Efficient Algorithms for Local and Global Accessibility Shading

Gavin Miller
Apple Computer, Inc.
1 Infinite Loop, MS 301-3J, Cupertino, CA 95014
gspm@apple.com

ABSTRACT
This paper discusses the use of two different approaches for computing the "accessibility" of a surface. These metrics characterize how easily a surface may be touched by a spherical probe. The paper also presents various acceleration techniques for accessibility. The idea of surface accessibility is extended to include "global accessibility" which measures the ability of a spherical probe to enter a structure from outside as well as to fit locally on the surface. The visual effect of shading using accessibility is shown to resemble the patina on certain tarnished surfaces which have then been cleaned.

Keywords: Surface accessibility shading, visualisation, aging.

1.0 Introduction

Most synthetic images look as though they are pictures of "new" objects. Visual cues to the aging of man-made products include the accumulation of dust on surfaces, the oxidation of metals (also known as tarnish), and the accumulation of dents and scratches. To counteract the effects of the aging process, many objects in the real world are cleaned and polished. This involves moving a soft rag over dust covered objects, or an abrasive rag over tarnished metals. Typically the cleaning surface, such as the rag, is unable to reach into sharply concave or otherwise inaccessible areas. Concave corners and creases tend to retain dirt or tarnish. To render these effects convincingly, it is necessary to formulate a mathematical definition of how accessible such regions are.

Fortunately, researchers in molecular modeling have long been concerned with defining surface "accessibility", since many chemical reactions depend on the access of one molecule to the surface of another. In Lee and Richards '71, the problem was first addressed in terms of a "solvent-accessible surface." (The original definition was replaced with the following one in Richards '77.) The solvent molecule is considered to be a spherical probe of fixed size, and is not allowed to intersect any of the original surface. The solvent accessible surface is the boundary of the volume which may be occupied by the solvent molecules. In geometrical terms the solvent accessible surface is found by offsetting the molecule outward by the probe radius and then offsetting inward by the same amount. In regions of concavity, the first offset surface will be self-intersecting. By trimming away the self-intersecting regions, the true offset envelope may be found. Because of this trimming, offsetting inwards does not restore the original surface. Instead, a constant radius fillet is introduced. In the case of a molecule with just two atoms, the fillet will be a portion of a torus. See Figure 1.

Connolly '83 described a graphics system for displaying solvent-accessible surfaces in real-time using a vector representation and for rendering shaded surfaces on a raster display. Edelsbrunner and Mucke '92, defined three-dimensional "alpha shapes" which, like solvent-accessible surfaces, are defined as the boundary of a volume inaccessible to a spherical probe of a given size. They simplify the resultant surface by "substituting straight edges for the circular ones and triangles for the spherical caps." They present an algorithm for creating approximate solvent-accessible surfaces as polygonal meshes.

SIGGRAPH '94, July 24-29, Orlando, Florida
© ACM 1994 ISBN: 0-89791-667-0 ...$5.00

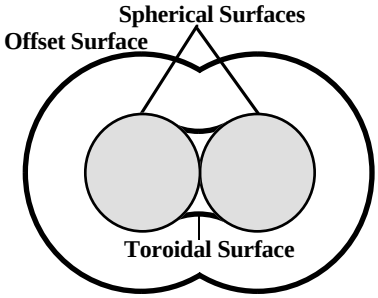

Figure 1. The solvent-accessible surface for a pair of spheres.

Unfortunately, we wish to shade the original surface with a measure of accessibility, rather than create a new surface. Kuhn et al '92 described a way to do this. They defined surface accessibility to be the radius of a sphere which may touch the surface tangentially and not intersect any other surface. Kuhn's definition will be referred to in this paper as "tangent-sphere accessibility". Kuhn et al '92 only described the computation of tangent-sphere accessibility when the original surface is a collection of spheres.

In Connolly '86, a different measure of surface shape is described. A sphere is centered on the surface point and the fraction of the sphere enclosed by the surface is a measure of the surface's curvature. The size of the sphere can be changed, effectively changing the filter size for the surface. While this is a useful shape analysis tool, it is not the best analogy for the tarnish cleaning process. Related work in Bloomenthal '91 creates smooth surfaces by convolving a binary volume representation with a Gaussian kernel and then finding the iso-surface. If the convolution value was used to shade the original surface, the results would be very similar to Connolly '86.

Work has also been described in the molecular modeling literature to determine whether solvents can reach inside a structure from outside. Connolly '91 describes the construction of a "molecular interstitial skeleton". The resultant skeletal structure gives an exhaustive measure of the radius of a sphere which could reach into interstitial regions. Unfortunately, the algorithm is restricted to surfaces made up of spheres, but it does test global accessibility for every size of probe simultaneously. For certain applications, however, measuring global accessibility for a probe with a fixed radius is quite satisfactory. Richards '79 described how to use a cube-connectedness algorithm to determine accessible regions within a structure when using a voxel representation.

2.0 Tangent Sphere Accessibility

We start by defining "tangent-sphere accessibility" as the radius of a sphere which may touch a surface point and not intersect any surface. The size of such a sphere will depend on the surface curvature (Figure 2) and the proximity of other surfaces (Figure 3). Note that the radius goes to zero where two surfaces intersect.

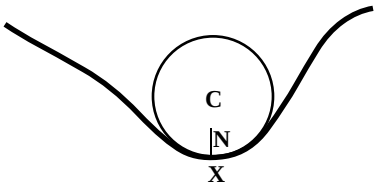

Figure 2. A sphere touching a surface tangentially with the maximum allowed radius.

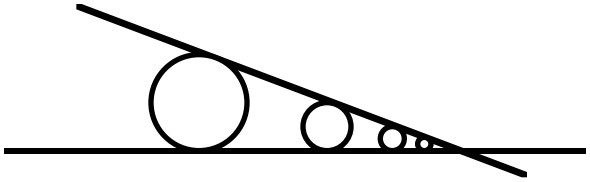

Figure 3. Accessibility for two intersecting planes.

2.1 Accessibility in the Presence of Other Geometry

When computing accessibility at a particular surface point, the surface normal and position is known. The accessibility could be computed by iterating on the sphere radius using a sphere-object intersection test (Figure 4). Unfortunately, this approach would be very time consuming since the intersection tests need to be done once per iteration. A second approach is to compute the accessibility directly from the geometry.

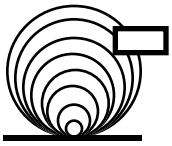

Figure 4. Nested tangential spheres

2.2 Accessibility in the Presence of Spheres

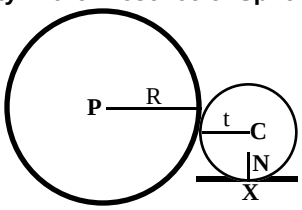

Figure 5. Accessibility in the presence of a sphere.

The accessibility, when the other surface in question is a sphere, is illustrated in Figure 5. The center of the tangential sphere C lies along the line formed by the surface position X, and the surface normal vector N. The radius of the tangential sphere is called t. This relationship is given by Equation (1).

$$\mathbf{C} = \mathbf{X} + t\,\mathbf{N} \tag{1}$$

For the tangential sphere to also touch the other sphere, whose center is P and whose radius is R, Equation (2) must also hold.

$$(\mathbf{C} - \mathbf{P}) \cdot (\mathbf{C} - \mathbf{P}) = (R + t)^2 \tag{2}$$

By substituting for C and then solving for t, it turns out that t can be computed directly using Equation (3).

$$t = \frac{R^2 - (\mathbf{X} - \mathbf{P}) \cdot (\mathbf{X} - \mathbf{P})}{2\,(\mathbf{N} \cdot (\mathbf{X} - \mathbf{P}) - R)} \tag{3}$$

If t is found to be less than zero, then the normal must point away from the other sphere enough that the accessibility is actually infinite. The two cases of finite and infinite accessibility are illustrated in Figures 6 and 7.

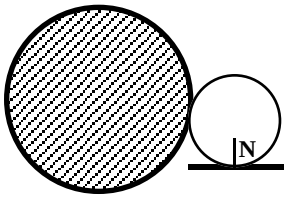

Figure 6. Finite accessibility in the presence of a sphere.

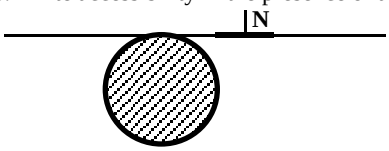

Figure 7. Infinite accessibility in the presence of a sphere.

The accessibility at a surface point is a value which can range between zero and infinity. To use this value as part of a shader, the accessibility needs to be used to generate a blend value between zero and one. This may be done using a linear or polynomial function, or using a negative exponential function. If there are more than two objects in the scene, the resultant accessibility is defined as the minimum of the accessibility values computed for each of the other objects in turn.

2.3 Accessibility in the Presence of Lines and Cylinders

Once again, the center of the tangential sphere is given by Equation 1. Given the center of the sphere C, we want to find the distance between this center and the nearest point on the axis of a cylinder. We define the cylinder to have a center P, a length l, and an axis unit vector L.

The nearest point to the sphere on the cylinder axis Q is defined by Equations (8) and (9).

$$u = (\mathbf{C} - \mathbf{P}) \cdot \mathbf{L} \tag{8}$$

Where u is the distance along the axis vector from the center of the cylinder and

$$\mathbf{Q} = u\,\mathbf{L} + \mathbf{P} \tag{9}$$

The condition that the tangential sphere also just touches the cylinder is given by Equation (10).

$$(\mathbf{C} - \mathbf{Q}) \cdot (\mathbf{C} - \mathbf{Q}) = (t + R)^2 \tag{10}$$

Where R is the radius of the cylinder.

Substituting in Equation (10) for C using Equation 1 and for Q using Equation 9 leads to a quadratic in t.

For the sake of clarity we define intermediate variables V and W in Equations (11) and (12).

$$\mathbf{V} = \mathbf{N} - (\mathbf{N} \cdot \mathbf{L})\,\mathbf{L} \tag{11}$$
$$\mathbf{W} = ((\mathbf{X} - \mathbf{P}) \cdot \mathbf{L})\mathbf{L} - (\mathbf{X} - \mathbf{P}) \tag{12}$$

Then the quadratic in t is of the form given by Equation (13).

$$a\,t^2 + b\,t + c = 0 \tag{13}$$

Where

$$a = 1 - \mathbf{V} \cdot \mathbf{V} \tag{14}$$
$$b = 2\,(R + \mathbf{V} \cdot \mathbf{W}) \tag{15}$$
$$c = R^2 - \mathbf{W} \cdot \mathbf{W} \tag{16}$$

The value of t is given by Equation (17).

$$t = \frac{-b \pm (b - 4\,a\,c)^{\frac{1}{2}}}{2\,a} \tag{17}$$

The smallest positive value of t is used. If a is very close to zero, then Equation (18) is used.

$$t = \frac{-c}{b} \tag{18}$$

So far we have treated the cylinder as if it was of infinite length. For a cylinder of finite length we need to take the end points into consideration. To see if the tangential sphere touches the cylinder within the end caps, we use the computed value of t to find the center of the tangential sphere C using Equation 1. The value of C is then used in Equation (8) to compute a value of u, the distance parameter along the cylinder axis. If the absolute value of u is less than half the cylinder length, then the value of t is valid. If this condition fails, then we ignore t and compute the accessibility for a sphere placed at the nearest end of the cylinder. This assumes that cylinders have spherical end caps.

2.4 Accessibility in the Presence of Polygons

To compute the accessibility for a surface element in the presence of polygons, we first consider the accessibility computation for an unbounded infinite plane. This geometry is illustrated in Figure 8.

Once again, the center of the tangential sphere is given by Equation 1. The infinite plane has a point P and a plane normal M. The distance between the infinite plane and the center of the tangential sphere must be equal to the radius of the tangential sphere, as defined by Equation (19).

$$t = (\mathbf{C} - \mathbf{P}) \cdot \mathbf{M} \tag{19}$$

Substituting for C using Equation 1 gives the value of t in Equation (20).

$$t = \frac{(\mathbf{X} - \mathbf{P}) \cdot \mathbf{M}}{(1 - \mathbf{N} \cdot \mathbf{M})} \tag{20}$$

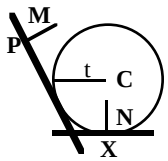

Figure 8. Accessibility in the presence of an infinite plane.

In the case of a finite convex polygon, the value of t is then used to compute Q, which is the point on the infinite plane which is closest to the center of the tangential sphere. This is done in Equation (21).

$$\mathbf{Q} = \mathbf{X} + t\,(\mathbf{N} - \mathbf{M}) \tag{21}$$

The point Q is then tested to see if it lies within the edges of the polygon. If it does, then the value of t is used. If it lies outside the boundary of the polygon, the edges and vertices of the polygon are used to compute the tangential radius. The latter computation is done by treating the polygon edges as cylinders of zero radius, and the polygon vertices as spheres of zero radius. For the case of a convex polyhedron, some of the edge and vertex computation only needs to be done once for the whole object, rather than for each polygonal face. Color Plate 1 shows a rectangular box intersecting with a sphere.

A "clock" made from a large number of spheres, cylinders and polygons is shown in Color Plate 2. Color Plate 2a shows the accessibility shading value and Color Plate 2b shows a diffusely shaded version of the clock. The accessibility shading is used to blend between the shaded surface and black (Color Plate 2c.) Color Plate 3 shows the combination of accessibility shading and environment mapping.

3.0 Efficient Tangent-Sphere Accessibility

As the number of objects increases, the cost of computing the tangent-sphere accessibility increases. For the naive algorithm, the cost is linear with the number of objects and proportional to the number of visible pixels. One way to reduce this cost is to reduce the number of objects used to compute the accessibility. This may be achieved by reducing the computation of accessibility to objects which lie within a certain spatial bound. This bound is then used to compute a list of potentially overlapping surfaces. This is an area which has been the study of computational geometers and a number of schemes have been suggested in the literature [Preparata '85].

3.1 Pixel-to-Pixel Coherence

A second class of optimization is possible using pixel-to-pixel coherence. Pixel-to-pixel coherence is like scan-line coherence in that results of the previous pixel are used to accelerate the computation for the current pixel. However, pixel-to-pixel coherence, as presented here, may carry over from one scan-line to the next and even between different primitives if they are close enough together, and have a similar enough normal vector.

For two adjacent pixels, the tangential spheres will largely overlap. Put another way, if we slightly enlarge the bound for one of the spheres, it will contain the sphere for the adjacent pixel. This coherence may be used by always employing an enlarged or "sloppy" bound to find the list of overlapping objects. For the next pixel, the new sphere is tested against the old sloppy bound. If the new sphere fits entirely within the old bound, then the old list of overlapping surfaces may be used for this pixel. This process continues until a sphere occurs which does not fit within the sloppy bound. In that case, a new sloppy bound is computed for the current tangential sphere, along with a new list of overlapping primitives. The sloppy bound is given by expanding the exact bound by a "slop margin".

If the chosen slop margin is too large, the returned list will have many surfaces which do not overlap the original bound. If the margin is too small, then the list will have to be recomputed from scratch for a high proportion of adjacent pixels. The optimal margin will depend on the resolution of the image. Coherence will increase as the image resolution increases, so a smaller margin will be optimal for larger images.

For the purposes of an initial evaluation, the clock image was rendered using tangent-sphere accessibility shading at a resolution of 128 by 256 pixels. The strategy used to find overlapping bounds was simply a linear search of the entire model. The slop margin was expressed in terms of a multiple of the maximum accessibility value which could affect the shading. Figure 9 shows the results of using differing values for the slop margin. (The timings are in seconds on an Apple Macintosh Quadra™ 900, which has a 25 MHz Motorola 68040.) The timings are for the portion of the computation concerned with finding the accessibility.

As expected, the rendering time decreased when the slop margin was increased from zero. The rendering time increased again when the margin of slop was very large. Most savings occurred for a margin of about four. This is unexpectedly large, and may be seen as a result of the rather low resolution of the image. The maximum savings due to using a sloppy bound for this image size was about a factor of six to one.

The results for an image resolution of 256 by 512 show that the minimum time occurs for a margin of error of about two. The greater degree of coherence means that a smaller margin will still query the database less frequently. The increase in speed due to the use of a sloppy bound for this higher resolution image is about ten to one. In general, the use of a sloppy bound becomes more effective as the image resolution increases.

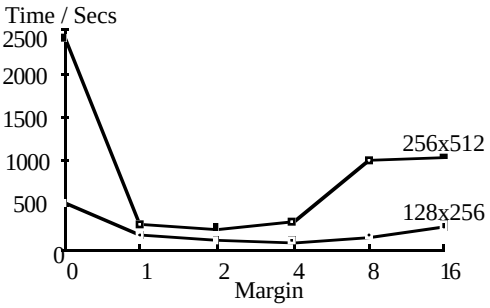

Figure 9. A graph of the rendering time vs. slop margin.

3.2 Limitations of Tangent-Sphere Accessibility

Unfortunately, tangent-sphere accessibility has a few computational disadvantages. Concave creases between adjacent polygons in a mesh will always have an inaccessible region. As the polygons become more parallel, the inaccessible region shrinks in size but never disappears. This leads to the very fine black lines and aliased "dots" visible in Color Plate 4. Unfortunately, this means that a smooth surface may not be approximated by a polygonal mesh in order to compute the accessibility. To overcome this problem, accessibility must either be computed for a mesh of normal-continuous patches, or redefined so that it is more forgiving of polygonal mesh approximations.

4.0 Surface Shading Using Offset-Distance Accessibility

An alternative definition for accessibility is to say it is equal to the distance to the nearest point on an offset of the surface, minus the offset radius. This definition is analogous to using a spherical cleaning device with an outer layer of abrasive bristles, as is shown in Figure 10.

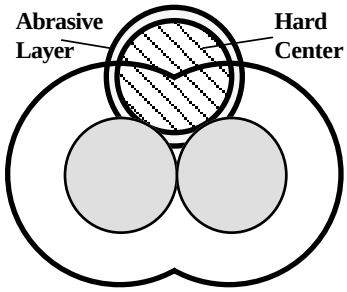

Figure 10. Spherical cleaning probe with abrasive layer.

In convex regions with no intersections, the distance is equal to the offset radius. In concave regions this value will increase depending on how inaccessible the surface is.

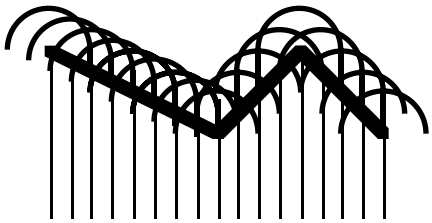
Figure 11. Positive offset of a height-field.

If a height field is represented by samples in a rectangular buffer A, then an offset of the surface may be computed in a second buffer B as follows:

• Buffer B is initialized to have the minimum value of buffer A at every sample.
• For each sample point in Buffer A, a sphere centered at that pixel's depth and location is scan-converted into Buffer B taking the maximum of the existing value and the value for the sphere.

This process is illustrated in Figure 11. To compute the offset-offset surface, the offsetting is now done in reverse:

• Buffer C is initialized to have the maximum value of buffer B at every sample.
• For each sample point in Buffer B, a sphere centered at that pixel's depth and location is scan-converted into Buffer C taking the minimum of the existing value and the value for the sphere. pixel.
• The distance to the nearest point on the offset-offset surface may then be approximated by taking the difference between Buffer C and the original height field in Buffer A.

This series of steps is called the offset-offset difference algorithm. It may be optimized by storing the depth of the offset sphere samples in a table.

An alternative algorithm is to compute the distance to the nearest pixel in the offset buffer (Buffer B) for every pixel in the original height field (Buffer A). At first glance this nearest distance calculation might require searching every pixel in Buffer B. This can be avoided by noting that the 3-D distance to the offset pixel is always greater than the 2-D distance in the plane of the buffer. Thus the pixels are tested in a spiral outward from the location of the sample in Buffer A, until the 3-D distance to the nearest sample in the offset buffer is closer than the radius of the spiral. In practice the spiral is a rectangular spiral aligned with the sample grid. This second approach is called the spiral search algorithm. If an upper limit is placed on distance values of interest, then the search may be terminated when the radius of the spiral reaches this distance value.

The offset-offset difference algorithm maintains the depth values at higher precision than the horizontal resolution of the buffer. Since, for certain applications, much of the important detail is in small variations in depth, this algorithm is preferable to the spiral search algorithm. It does slightly over-estimate the distance to the nearest point in the offset surface, but the errors are acceptable.

Color Plate 5 shows the offset-offset difference algorithm applied to a depth map. The depth map has been Lambert shaded, the accessibility computed, and then the results of the two calculations multiplied together. Color Plate 6 shows the accessibility map of a flower molding. By doing a color fill on the background, and then inverting the intensities, the impression of a wood-block print is created.

It is possible to use the same algorithm to approximate the accessibility for a surface when it is a mesh other than a height field. Color Plates 7 and 8 show Cyberware scans of human heads shaded using offset-offset difference accessibility. (The Cyberware scans were in the form of radial displacements to a cylindrical surface.) The banding artifacts on the neck in Color Plate 7 are in the original data.

The effect of changing the offset radius is compared in Color Plate 9. The display shading range was kept in a constant ratio to the offset radius. As the offset radius was increased, the border around the nose increased and the interior of the mouth vanished completely in the last two images. The times taken for the algorithm are shown in Table 2. They increased as the square of the offset radius, as was to be expected.

Offset Radius/pixels	Time/seconds
1	16.3
2	35.8
4	112.4
8	407.1
16	1547.6
32	5925.7

Table 2: Offset-offset times for increasing offset radius.

A disadvantage of the offset-offset difference algorithm is that it is parametric on the surface and does not compute the effect of intersections between different pieces of geometry. To compute offset-distance accessibility for intersecting objects, a new approach is required. If objects are represented as the boundaries of volumes stored in a voxel array, then offsetting operations may be applied to the volumetric representation as easily as for height fields. The model is considered to be contained within a three dimensional rectangular array of samples. These samples only need to be one-bit since they represent only the presence or absence of material.

To offset the volume in Voxel Buffer A, the following steps are taken:

• A second voxel buffer, Voxel Buffer B, is first initialized to be a copy of Voxel Buffer A.
• Then, for every voxel in Voxel Buffer A which has a value of one and has a zero neighbor, a sphere is scan-converted into Voxel Buffer B. The scan-conversion sets voxels which lie along the span to have a value of one.
• Voxel Buffer B then represents the offset volume.

To render the original surface with accessibility shading, the surface is scan-converted into a Z-buffer. For each surface point visible in the screen image, a corresponding point is found in the offset voxel buffer (Voxel Buffer B). This buffer is then searched in an expanding sphere until a zero-valued pixel is encountered. The distance between this zero-valued pixel and the original surface point is then used as the accessibility. The results of this algorithm are shown in Color Plate 10a. The shading range for the accessibility has been set to be equal to one voxel spacing. This was done to highlight the nature of the error in approximating the offset volume with a voxel map. An interesting pattern of Vornoi-diagram-like regions appears. These mark the areas for which the nearest zero-valued voxel in the offset surface is the same. In practice, the accessibility shading range is set to several voxel spacings and then the errors become acceptable.

Color Plate 11 compares parametric offset-offset difference shading, and volumetric offset-distance accessibility shading based on the voxel representation. The differences between the two techniques correspond intuitively with changing the offset radius slightly in different regions. From this experiment, it is clear that the results of the parametric and volumetric algorithms are not significantly different for the example chosen. The parametric technique, which is much faster, may be employed usefully on data-sets of this type.

As defined above, the volumetric offsetting algorithm takes time which increases as the cube of the offset radius. In practice this may be reduced to a quadratic dependence on the radius by using different representations for the voxel buffer. One approach is to represent the buffer as one bit numbers packed into long words. This means that 32 voxels may be scan-converted in one instruction. Logical operations on sets of voxels may be computed by applying the same operations to the long words which represent them. (Carpenter '84 used this approach successfully in implementing bit-mask operations as part of the A-buffer algorithm.) This representation does not eliminate the radius-cubed dependence theoretically, but in practice the offset radius is rarely set to more than one hundred or so voxel spacings, so computing the ends of the spans swamps the cost of scan-conversion.

A second possible representation, is to have a two-dimensional array of lists. For each X-axis aligned row in the voxel space, a list of transitions between one and zero could be stored. Scan-converting into such a list then only depends on the depth complexity of the list rather than the length of the span. It should also have memory savings for all but the most pathological volume sets, but might take longer to interrogate for the nearest zero-valued pixel compared to the optimized one-bit representation for practical ranges of offset radius.

A different technique for optimization is to compute the offset distance at voxel locations around the original surface, and then to interpolate the results for intermediate positions. The results of this algorithm are shown in Color Plate 10b. The errors are largely unchanged. A sparse voxel data-structure is needed to store the distance values for voxels which are adjacent to the original surface. Such a representation may also be used to render the surface multiple times when only the viewing transformation is changed. It is the voxel equivalent of storing the surface color in a parametric texture map.

The naive way to compute the distance to the nearest zero-valued pixel in the offset volume also increases as the cube of the offset radius. In practice this may be reduced to the square of the radius either by changing to the 2-D array of lists or by treating the 1-bit representation in slice order as follows:

• Loop from top to bottom for each horizontal slice of the voxel buffer.
• For each pixel in the horizontal slice, compute for each voxel the location of the nearest zero-valued pixel above it. This will either be the location of the nearest zero-valued voxel for the same location in the previous slice, or it will be the location of the current voxel (if it has a value of one).
For this slice we now have a 2-D array of samples which represent the height of the nearest zero-valued voxel above each slice voxel.
• For each slice voxel which is adjacent to the original surface, compute the distance to the nearest zero-valued voxel in the offset volume by using the spiral search algorithm applied to the 2-D array of samples. This corresponds to examining the entire offset voxel buffer above the surface point and finding the nearest non-zero voxel.
• The resultant accessibility value is placed in the sparse voxel representation.

To complete the calculation, the algorithm is repeated, but sweeping from bottom to top. This time the algorithm keeps track of the position of the nearest zero-valued pixel below the current slice. The distance from the top-down sweep is compared to the distance from the bottom-up sweep. The smaller of the two is the valid 3-D distance to the nearest zero-valued voxel in the offset volume. The original surface is then scan-converted using accessibility values interpolated from the sparse voxel representation. The times taken to compute volumetric accessibility for the head in Color Plate 10, for different offset radii are given in Table 3. The voxel set is 512 voxels on a side.

Offset Radius/pixels	Offset / secs	Shading / secs
3	298	217
6	599	404
12	1967	1045
24	8173	3396

Table 3. Volumetric accessibility times for different offset radii.

4.1 Global Accessibility using a Voxel Representation

Volumetric rather than parametric accessibility is clearly required if objects intersect in a scene. An example is shown in Color Plate 12.a It shows the effect of shading the volumetric accessibility as a red pigmentation. Locally inaccessible regions are emphasized. However, the interior of the structure is shown to be accessible, as is the small sphere in the center. This is because there is room inside the structure for a probe sphere to touch the surface without intersecting any other surface. To prevent these regions from being accessible, it is required to compute whether the probe sphere could enter the structure from outside. How to compute this is simplified by the observation that testing the intersection of a sphere with a volume is equivalent to testing a point at the center of the sphere against an offset of the volume, where the offset radius is the sphere radius.

Computing global accessibility merely requires that we replaced the test for the nearest zero-valued voxel in the offset buffer with a test for the nearest zero-valued voxel which is connected to a

point outside of the structure. This may be computed using a seed-fill algorithm applied to the offset volume. The resultant volume is then used for the offset distance calculation. For a 512-cubed voxel space, a simple seed-fill algorithm takes about 2 minutes on an Iris Crimson with a 66 MHz R4000 processor.

Color Plate 12b shows a global accessibility-based pigmentation for the same model. The small sphere and interior regions of the large spheres are now inaccessible, as expected. A more complex example is shown in Color Plate 13a, where a set of random cylinders has been shaded with diffuse shading. The local accessibility shading in Color Plate 13b shows white regions in the body of the set of cylinders. The global accessibility shown in Color Plate 13c eliminates these regions, providing a strong visual indication that such a structure would be impermeable to spherical probes of that size. Color Plate 13d shows the difference between the local and global accessibility values, and so highlights regions accessible to only those spheres which would be trapped inside the lattice.

5.0 Conclusions

A variety of definitions of surface accessibility were explored. New analytical results were derived for "tangent-sphere accessibility" in the presence of cylinders and polygons. An algorithm was also devised which accelerated the computation of tangent-sphere accessibility for large numbers of objects. However, tangent-sphere accessibility was found to be inappropriate for shading polygonal meshes.

A second definition of accessibility was introduced based on the distance to the nearest point on the offset of a surface. This algorithm yielded interesting visual effects which suggested an aged or tarnished appearance. The algorithm was extended to deal with general 3-D surfaces using an efficient voxel representation. The voxel-based approach was expanded to compute "global accessibility" which took into account the ability of a probe of fixed size to travel into a structure from outside.

6.0 References

Bloomenthal, Jules, "Convolution Surfaces", Computer Graphics, Vol. 25, No. 4, July 1991 pp 251-256.

Carpenter, Loren, "The A-buffer, An Anitialiased Hidden Surface Method", Computer Graphics, Vol. 18, No. 3, July 1984 pp 103-108.

Connolly, M., "Solvent-accessible surfaces of proteins and nucleic acids", Science, Vol. 221, No. 4612 (19 August 1983) p 303.

Connolly, M. L., "Measurement of protein surface shape by solid angles", Journal of Molecular Graphics, Vol. 4, No. 1, March 1986.

Edelsbrunner, Herbert and Ernst P. Mucke, "Three-dimensional Alpha Shapes", Proceedings of 1992 Workshop on Volume Visualisation, Boston, October 19-20, 1992.

Kuhn, Leslie A., Michael A. Siani, Michael E. Pique, Cindy L. Fisher, Elizabeth D. Getzoff, and Joan A. Tainer, "The Interdependence of Protein Surface Topography and Bound Water Molecules Revealed by Surface Accessibility and Fractal Density Measures", J. Mol. Biol., (1992) 228, pp 13-22.

Lee, B. and F. M. Richards, "The Interpretation of protein structures: estimation of static accessibility", J. Mol. Biol., Vol. 55 (1971) p 151.

Preparata, Franco P., Michael Ian Shamos, "Computational Geometry, An Introduction", Springer Verlag, 1985.

Richards, F. M., Annu. Rev. Biophysics. Bioeng. 6. 151 (1977)

Richards, F. M., "Packing Defects, Cavities, Volume Fluctuations, and Access to the Interior of Proteins, Including Some General Comments on Surface Area and Protein Structure", Carlsberg Res. Commun. 44, (1979) pp 47-63.

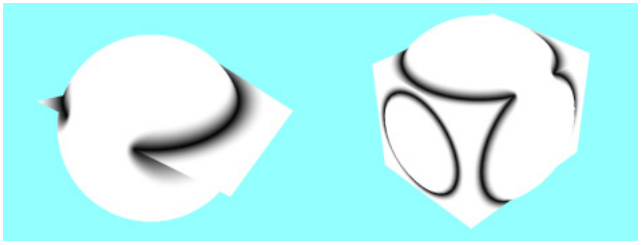

Plate 1. Tangent-sphere accessibility for a sphere intersecting a) a polygon, b) a rectangular box.

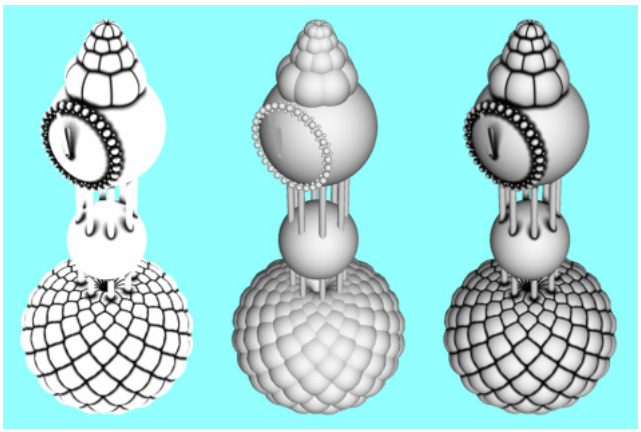

Plate 2 Clock model with a) Tangent-sphere accessibility b) Lambert Shading, c) Composite Shading.

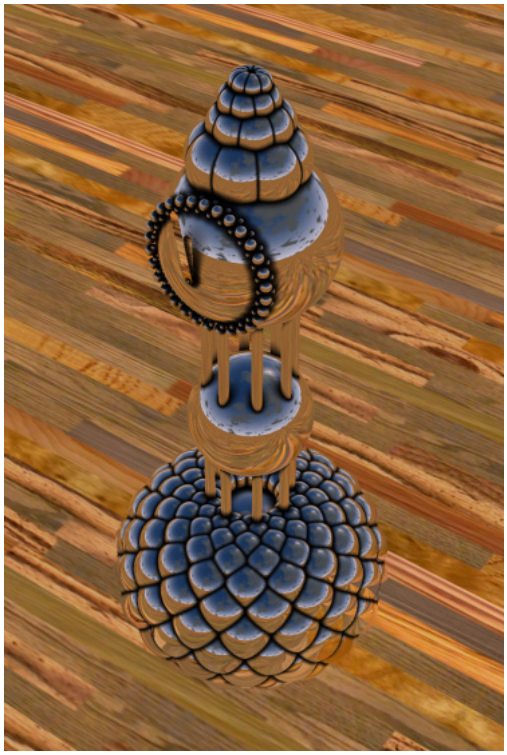

Plate 3: Tangent-sphere accessibility with environment mapping.

Plate 4: Polygonal Mesh with a) Tangent-sphere accessibility shading, b) Lambert shading, c) composite shading.

Plate 6: Flower mould as a) Offset-offset accessibility map, b) Inverted map with background fill.

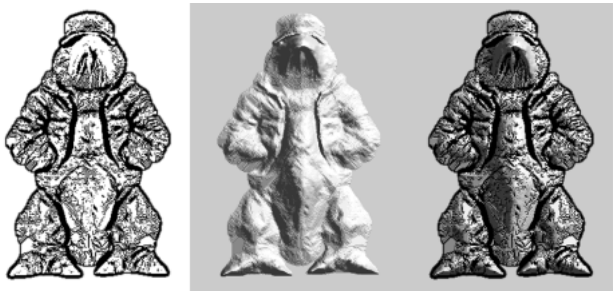

Plate 5: Height field with a) Lambert Shading, b) Offset-offset accessibility shading , c) Composite shading.

Plate 8: Greek Gods.

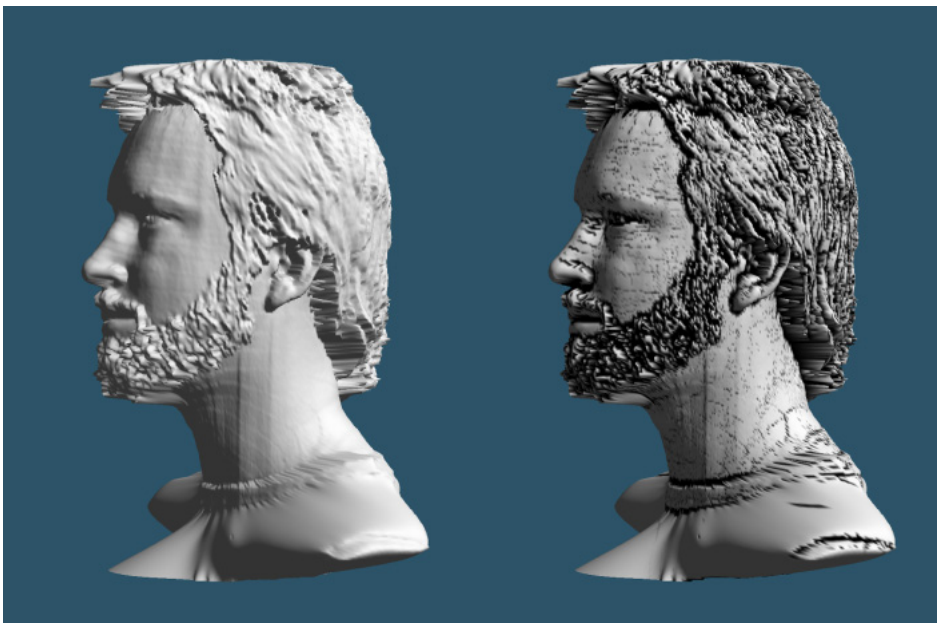

Plate 7: Cyberware scan with a) Lambert shading, b) Offset-offset accessibility shading.

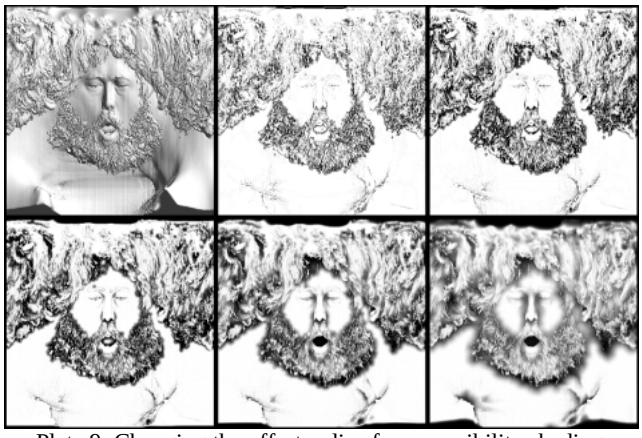

Plate 9: Changing the offset radius for accessibility shading.

Plate 11: a) Parametric Accessibility b) Volumetric Accessibility.

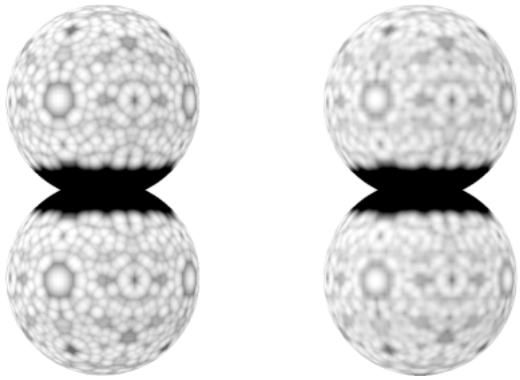

Plate 10: Error for voxel-based accessibility a) Computed at each pixel, b) Interpolated from lattice.

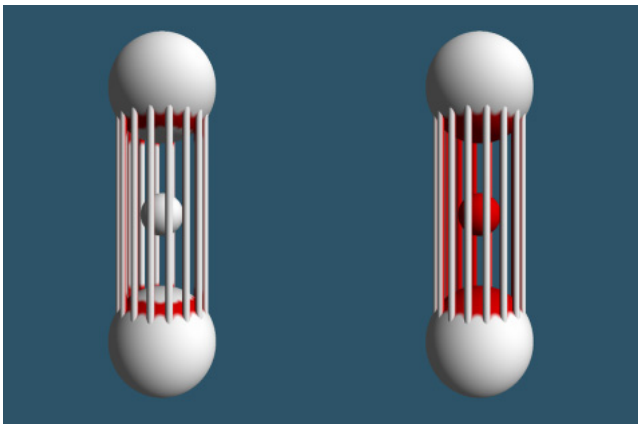

Plate 12: Cage Model as a) Local accessibility , b) Global accessibility.

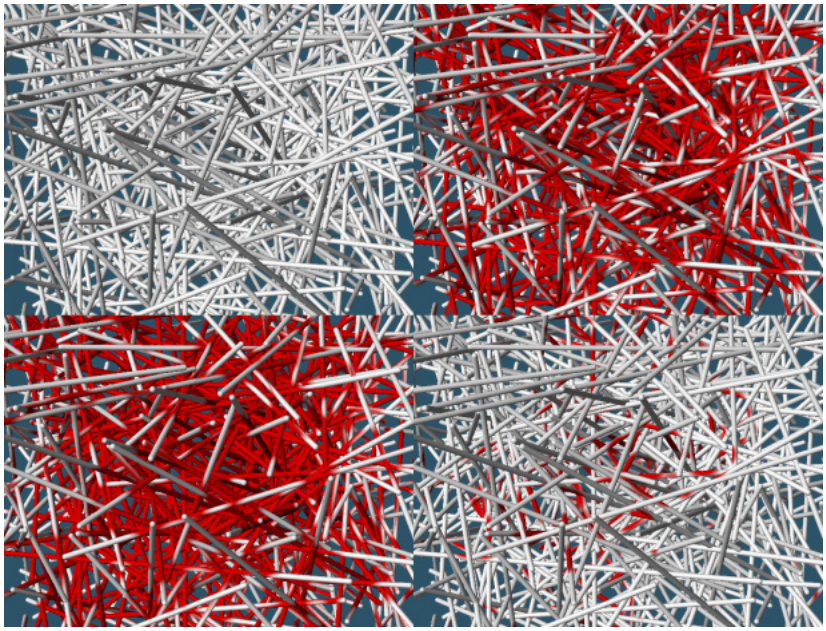

Plate 13: Random Cylinders as a) Lambert shading, b) Local accessibility, c) Global accessibility, d) Local and not global accessibility.