1984

Generating Shadows with an Umbra and Penumbra

Lynne Shapiro

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-84-12.

Generating Shadows with an Umbra and Penumbra

Abstract

An algorithm for generating shadows with an umbra and penumbra due to distributed light sources is presented. The method used is based on the inclusion of the shadow volumes in the object data processed by a depth-buffer visible surface computation. The standard depth-buffer data structure is modified to allow the handling of shadows with a depth-buffer hidden surface algorithm. The algorithm involves breaking the light source up into a set of point sources and superimposing the shadows generated by each point source to obtain the final shadow.

Disciplines

Computer Engineering | Computer Sciences

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-84-12.

GENERATING SHADOWS WITH AN UMBRA AND PENUMBRA

Lynne Shapiro
Norman I. Badler
MS-CIS-84-12

Department of Computer and Information Science
Moore School/D2
University of Pennsylvania
Philadelphia, PA 19104

1984

* Acknowledgement: NASA grant NAS9-16634 and NAS9-17239.

Generating Shadows With An Umbra And Penumbra

Lynne Shapiro
Norman I. Badler
Department of Computer and Information Science
Moore School/D2
University of Pennsylvania
Philadelphia, Pennsylvania 19104

Abstract

An algorithm for generating shadows with an umbra and penumbra due to distributed light sources is presented. The method used is based on the inclusion of the shadow volumes in the object data processed by a depth-buffer visible surface computation. The standard depth-buffer data structure is modified to allow the handling of shadows with a depth-buffer hidden surface algorithm. The algorithm involves breaking the light source up into a set of point sources and superimposing the shadows generated by each point source to obtain the final shadow.

Introduction

A shadow is a dark image cast on a surface by an object which blocks light. The use of shadows in computer generated pictures has historically been advocated for enhanced realism. Traditionally, a light source has been modeled as a point or a direction that specifies the parallel rays[1, 2, 3, 4, 5, 6, 7].

Early papers on shadow generation were presented by Appel[8, 1], and Bouknight and Kelly[2, 3, 9]. More recently, Crow[4] presented a shadow algorithm that involves calculating the surface enclosing the volume of space swept out by the shadow (umbra) of an object (see Figure 1). The shadow surfaces are then added to the data and treated as invisible surfaces. Assuming a point light source or a source at infinity (specified by a direction), the shadow surface is given by planes defined by the silhouette edges of the object and the light source position. Silhouette edges are those edges of the object which separate front faces (in which the outward normal to the face forms an acute angle with the vector to the light) from back faces (in which the outward normal to the face does not form an acute angle). If a visible point lies within the shadow volume, then the point is in shadow.

Atherton, Weiler and Greenberg[5] presented an approach to

shadow generation based on hidden surface elimination using polygon area sorting. Shadow descriptions are created by viewing the environment from the position of the light source. The hidden surface elimination algorithm is used to delineate "illuminated polygons," which are areas not shadowed. These polygons are added to the original environment and treated as surface details on their source polygons.

Another method described by Williams[6] allows display of shadows cast by objects modeled with smooth surface patches. Points are mapped from the observer's view to the light source view to determine if the point is visible from the light source. If the point is not visible, it is in shadow. Visibility is determined using a z-buffer visible surface algorithm.

In 1980, Whitted[7] presented an improved illumination model that allows simulation of shadows due to curved and polygonal surfaces. The algorithm uses "ray tracing" to accurately model reflection, refraction and shadowing.

In reality, a light source is not a single point; it has a distribution of finite size. Hall and Greenberg[10] recently presented a method for simulating the light distribution of linear light sources and light sources defined by surfaces of

revolution. However, they do not address the issue of shadowing due to such light sources. Work is currently being done with ray tracing algorithms that will be able to handle shadows with umbra and penumbra. The results of these efforts will be published in the proceedings of the 1984 SIGGRAPH conference this July.

Objects that intercept light from distributed sources cast shadows consisting of an umbra and penumbra[4, 11]. The umbra is that part of the shadow which is completely cut off from the light source. The penumbra is that part which receives some light from the source, but not all of it. The penumbra surrounds the umbra creating a smooth transition from dark to light areas.

We present an algorithm for generating shadows due to distributed light sources. The method employed extends Crow's shadow volume algorithm to produce shadows consisting of penumbras as well as umbras.

The light source model and shadow algorithms we describe are part of TEMPUS, a system developed at University of Pennsylvania for NASA Johnson Space Center to be used for the simulation of astronauts in a workstation environment. The major application of this system is to evaluate workstations in terms of the tasks which are to be performed

in them. The inclusion of shadows in computer generated scenes of workstation environments clarifies spatial relationships and illustrates degrees of visibility in the existing workstation.

A Modified Depth-Buffer

Objects to be displayed are created or read from data files using the SurfsUp (Surface System of the University of Pennsylvania) modeling system. SurfsUp represents objects as polyhedral surfaces. The objects are stored in a symbol table where they can be retrieved and used for input to the shaded graphics part of the TEMPUS system. (Note: Shadow rendering can be toggled on and off.)

A modified depth-buffer is used to store information necessary to determine visible surfaces and the shadowing of these surfaces. Each cell in the two-dimensional array represents a visible point that is described by a record. In addition to the standard *z* value, the following information is included in the record: an object pointer, the calculated normal, and two pointers (Sf,Sb) for shadows (see Figure2). Intensity values are ***not*** computed and stored during the tiling process, which is why the object pointer and normal fields are necessary. Although these additional fields increase the size of the buffer structure, they allow for the handling of transparency and shadowing with a depth-buffer algorithm.

To overcome memory limitations, a "multiple buffer" method is employed. That is, the screen is divided into multiple buffer boxes (e.g. for our 512x512 frame buffer, it is convenient to make the buffer boxes 64x64).

The object pointer points to a description of the polygon that is visible at this point. Only one description record is used for each polygon. Color, transmittance, glossiness, and reflectance attributes of a polygon can be accessed through the object pointer. The normal is that of the polygon at this point: possibly an interpolated value that is computed during the tiling process.

In order to determine whether or not a point is in shadow, it is necessary to maintain information about the shadow volumes that surround the point. The Sf pointer points to a list of records that describe front facing shadow polygons (those in which the outward normal to the polygon forms an acute angle with the vector to the eye). The Sb pointer points to a list of records that describe back facing shadow polygons (those in which the outward normal to the polygon does not form an acute angle). There is at most one record per light source maintained in each Sf or Sb list. Each record stores an identifier to indicate the light source that is being blocked and a "point source level." The records in the Sf list have additional fields for "object level" and "darkness level."

(The levels are described in a later section.)

Shadow volumes are computed and rendered after all visible objects have been rendered into the depth buffer. Preliminary intensity values are computed for the opaque objects in the image (taking into account any effects from shadows). Transparent objects are sorted by z values. One at a time the transparent objects are rendered into a temporary buffer and intensity values are computed. The computed intensity values are then used to attenuate the preliminary intensity values for every pixel where the transparent object is visible. The rendering, shadowing, and shading algorithms are repeated for each of the buffer boxes. When the intensity values of a buffer box have been computed, they are sent to the display.

The system is implemented in Pascal (with a bit of Fortran) and runs on a VAX 11/780 under VMS. A Grinnell GMR Series 27 (or similar frame buffer) is used to display shaded images.

Light Source Model

Distributed light sources are defined by intensity, geometric shape and location. To be consistent with the rest of the system, the shape of a light source is modeled as a polyhedral surface. Rectangular light sources can be easily modeled and

circular light sources can be closely approximated. Other less common shapes can also be modeled.

Intensity is represented as a (hue, saturation, value) triple. For simplicity, spectral intensity distribution is assumed to be constant for a particular light source. However, the algorithm could be extended to allow sources with Gaussian or other intensity distributions.

The geometric shape and location of a distributed light source is specified in world coordinates relative to the objects in the environment. The light rays emanate from the surface of the source.

Traditional point sources and sources at infinity can also be handled. A point source has an intensity value and an (x,y,z) location. The light source rays emanate from the point in all directions (similar to a bare light bulb). A source at infinity is represented by an intensity value and a single direction that specifies the parallel rays.

Mathematical Model Of The Penumbra Effect

The amount of light reaching any point in the penumbra region can be calculated by a summation[12, 11]. The light source is broken up into a set of point sources and the effects of each point source are superimposed.

Shadow Computation By Superposition

The method for generating a shadow by superposition consists of two parts. The first is the generation of a set of point sources that models the distributed light source. The second is the determination of the shadow volumes due to each point source and an object which blocks its light. These shadow volumes are then superimposed to produce the shadow cast by an object intercepting light from a distributed source.

A set of point sources must be computed that adequately models the effects of the distributed light source. Since light emanates only from the surface of the source, it is sufficient to take points that lie on the polyhedral surface. Points can be generated randomly or by superimposing a lattice of points at some resolution. Uniform random generation of points appears to be better than the use of a lattice as evenly spaced points may cause visual banding effects.

Generation of the point source set is done once for each light source when the light source object is created. This is a time/space tradeoff, but it is necessary when generating frames for animation since changing the point source set between frames would cause strange lighting effects. A sufficient number of points should be generated so that the addition of one more point to the set of point sources will not

visibly affect the shadow generated. However, this depends on how close the object is to the light source and how big the object is relative to the light source. It may be impossible to avoid banding in situations where the object is very close to the light source. Since the size of the point source set is a user-specified parameter, the user can exercise some control over the amount of banding that occurs. The total number of point sources generated is distributed among the faces of the light source surface such that the number of point sources in a face is proportional to the relative area of the face.

The silhouette of an object is computed relative to the light source. Therefore, each point source in the set "sees" a different silhouette and a different shadow volume is generated from each silhouette. (See[4] for details).

After all visible objects have been processed by the depth-buffer hidden surface algorithm the shadow volumes of every visible object (due to each light source) are determined and processed by the same algorithm. Each pixel that is surrounded by shadow volumes is marked using the Sf and Sb lists as described earlier. The number of shadow volumes that surround a pixel (the darkness level) tells how many points in the point source set of a distributed light source do *not* illuminate the visible pixel. For every pixel, there is one darkness level associated with each light source. However,

the darkness level represents the effects of all objects that reduce the amount of light reaching a pixel. The darkness level is stored in the front facing shadow description record for the associated light source. The amount of light illuminating a pixel from a distributed light source *i* is:

$$I_{att} = (I/n)(n - dl)$$

where

I is the total intensity of light source *i*,
n is the size of the point source set of light source *i*, and
dl is the darkness level at the pixel for light source *i*.

Consider the two-dimensional example in Figure 3. L is a distributed light source with intensity I that has been modeled as three point sources l1, l2, and l3. S is the silhouette of an object. For simplicity, assume that L is so small that l1, l2, and l3 all see the same silhouette (S). For this example, any point P that is in any of the areas A1 through A5 is in shadow. The amount of light that reaches such a point is I/3 x (3 - darkness level of P), where I/3 is the intensity per point source and (3 - darkness level of P) is the number of point sources that illuminate the point.

In order to compute the darkness level at a pixel, two other

levels must be computed first: the point source level and the object level. The point source level maintains information about the effects on a pixel of the shadow volume of a single point source in the point source set of light source i. The possible values for the point source level are zero (pixel is not contained in this shadow volume) and one (pixel is contained in this shadow volume). If a pixel is behind a front face of the shadow volume, the point source level of Sf_i is set to one. If a pixel is in front of a back face of the shadow volume, the point source level of Sb_i is set to one. When all faces of the shadow volume of a point source are rendered, the point source level values are used to determine if a pixel is shadowed from the point source. If the point source level of Sf_i equals one and the point source level of Sb_i equals one then the pixel is in shadow and the object level of Sf_i is incremented. The object level is used to keep a count of the number of point sources in a distributed light source i that are blocked from a pixel due to a single object.

When all shadow volumes (due to all light sources) of an object have been rendered, the object level is used to update the darkness level. For a given pixel, the darkness level is given by:

Darkness Level =
Maximum(Darkness Level, Object Level).

It is necessary to take the maximum value to ensure that 1) a

point source is not counted more than once, and 2) larger objects (which block out more point sources) prevail (see Figure 4). The point source level is reset to zero before rendering the shadow volume of a new point source. The object level is reset to zero before rendering the shadow volumes of a new object.

Below is Pascal psuedo-code for this algorithm:

Rendering:

```
For each visible object o
  Reset object level of all Sf/Sb
    for all light sources
  For each light source i
    For each point source ps in light source i
      Reset point source level of all Sf/Sb
        for light source i
      Generate silhouette of object o seen by
            point source ps
      Generate shadow volume of silhouette
      For each face in shadow volume
        For every pixel shadow face covers
          If shadow face affects this pixel
            then
               {i.e. Zsf_i < Zpixel
                     or Zsb_i > Zpixel}
            If there is no Sf_i (Sb_i) then
              Create Sf_i (Sb_i) with
                point source level = 1
                object level = 0
                darkness level = 0
            Else
              {there is an Sf/Sb for
               light source i, store
               effect of point source ps}
              point source level = 1
```

```
        {Update object level of point source i
         to include any effects from point
         source ps}
        For all pixels where
                (Sf_i <> NIL) AND (Sb_i <> NIL)
          If point source level of Sf_i = 1
            AND point source level of Sb_i = 1
            then Increment object level
                 of light source i
    {Update darkness level of pixel to
     include any effects of object o
     for all light sources}
    For all pixels
      If Sf <> NIL then
        For each Sf_i
          Darkness level of light source i =
          Max(Darkness level of light source i,
              Object level of light source i)
```

After all visible objects have been processed by the shadow rendering code, intensity values are calculated. The basic Phong[13, 14] shading model is used, however, the intensity of the distributed light source is attenuated by the darkness level. The intensity at a pixel is given by:

$$I = I_a k_a + \Sigma\,(I_{att}/(c + d))(k_d(\bar{L}.\bar{N}) + W(\theta)(\bar{R}.\bar{E})^n)$$

where

I_a is the intensity of the ambient light,
k_a is a constant indicating how much
ambient light is reflected from object's surface,
c is a constant,
d is the distance from the point to the light
source (if the light source is not at infinity),
k_d is the diffuse reflection coefficient of the surface,

$\bar{L}$ is the vector to the light source,

$\bar{N}$ is the unit surface normal,
$W(\theta)$ is the specular reflection coefficient
of the material,

$\bar{R}$ is the reflected ray,

$\bar{E}$ is the vector to the eye,
n is an exponent that represents the
glossiness of the surface, and
I_{att}, the attenuated intensity of the light source,
is the amount of light illuminating a pixel from a
distributed light source as previously defined.

In order to simplify the shading computations, the spatial intensity distribution of the light source is assumed to be constant. However, the algorithm could be modified to allow spatial intensity to be extracted from the point source set or to be described by goniometric diagrams[10].

Conclusions

The ablility to handle distributed light sources and the shadows generated by them improves models of various environments and enhances the realism of computer

generated images. However, these computations require considerable time and space, particularly since shadow polygons must be maintained in a modified depth-buffer. The time/space tradeoffs (e.g. the number of buffer boxes vs. the size of the buffer boxes) can be adapted to the amount of memory and processing power available to the given user.

Depth-buffer comparisions and ray tracing are likely to be the two main visible surface algorithms in the future. The algorithm presented in this paper can be used with ray tracing algorithms by treating shadow polygons as translucent polygons. The amount of light reaching the eye along a ray will have to be attenuated based on the number of shadow volumes the ray pierces.

The method used to construct silhouettes and shadow volumes depends upon the objects being convex, closed surfaces. The convexity restriction can be circumvented by splitting a non-convex object into one or more convex objects. The closed restriction can be circumvented by pasting back faces onto objects to ensure that an outward facing normal can be determined. It is possible to extend this algorithm to handle shadows generated by objects that are not polygonal (e.g. spheres) provided that the silhouettes of the objects can be determined and represented as polygons.

Acknowledgments

Special thanks to Dr. Henry Stark for pointing us in the right direction, Gary Sentman, Gerry Radack and David Brotman for their technical advice and support, and Mark Newhouse for his insights. We also wish to thank Jeri Brown, Barbara Woolford and Jim Lewis of NASA, whose appreciation and support with NASA contract NAS9-16634 made this research possible.

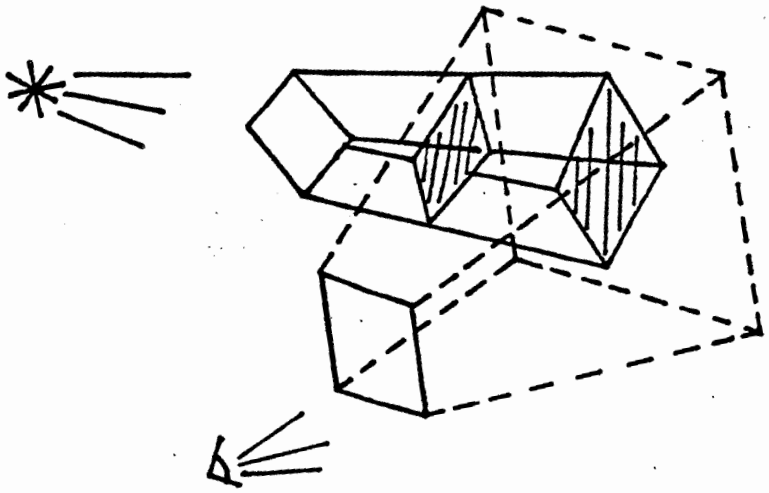


Figure 1: Shadow Volume

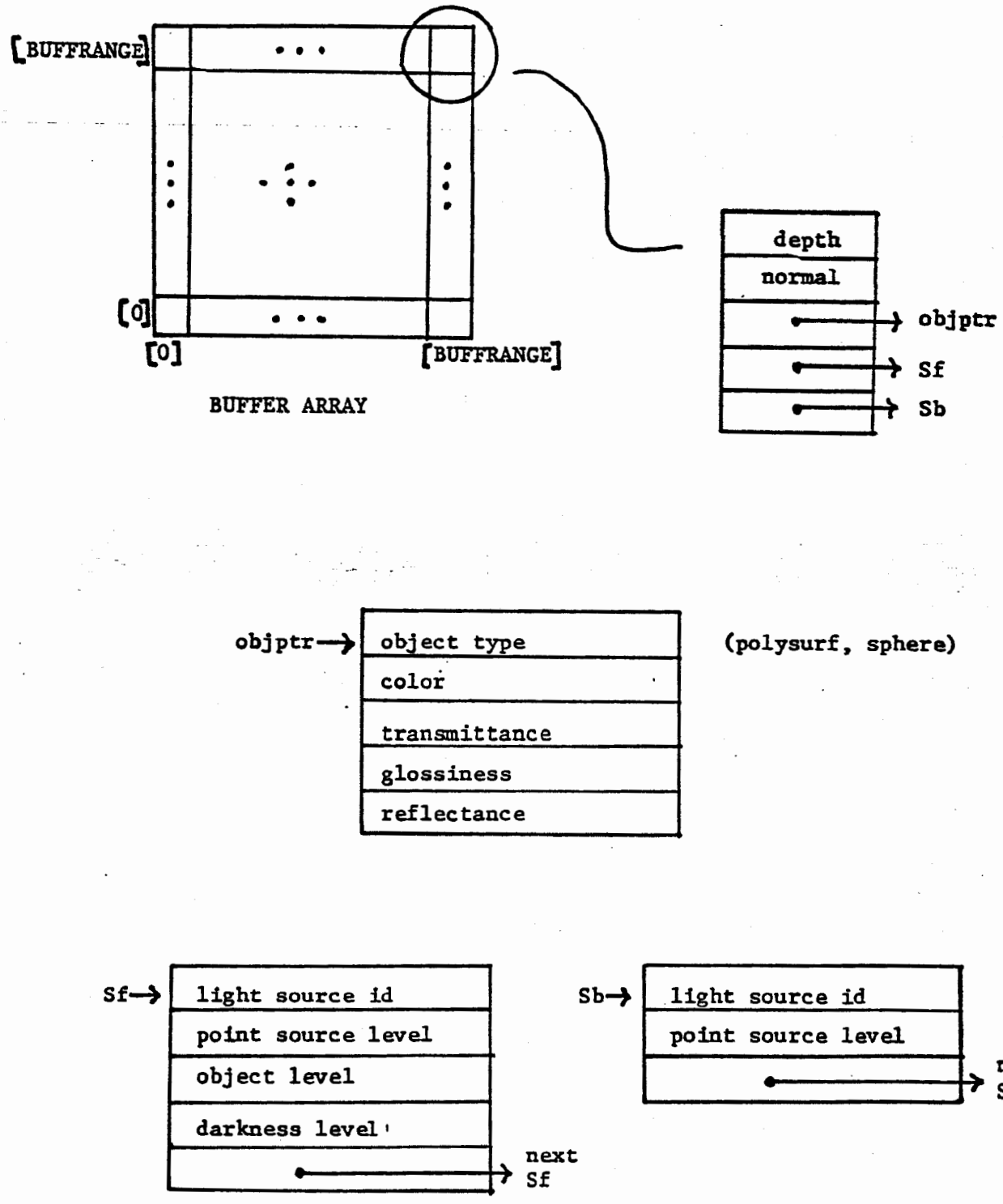
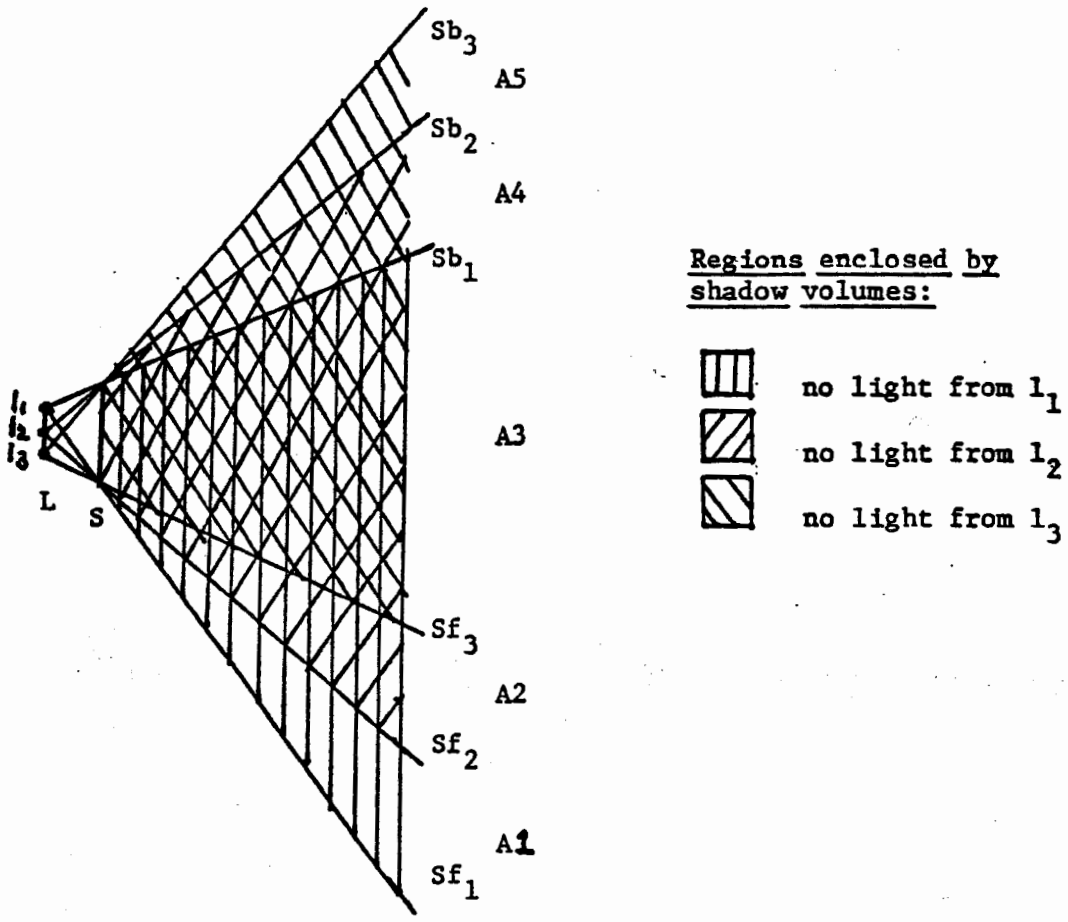


Figure 2: Modified Depth-Buffer



If point P is in:	P receives no light from:	Darkness level at P:
A1	l_1	1 } PENUMBRA
A2	l_1, l_2	2 } PENUMBRA
A3	l_1, l_2, l_3	3 — UMBRA
A4	l_2, l_3	2 } PENUMBRA
A5	l_3	1 } PENUMBRA

Figure 3: Superposition of Shadow Volumes

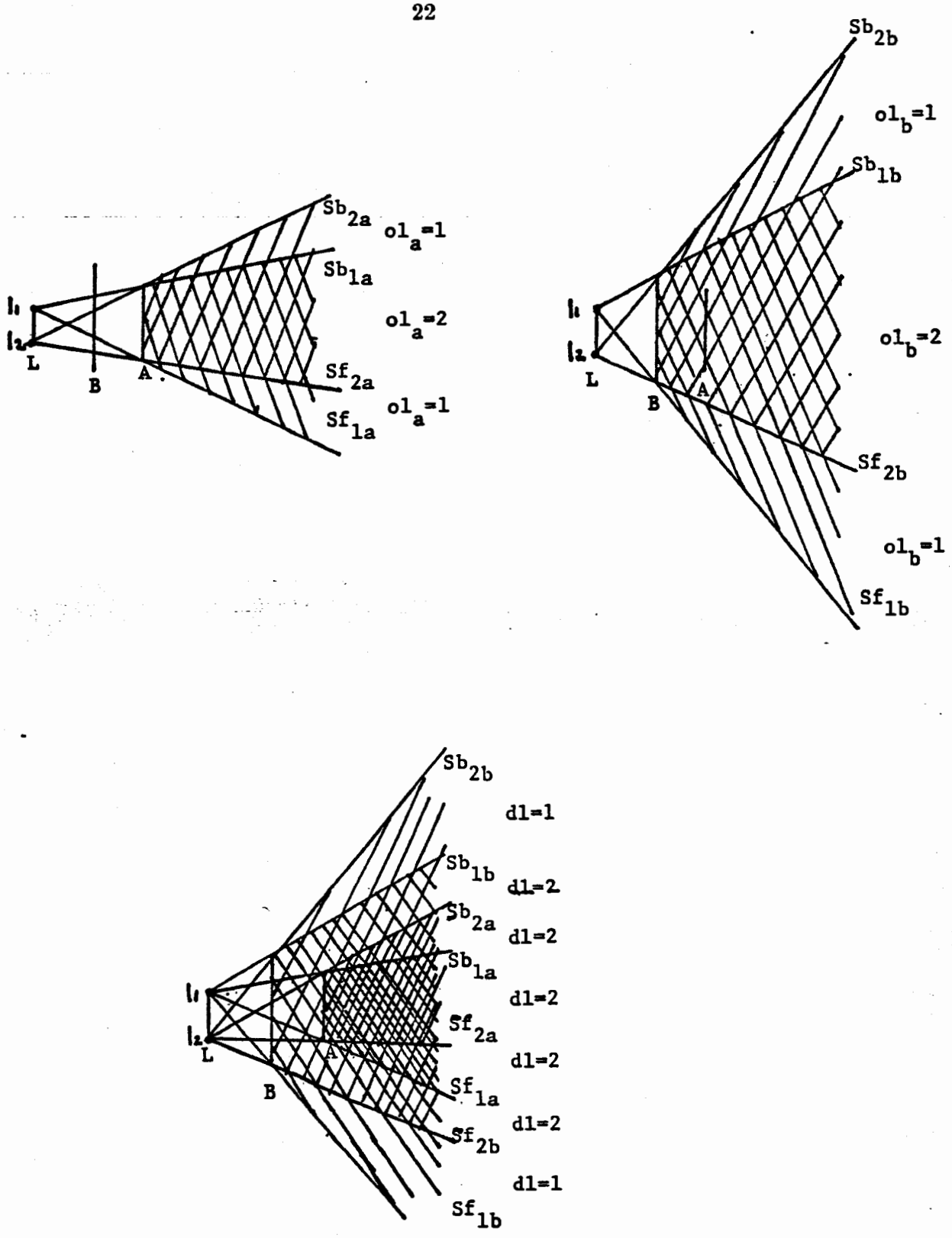


Figure 4: Two Objects Blocking the Same Light Source

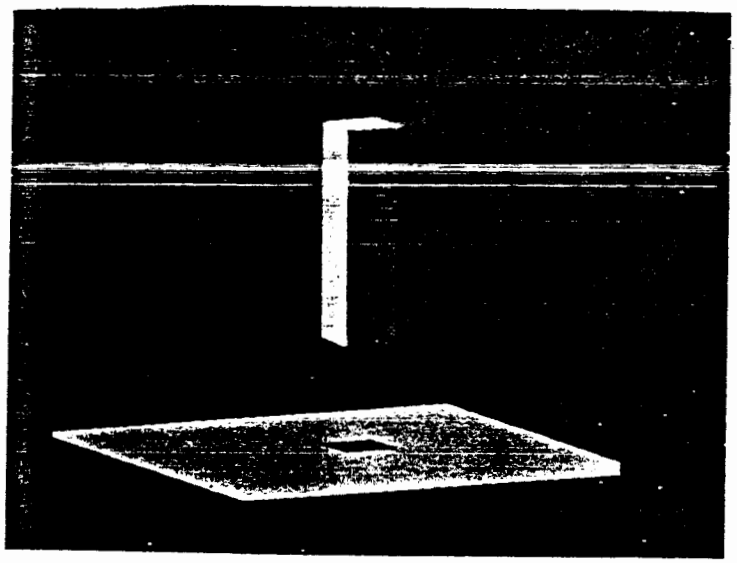


Figure 5: Shadow due to light source at infinity with rays in the -y direction. There is a second point source at infinity with rays in the +x direction.

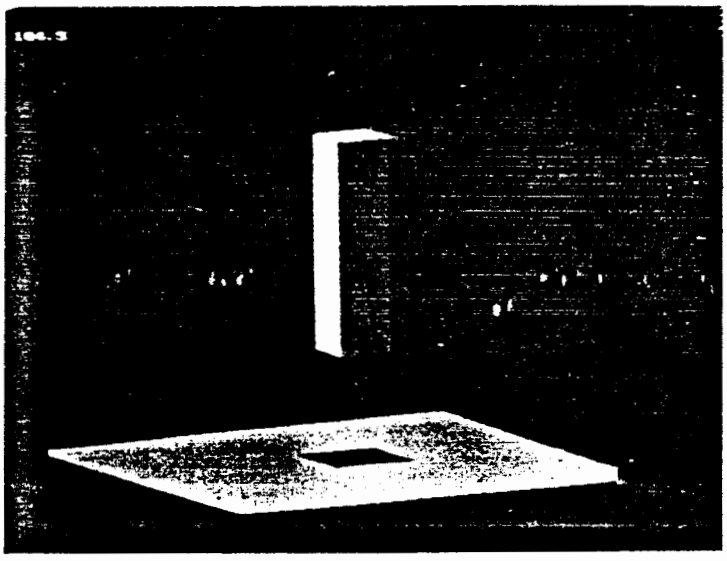


Figure 6: Shadow due to point source located above the center of the rectangular parallelpiped at a height four times the height of the parallelpiped.

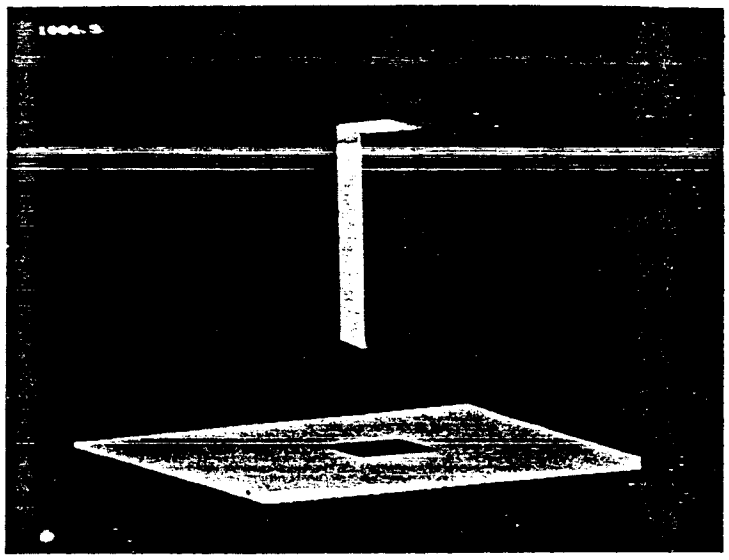


Figure 7: Shadow due to a square light source (about half the size of the top face of the parallelpiped) centered above the object at a height four times the height of the object. The square light source is modeled as 10 point sources.

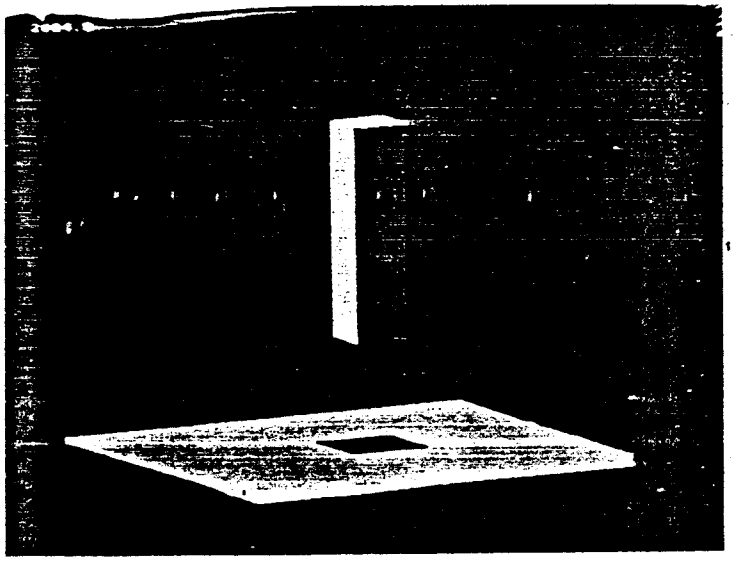


Figure 8: Square light source modeled as 20 point sources.

PRINT NOT AVAILABLE
AT THIS TIME

Figure 9: Zoomed in view of shadow in Figure 8.

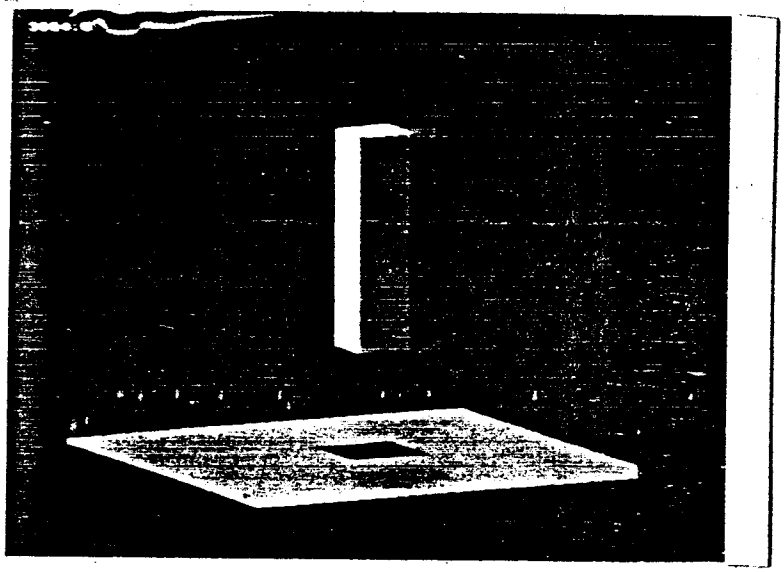


Figure 10: Square light source modeled as 30 point sources. Notice the smoothing and graying of the shadow edges as the size of the point source set is increased.

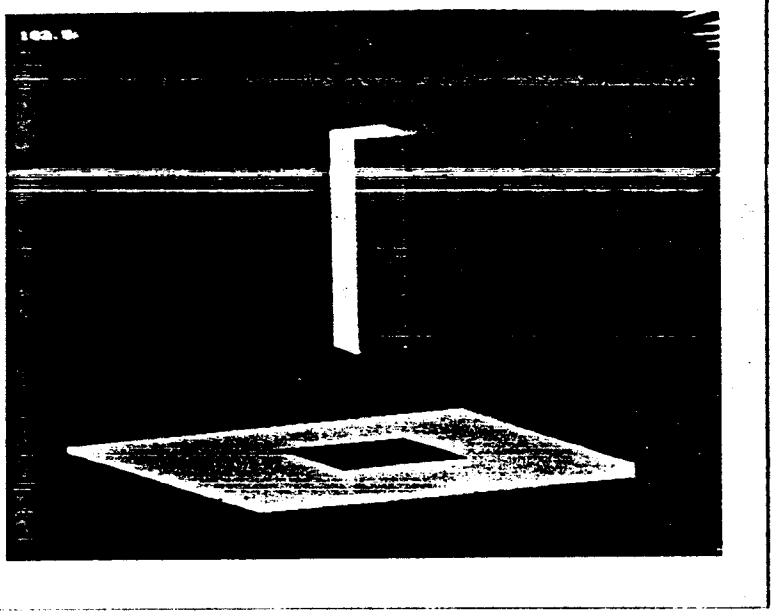


Figure 11: Shadow due to point source located above the center of the rectangular parallelpiped at a height two times the height of the parallelpiped.

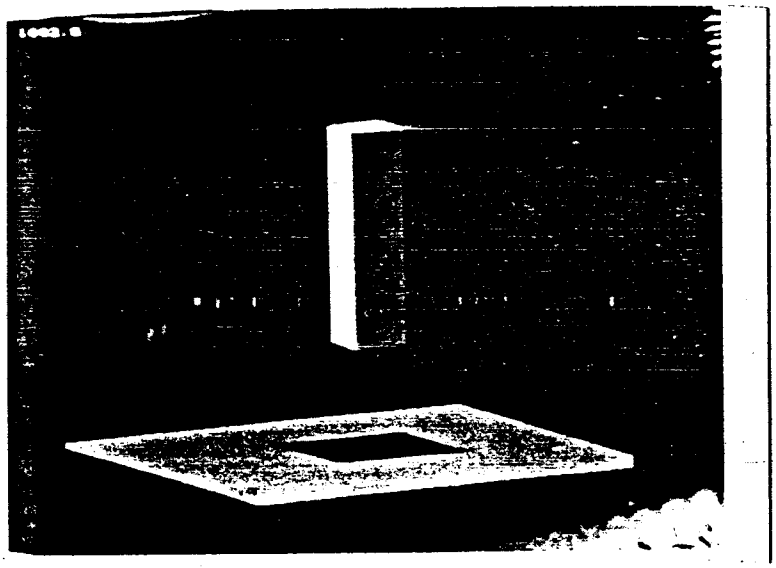


Figure 12: Shadow due to a square light source (about half the size of the top face of the parallelpiped) centered above the object at a height two times the height of the object. The square light source is modeled as 10 point sources.

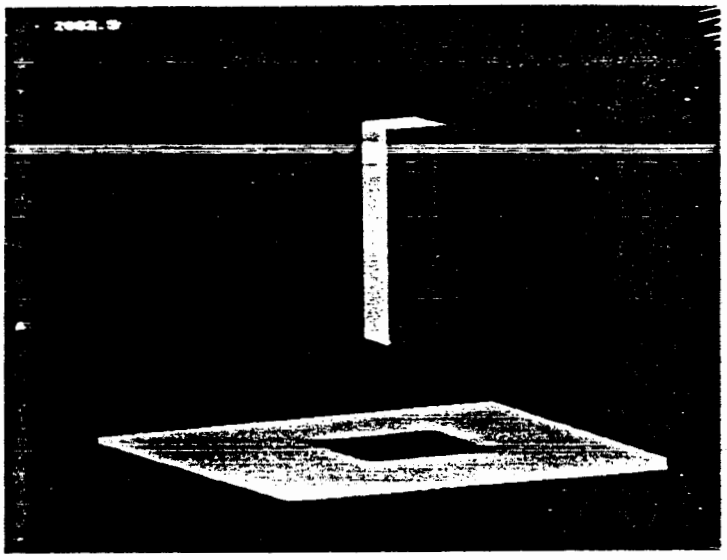


Figure 13: Square light source modeled as 20 point sources.

PRINT NOT AVAILABLE AT THIS TIME

Figure 14: Zoomed in view of shadow in Figure 13.

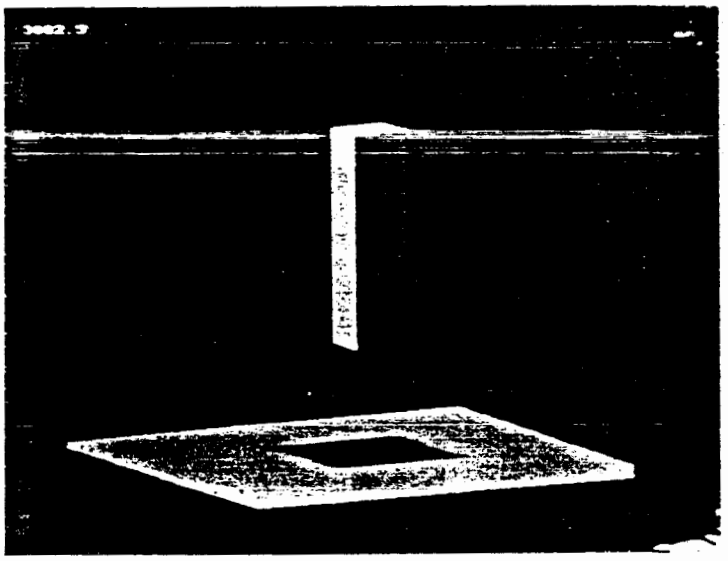


Figure 15: Square light source modeled as 30 point sources.

References

1. Appel, A., "Some Techniques for Shading Machine Renderings of Solids," *Proceedings of the Spring Joint Computer Conference, AFIPS,* Vol. 32, 1968, pp. 37-49.

2. Bouknight, J., "A Procedure for Generation of Three-dimensional Half-toned Computer Graphics Presentations," *CACM,* Vol. 13, No. 9, 1970, pp. 527-536.

3. Bouknight, J., and Kelley, K., "An Algorithm for Producing Half-tone Computer Graphics Presentations With Shadows and Movable Light Sources," *Proceedings of the Spring Joint Computer Conference, AFIPS,* Vol. 36, 1970, pp. 1-10.

4. Crow, F., "Shadow Algorithms for Computer Graphics," *Computer Graphics,* Vol. 11, No. 2, 1977, pp. 242-248.

5. Atherton, P., Weiler, K. and Greenberg, D., "Polygon Shadow Generation," *Computer Graphics,* Vol. 12,

No. 3, 1978, pp. 275-281.

6. Williams, L., "Casting Curved Shadows on Curved Surfaces," *Computer Graphics,* Vol. 12, No. 3, 1978, pp. 270-274.

7. Whitted, T., "An Improved Illumination Model for Shaded Display," *CACM,* Vol. 23, No. 6, 1980, pp. 343-349.

8. Appel, A., "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," *Proceedings of the ACM National Meeting,* Vol. 14, 1967, pp. 387-393.

9. Kelley, K., "A Computer Graphics Program for the Generation of Half-toned Images with Shadows," Master's thesis, University of Illinois, 1970.

10. Hall, R., and Greenberg, D., "A Testbed for Realistic Image Synthesis," *IEEE Computer Graphics and Applications,* Vol. 3, No. 8, 1983, pp. 10-20.

11. Stark, H., "Applications of Vander Lugt Filtering," Personal notes, not published,.

12. Minkoff, J., Hilal, S., Konig, W., Arm, M. and

Lambert, L., "Optical Filtering to Compensate for Degradation of Radiographics Images Produced by Extended Sources," *Applied Optics,* Vol. 7, No. 4, 1968, pp. 633-641.

13. Phong, B., "Illumination for Computer Generated Images," *CACM,* Vol. 18, No. 6, 1975, pp. 311-317.

14. Foley, J., and VanDam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, , Vol. 1, 1982.