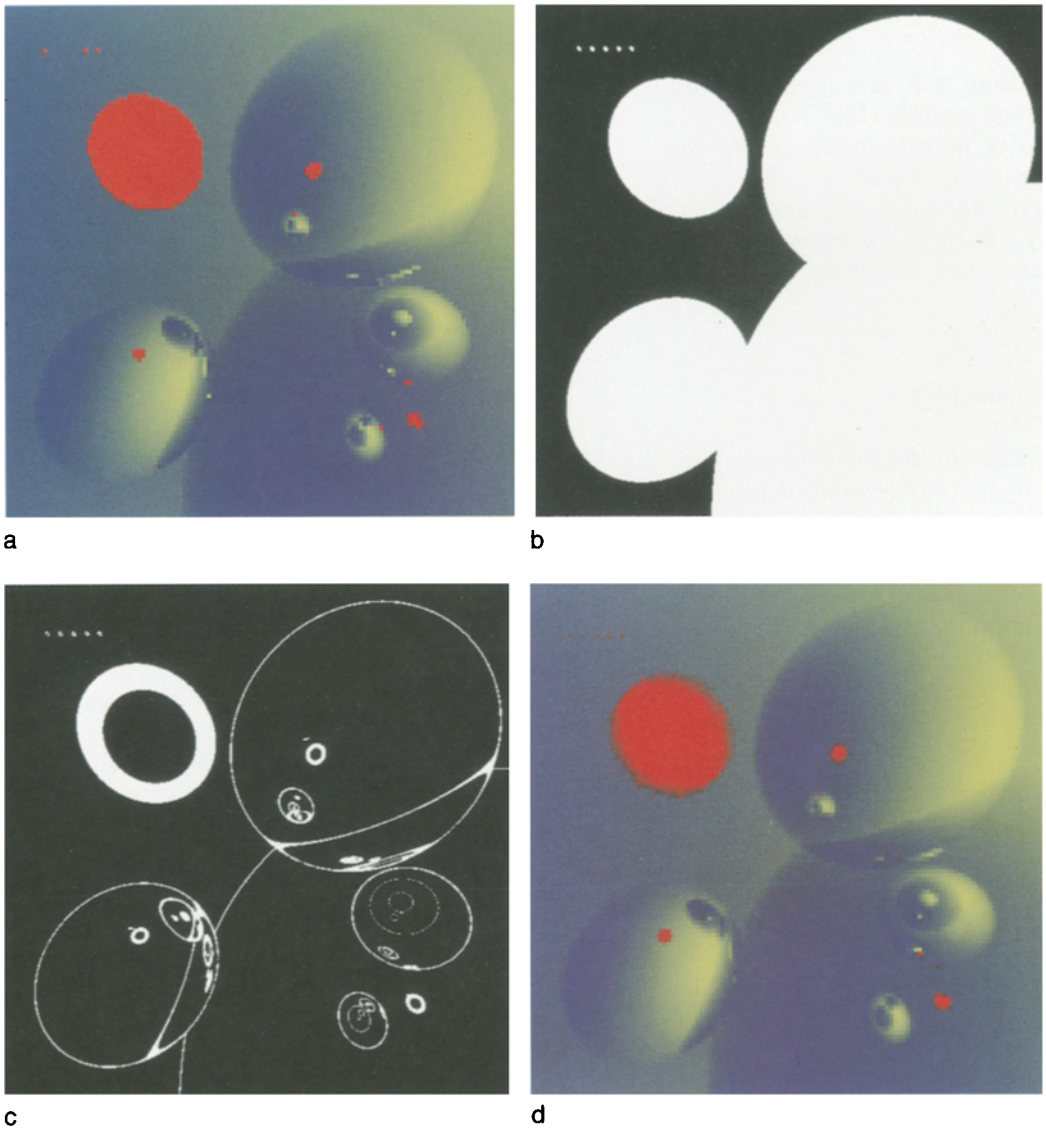

Fig. 4a–d. Examples of bound tracing with reflection and motion blur. **a** Image without anti-aliasing (128 * 128); **b** pixels in need of anti-aliasing using external coherence only (512 * 512); **c** pixels in need of anti-aliasing using ray-bound tracing (512 * 512); **d** anti-aliased image (128 * 128)

2) For the detection of object edges, a bound of local coordinates of the intersection bound can be directly used.
3) For the detection of the interior silhouette edges of the object, a bound of normal vectors of the intersection bound can be used. That is, if there are two parallel vectors: one in the direction bound of a ray bound and another in the direction bound of a normal vector of an intersection bound, there is a possibility of interior silhouette edges.
4) For the detection of the highlight aliasing, bounds of intersection points and normal vectors can be used along with the locations of the light sources. That is, if a ray to the light source is included (or nearly included, depending on shininess of the surface) in a secondary ray bound of reflection rays, there is a possibility of highlight aliasing.
5) For the detection of lack of coherence after reflection or refraction, the next generation ray bound can be computed and traced unless the maximum number of generations of ray bound have not been reached. The new bound of the ray origins is the bound of the intersection points of the intersection bound. The new bound of the direction can be computed using the bound of the direction of the old ray bound and the bound of the direction of the normal vectors of the intersection bound. For the detailed description, see Appendix D.
6) For the detection of shadow edges, another ray bound is generated using the intersection bound and the location of light sources.
7) For the detection of lack of coherence caused by compresses texture, a bound of local texture coordinates of the intersection bound is directly applicable.

Various blurring effects with distributed ray tracing can also be processed as surface coherence.

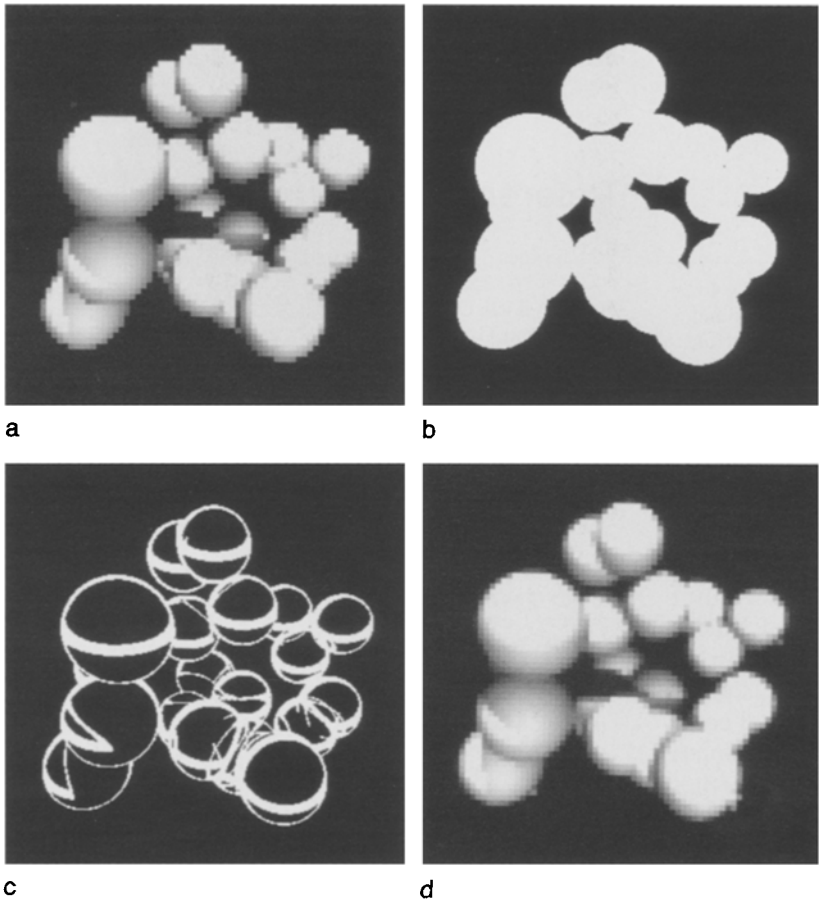

Fig. 5a–d. Examples of bound tracing with shadows. **a** Image without anti-aliasing (128 * 128); **b** pixels in need of anti-aliasing using external coherence only (512 * 512); **c** pixels in need of anti-aliasing using ray-bound tracing (512 * 512); **d** anti-aliased image (128 * 128)

4 Performance measurement

Figure 4a is a non-anti-aliased image of three perfectly reflective spheres, five very small red spheres and one moving red sphere with 128 by 128 resolution. Without anti-aliasing, only three of five very small spheres are visible and the moving sphere is not blurred. Figure 4b and c shows the result of ray-bound tracing. Figure 4b uses only external coherence. White pixels show areas that should be anti-aliased. Figure 4d is the final anti-aliased image using bound tracing and distributed ray tracing with 16 samples per pixel. As can be seen, even very small spheres are detected and anti-aliased properly.
Figure 5a–d shows similar examples with 20 diffuse spheres with shadows cast from two light sources.
The computation times in Tables 1 and 2 are measured on SONY NEWS with a floating point coprocessor (comparable speed to SUN3/160 without FPA). Acceleration technique in Ohta and Maekawa (1987) is used for ray-object intersection. It is observed that though ray-bound tracing is more time consuming than simple ray tracing, total computation time is still reduced. As the last column shows, computation time can be shortened further by using a ray bound containing 16 pixels (arranged in a 4 by 4 pixel square). If a large ray bound is traced coherently, further coherence checking of individual pixels is unnecessary, thus reducing the total number of ray bounds traced.

Table 1. Computation time (s) of Fig. 4d

Resolution	Without anti-aliasing	Anti-alias all pixels	Use external coherence	Full ray-bound tracing	4 * 4 ray bound tracing
128 * 128	31.1	501.5	485	279.3	207.6
512 * 512	469.6	7990.1	7835.3	3440.9	1758.3

Table 2. Computation time (s) of Fig. 5d

Resolution	Without anti-aliasing	Anti-alias all pixels	Full ray-bound tracing	4 * 4 ray bound tracing
128 * 128	26.9	439.6	199.3	159.5
512 * 512	295.4	4913.7	2382.1	1536.9

5 Concluding remarks

The concepts of ray bound and ray-bound tracing are introduced to perform perfect and efficient anti-aliasing with ray tracing. Rather rough (by a circle and a sphere) ray-bound tracing greatly eliminates unnecessary anti-aliasing.
Ray-bound tracing is applicable to various shading algorithms including shadowing and reflection.
Although examples used in this paper treat only spheres as object shapes, the method is easily extended to treat quadric surfaces and relatively flat surfaces as shown in Appendices B and C. Moreover, techniques of interval analysis (Toth 1985) seem to be applicable to trace ray bounds against bi-cubic patches.

Acknowledgement. The authors appreciate fruitful discussions with Hiroshi Morishima, from which the original concept of ray bound is derived, and various suggestions from Dave Gordon, which greatly improved the quality of the paper. The authors are also grateful to Taiyo Kikaku Co. for willingly offering their equipment for this research.

References

Amanatides J (1984) Ray tracing with cones. Comput Graph 18:129–135
Arvo J, Kirk D (1987) Fast ray tracing by ray classification. Comput Graph 21:55–64
Cook RL (1986) Practical aspects of distributed ray tracing. In: Course Note no 12 Sigraph 86
Cook RL, Porter T, Carpenter L (1984) Distributed ray tracing. Comput Graph 18:137–145
Fujimoto A, Tanaka T, Iwata K (1986) Accelerated ray tracing. IEEE Comput Graph Appl 6:16–26
Heckbert PS, Hanrahan P (1984) Beam tracing polygonal objects. Comput Graph 18:119–127
Joy KI, Bhetanabhotla MN (1986) Ray tracing parametric surface patches utilizing numerical techniques and ray coherence. Comput Graph 20:279–285
Lee ME, Redner RA, Uselton SP (1985) Statistically optimized sampling for distributed ray tracing. Comput Graph 19:61–67
Ohta M, Maekawa M (1987) Ray coherence theorem and constant time ray tracing algorithm. Proc CG International '87, pp 303–314
Toth DL (1985) On ray tracing parametric surfaces. Comput Graph 19:171–179
Whitted T (1980) An improved illumination model for shaded display. Commun ACM 23:343–349

Appendix A

Intersection bound between a sphere and a ray bound

Because externally coherent and incoherent cases are easily detected, assume that a sphere and a ray bound intersect coherently as shown in Eq. A. 1. Then the problem is reduced to the simpler problem of intersecting a cone and a sphere, as shown in Eq. A.2. So, the following relations exist.

$$\boldsymbol{l}\cdot\boldsymbol{l}_0 \geq \cos\theta \tag{A. 1}$$

$$\boldsymbol{x}=\boldsymbol{o}+t\boldsymbol{l} \tag{A. 2}$$

$$|\boldsymbol{x}-\boldsymbol{c}|^2=R^2. \tag{A. 3}$$

From Eqs. (A. 2) and (A. 3),

$$t^2-2\boldsymbol{l}\cdot(\boldsymbol{c}-\boldsymbol{o})\,t+|\boldsymbol{c}-\boldsymbol{o}|^2-\mathrm{R}^2=0. \tag{A. 4}$$

With the restriction of (A. 1), the range of $2\boldsymbol{l}\cdot(\boldsymbol{c}-\boldsymbol{o})$ and then the range of the smaller root of (A.4) can be determined [in the coherent case, as in Eq. A. 1, (A. 4) always has two positive real roots]. So, let the maximum value of the smaller root be $t_{\max}$. Then, from the cosine theorem,

$$r^2=t_0^2+t_{\max}^2-2t_0t_{\max}\cos\theta. \tag{A. 5}$$

Thus, a radius of the intersection bound is calculated. The range of the direction of the normal vector of the intersection bound θ_n is then calculated as

$$\theta_n=2\sin^{-1}r/2R. \tag{A. 6}$$

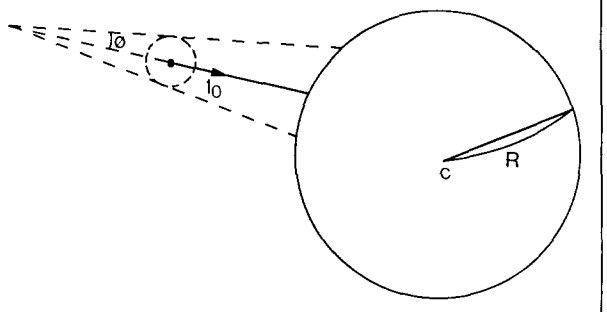

Fig. A 1. Intersection of a sphere and a ray bound

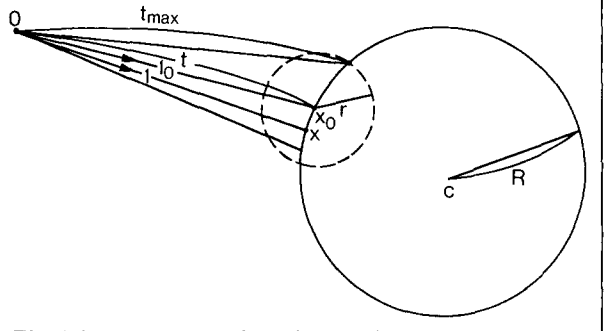

Fig. A 2. Intersection of a sphere and a cone

Appendix B

Intersection bound between a quadratic surface and a ray bound

A quadratic surface can be represented as:

$$^t(\boldsymbol{x}-\boldsymbol{c})\,A(\boldsymbol{x}-\boldsymbol{c})-r=0 \tag{B. 1}$$

where A is a $3*3$ symmetric matrix. A ray in a ray bound (with an origin bound centered at $\boldsymbol{a}_0$, radius of r_a and a direction bound centered at $\boldsymbol{l}_0$, angle of θ_l) can be represented as:

$$\boldsymbol{x}=\boldsymbol{a}+t\boldsymbol{l} \tag{B. 2}$$

$$|\boldsymbol{a}-\boldsymbol{a}_0|\leq r_a \tag{B. 3}$$

$$\boldsymbol{l}\cdot\boldsymbol{l}_0\geq\cos\theta_l. \tag{B. 4}$$

Intersection between them can be calculated by solving the Eqs. (B. 1) and (B. 2). In general, the problem is reduced to a quadratic equation:

$$at^2+bt+c=0. \tag{B. 5}$$

Using inequations (B. 3) and (B. 4), it is easy to obtain ranges of coefficients a, b, c in (B. 5). Then, it is a simple but tedious task to classify the solution of (B. 5). If (B. 5) has no positive real solution regardless of the possible values of its coefficients, the ray bound and the quadratic surface is externally coherent. If (B. 5) always has two positive real solutions or always has one positive real solution, the ray bound and the quadratic surface is surface coherent. Otherwise, no coherence can be observed by this method. In surface coherent cases, a bound of intersection points can be obtained using (B. 2) [with (B. 3) and (B. 4)] and normal vectors can be bounded by evaluating

$$A\boldsymbol{x}/|A\boldsymbol{x}|.$$

Appendix C

Intersection bound between a relative flat C_1 surface and a ray bound

The method in this appendix is applicable only to relatively flat C_1 surfaces. It is therefore suitable to treat an object defined with spline surfaces. It should be noted that, subdivided spline surfaces are generally more flat than the original spline surface.

A ray in a ray bound (with an origin bound centered at $\boldsymbol{a}_0$, radius of r_a and a direction bound centered at $\boldsymbol{l}_0$ angle of θ_l) can be represented as:

$$\boldsymbol{x}=\boldsymbol{a}+t\boldsymbol{l} \tag{C. 1}$$

$$|\boldsymbol{a}-\boldsymbol{a}_0|\leq r_a \tag{C. 2}$$

$$\boldsymbol{l}\cdot\boldsymbol{l}_0\geq\cos\theta_l. \tag{C. 3}$$

Assume a part of a C_1 surface is relatively flat and its normal vector is bounded (easy with spline surface) as

$$\boldsymbol{n}\cdot\boldsymbol{n}_0\geq\cos\theta_n. \tag{C. 4}$$

If the surface does not intersect with a ray with the origin of $\boldsymbol{a}_0$ and the direction of $\boldsymbol{l}_0$, no surface coherence can be observed by this method. So, try for the external coherence or, if the surface is a subdivision of a larger surface, change the subdivision of the original surface.

Even if the surface and the ray intersect at point $\boldsymbol{x}_0$, if there are two parallel vectors: one in the direction bound of a ray bound and the other in the direction bound of a normal vector of the surface, there is a possibility of internal silhouette edge, so no coherence can be observed. Further subdivision of the surface may improve the situation.

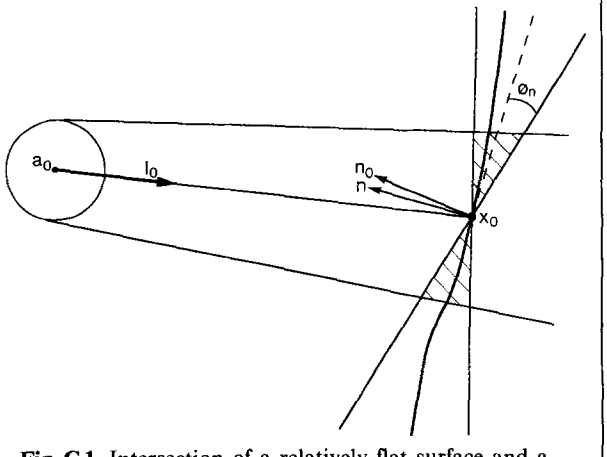

Fig. C 1. Intersection of a relatively flat surface and a ray bound

Otherwise, from Eq. C. 1, it is obvious that the ray-surface intersection occurs at most once and only in the dashed section. Thus, the bound of intersection point is easily calculated. To check that the ray-surface intersection occurs at least once, it is enough to check the external coherence of the ray bound and some bounding volumes (easily obtained for spline surfaces) of edges of the surface.

For the bound of normal vector, inequation (C. 4) can be directly used.

Appendix D

Ray bound after reflection

The radius of the origin of the new ray bound is simply a radius of the intersection bound. But the bounding of the new direction is much more complex. If an incoming ray with the direction $\boldsymbol{u}(|\boldsymbol{u}|=1)$ is reflected by a surface with the normal vector $\boldsymbol{n}(|\boldsymbol{n}|=1)$, the direction v of the outgoing ray is calculated as

$$\boldsymbol{v}=\boldsymbol{u}-2(\boldsymbol{n}\cdot\boldsymbol{u})\,\boldsymbol{n}. \tag{D. 1}$$

Let the center and the range of the direction of the incoming ray bound be u_0 and θ_u, those of the normal vector be n_0 and θ_n. Then,

$$\boldsymbol{u}_0\cdot\boldsymbol{u}\geq\cos\theta_u \tag{D. 2}$$

$$\boldsymbol{n}_0\cdot\boldsymbol{n}\geq\cos\theta_n. \tag{D. 3}$$

Let

$$\Delta\boldsymbol{u}=\boldsymbol{u}-\boldsymbol{u}_0 \tag{D. 4}$$

and

$$\Delta\boldsymbol{n}=\boldsymbol{n}-\boldsymbol{n}_0. \tag{D. 5}$$

From (D. 2) and (D. 3),

$$|\Delta\boldsymbol{u}|\leq\sin\theta_u/2 \tag{D. 6}$$

$$|\Delta\boldsymbol{n}|\leq\sin\theta_n/2. \tag{D. 7}$$

The center of the direction v_0 of the outgoing ray bound is approximated as

$$\boldsymbol{v}_0=\boldsymbol{u}_0-2(\boldsymbol{n}_0\cdot\boldsymbol{u}_0)\,\boldsymbol{n}_0. \tag{D. 8}$$

Then, the range of the outgoing direction θ_v is bounded as:

$$\cos\theta_v = \min \boldsymbol{v}_0 \cdot \boldsymbol{v}. \tag{D. 9}$$

By eliminating v and v_0 (D. 1) and (D.8) and then eliminating u and n, (D. 9) becomes

$$\begin{aligned}\cos\theta_v = \min\,(&1 + \Delta\boldsymbol{u}\cdot\boldsymbol{u}_0 + 2(\boldsymbol{n}_0\cdot\boldsymbol{u}_0)(\Delta\boldsymbol{n}\cdot\Delta\boldsymbol{u}) \\ &- 2(\Delta\boldsymbol{n}\cdot\boldsymbol{u}_0)^2 \\ &- 2(\Delta\boldsymbol{n}\cdot\boldsymbol{u}_0)(\boldsymbol{n}_0\cdot\Delta\boldsymbol{u}) - 2(\Delta\boldsymbol{n}\cdot\boldsymbol{u}_0)(\Delta\boldsymbol{n}\cdot\Delta\boldsymbol{u}) \\ &+ 4(\Delta\boldsymbol{n}\cdot\boldsymbol{n}_0)(\boldsymbol{n}_0\cdot\boldsymbol{u}_0)(\boldsymbol{n}_0\cdot\boldsymbol{u}_0 + \Delta\boldsymbol{n}\cdot\boldsymbol{u}_0 + \boldsymbol{n}_0\cdot\Delta\boldsymbol{u} \\ &+ \Delta\boldsymbol{u}\cdot\Delta\boldsymbol{n})).\end{aligned} \tag{D. 10}$$

So, a safe value of θ_v is,

$$\begin{aligned}\theta_v = \cos^{-1}\,(&\cos\theta_u - 2|\boldsymbol{u}_0\cdot\boldsymbol{n}_0||\Delta\boldsymbol{n}||\Delta\boldsymbol{u}| - 2|\Delta\boldsymbol{n}|^2 \\ &- 2|\Delta\boldsymbol{n}||\Delta\boldsymbol{u}| - 2|\Delta\boldsymbol{n}|^2|\Delta\boldsymbol{u}| \\ &- 4(1-\cos\theta_n)|\boldsymbol{u}_0\cdot\boldsymbol{n}_0|(|\boldsymbol{u}_0\cdot n_0| + |\Delta\boldsymbol{u}| + |\Delta\boldsymbol{n}| \\ &+ |\Delta\boldsymbol{n}||\Delta\boldsymbol{u}|))\end{aligned} \tag{D. 11}$$

$(|\Delta\boldsymbol{u}| = 2\sin\theta_u/2,\ |\Delta\boldsymbol{n}| = 2\sin\theta_n/2).$

Thus, using (D. 6) and (D. 7) the new ray bound is calculated.

MASATAKA OHTA received B.S. in computer science 1982 and then M.S. in 1984 from the University of Tokyo. At the same time, he engaged in creating computer graphics systems and computer generated images at Life Structure Institute, Seibu Digital Communications and FROGS. He is presently a research associate at the Computer Center of Tokyo Institute of Technology. Ohta is a member of ACM and the Information Processing Society of Japan.

MAMORU MAEKAWA received the B. Eng. degree in applied mathematics and physics from Kyoto University, Kyoto, Japan in 1965, and the M.S. and Ph. D degrees in computer science from the University of Minnesota, Minneapolis in 1971 and 1973 respectively.
From 1965 to 1970, he worked for Toshiba designing several operating systems and taught at the Department of Computer Science, university of Iowa from 1973 to 1975. In 1975, he joined the Toshiba Research and Development Center. He is presently an Associate Professor with the Department of Information Science, Faculty of Science, University of Tokyo, Tokyo, Japan. During 1981–1982 academic year, he was on leave at the Department of Computer Sciences, University of Texas at Austin. Dr. Maekawa is a member of ACM, IEEE, the Information Processing Society of Japan, and the Institute of Software Science.