11-1-1993

Visible Light and X-Ray Ray Tracing of Generalized Cylinders

Jean Hsu
Purdue University School of Electrical Engineering

David M. Chelberg
Purdue University School of Electrical Engineering

Visible Light and X-Ray Ray Tracing of Generalized Cylinders

Jean Hsu
David M. Chelberg

TR-EE 93-43
November 1993

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285

Visible Light and X-Ray Ray Tracing of Generalized Cylinders

Jean Hsu
David M. Chelberg

Geometric Modeling and Perceptual Processing Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, Indiana

This work was supported in part by a Purdue University Trask fund Grant, a Digital Equipment Corporation Faculty Incentives for Excellence Grant and an Indiana Elks Grant.

ABSTRACT

A new algorithm for ray tracing generalized cylinders whose axis is an arbitrary three dimensional space curve and whose cross-sectional contour can be varied according to a general sweeping rule is presented. The only restriction placed on the class of generalized cylinders that can be ray-traced is that the sweeping rule of the generalized cylinder must be invertible. This algorithm handles a broader class of generalized cylinders than any other reported ray tracer. It has been integrated into a general geometric modeling systenn that can render objects utilizing visible light as well as simulated X-rays.

Generalized cylinders are often used in modeling systems because they compactly represent objects. Many commonly occurring objects including snakes, horses, airplanes, flower vases, and organs of the human abdomen such as the stomach and liver can be described naturally and conveniently in terms of one or more generalized cylinder primitives. By extending the class of generalized cylinders that can be conveniently modeled, the presented algorithm enhances the utility of modeling systems based on generalized cylinders. X-ray images of the internal bone structure of a knee joint, and a visible light image of a fan blade assembly are presented.

Index Terms : computer-generated images; generalized cylinder; ray tracing; simulated X-ray images; radiograph;

I INTRODUCTION

Ray tracing is a well known technique that is widely used for generation of realistic images [1, 2, 3]. Using this technique, the visibility of surfaces is determined by tracing imaginary light rays from a viewer's eye to the objects in a scene. A similar technique, X-ray ray tracing, for generation of X-ray images is also described in this paper. A rnajor issue in ray tracing is object representation. Accurate representation of objects by geometric models is a pre-requisite for realistic image generation.

Generalized cylinders [4] are a class of primitive objects that can be used as general shape descriptors for many objects [5]. A generalized cylinder is a volumetric solid generated by sweeping an arbitrarily shaped closed cross section along an arbitrary three-dimensional space curve (axis) (See Figure 1). The Frenet frame [6, 7] offers a natural method of orienting the cross section relative to the axis. The cross section may be varied (deformed) as it is swept along the axis. This deformation function is usually called the sweeping rule. The expressiveness and compactness of generalized cylinders make them a good choice as primitives for object representation. This class of flexible parametric shapes is capable of modeling many different types of objects. Simple descriptions for many natural shapes exist. Complex shapes can be conveniently described by segmentation into a number of simpler generalized cylinders, each with no abrupt change in size or shape of the cross section, or in the axis direction. Objects represented by generalized cylinders are also easily modifiable. Examples of objects that have been represented by generalized cylinders in other research work include animals [8, 9], organs [10, 11] and various man-made objects [12, 13, 14].

Some research on ray tracing restricted classes of generalized cylinders has been reported. An algorithm for visible light ray tracing of generalized cylinders with constant cross-sectional contour is described in [15]. The work was later extended to include scaling of the contour curve in two orthogonal directions [16]. Visible light ray tracing of generalized cylinders with a varying cross-sectional curve along a straight axis has also been investigated [14, 17].

In this paper, we describe a method for ray tracing generalized cylinders whose axis is

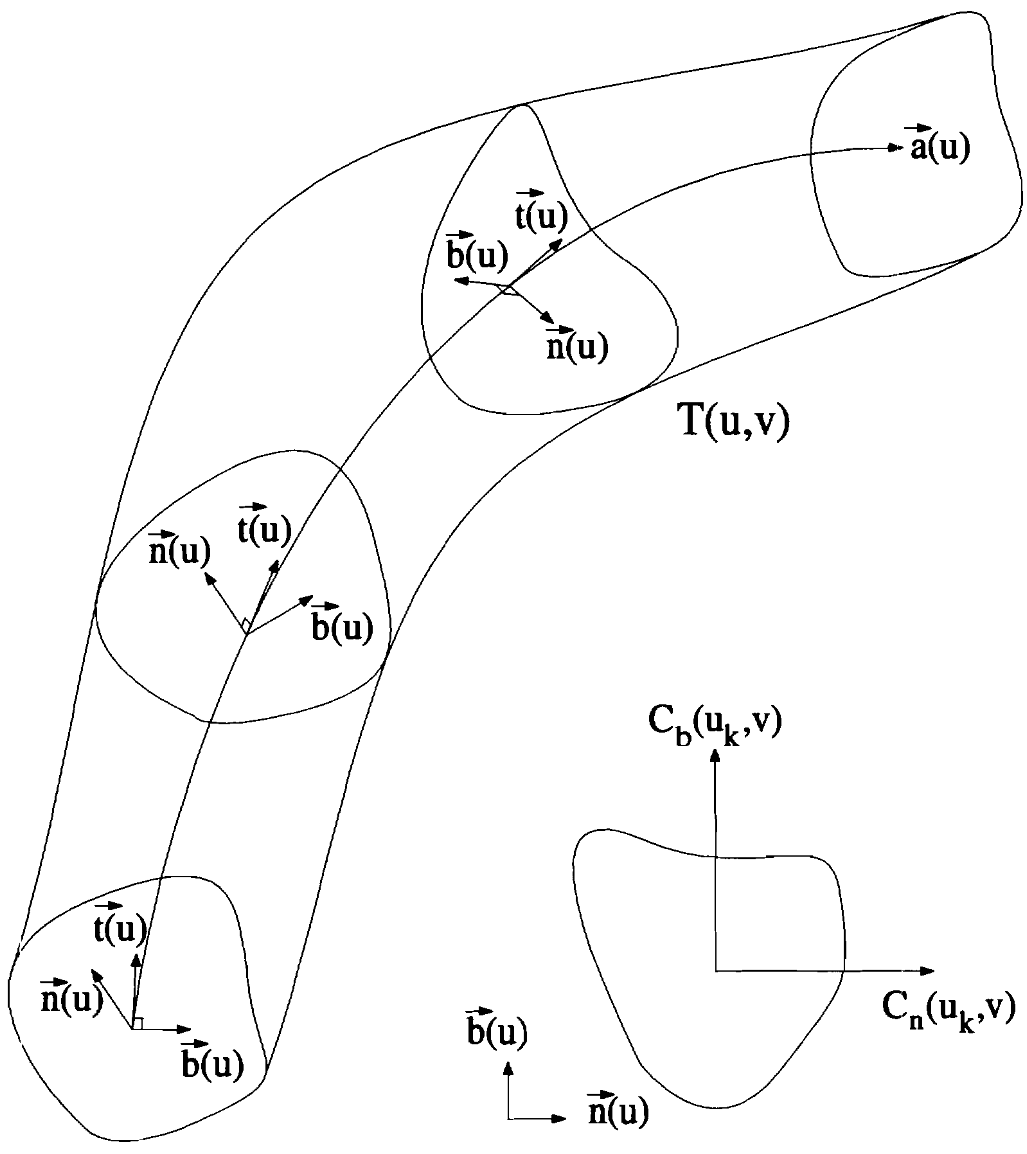


Figure 1: Notations used to describe a generalized cylinder.

an arbitrary 3D space curve and whose sweeping rule is invertible. A geometric modeling system has been implemented that can ray trace both visible light and X-ray images. To our knowledge, ray tracing of such a broad class of generalized cylinders has not been attempted before. Some example images generated by our geometric modeling system are shown in section IV. The example images give a glimpse of the wide class of objects whose realistic visible light and X-ray images can be generated.

II RAY TRACING GENERALIZED CYLINDERS

The ray tracing problem can be stated as follows:

> Given a scene description, an eye position and an image plane, find the 2D image that is observed when the scene is viewed from the eye position.

An equivalent statement for the X-ray ray tracer reads:

> Given a scene description, an X-ray source and an image plane, find the 2D X-ray image that is formed on the image plane.

The imagt: plane may be thought of as a rectangular grid of the desired resolution. A ray is cast from the eye (source) to a grid location on the image plane (see Figure 2). Intersection points between the ray and objects in the scene are found and used to compute the value for the grid location. The image is fully rendered when the value of each grid location on the image plane has been determined. It is therefore evident that at the "heart" of every ray tracer.,there are routines for computation of intersection points between a ray and each allowable :primitive object in the scene. In the following subsections, we describe the details of how intersection points between a ray and a generalized cylinder can be computed. We also describe how the resultant image can then be generated from the computed intersection points.

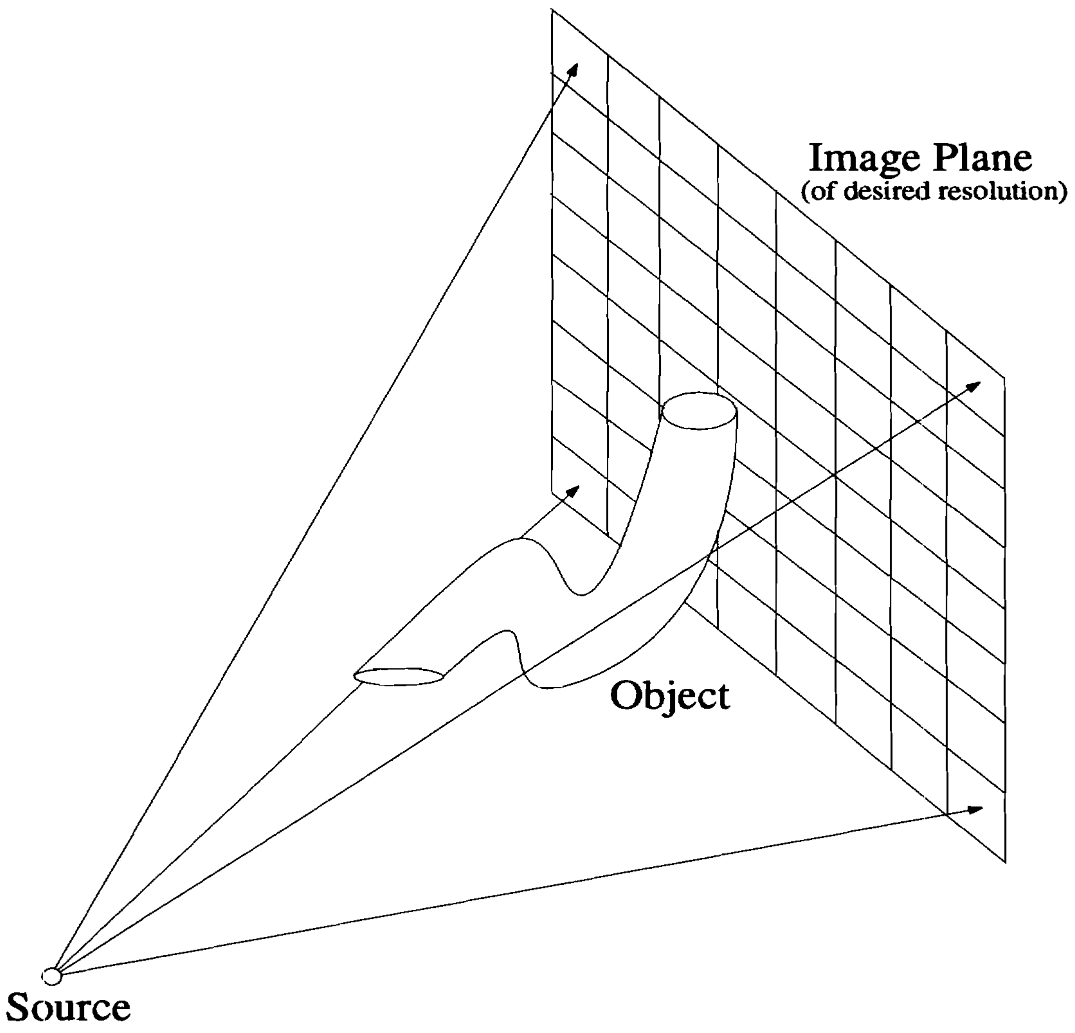


Figure 2: Basic principles behind a ray tracer.

A Formal Definition of a Ray and a Generalized-Cylinder

The problem we want to solve is to find the intersection points between a ray, $\mathbf{r}(w) = \mathbf{p} + w\mathbf{d}$, and a generalized cylinder. A generalized cylinder can be described by a 3D axis curve, $\mathbf{a}(u) = (a_x(u), a_y(u), a_z(u))$ where $u_i \leq u \leq u_f$ and a closed 2D contour curve, $\mathbf{c}(u, v) = (c_n(u, v), c_b(u, v)) = \mathbf{s}(\mathbf{cs}(v), u)$ where $v_i \leq v \leq v_f$ and s is the sweeping rule that specifies how a particular cross-sectional contour, $\mathbf{cs}(v)$, changes along the axis of the generalized cylinder. The axis curve is assumed to be regularly parametrized, i.e. $\mathbf{a}'(u) \neq 0$, $u \in [u_i, u_f]$. The shape of the contour at each point along the axis is given by $\mathbf{c}(u, v)$. For the orientation of the contour, the Frenet frame [6, 7] is used. The Frenet frame depends only on the local shape of the trajectory. It is independent of the axis parametrization and of the coordinate system in which it is defined. These qualities make the Frenet frame an ideal choice as a local coordinate system for each point along the axis. The Frenet frame consists of three orthogonal unit vectors:

tangent vector: $\hat{\mathbf{t}}(u) = \dfrac{\mathbf{a}'(u)}{|\mathbf{a}'(u)|}$

binormal vector: $\hat{\mathbf{b}}(u) = \dfrac{\hat{\mathbf{t}}(u) \times \mathbf{a}''(u)}{|\hat{\mathbf{t}}(u) \times \mathbf{a}''(u)|} = \dfrac{\mathbf{a}'(u) \times \mathbf{a}''(u)}{|\mathbf{a}'(u) \times \mathbf{a}''(u)|}$

normal vector: $\hat{\mathbf{n}}(u) = \hat{\mathbf{b}}(u) \times \hat{\mathbf{t}}(u) = \dfrac{|\mathbf{a}'(u)|^2\mathbf{a}''(u) - (\mathbf{a}'(u) \cdot \mathbf{a}''(u))\mathbf{a}(u)}{|\mathbf{a}'(u)||\mathbf{a}'(u) \times \mathbf{a}''(u)|}$.

$\hat{\mathbf{t}}(u)$ is normal to the plane of the contour. $\hat{\mathbf{n}}(u)$ and $\hat{\mathbf{b}}(u)$ define the directions of the first and second coordinate axis in the contour plane. The t b plane is called the rectifying plane, the n b plane is called the normal plane and the n t plane is called the osculating plane.

Note however that the Frenet frame is not defined at points where $\mathbf{a}'(u)$ is linearly dependent on $\mathbf{a}''(u)$, i.e. where the curvature, $\kappa(u)$, is zero. This is not a major problem since for a space curve, $\kappa(u)$ is typically not equal to zero. The following fact has been proven in [18]:

If $\kappa(u) \equiv 0$ on the interval of definition, then $\mathbf{a}(u)$ is a straight line segment on that interval, and conversely, for a straight line, $\kappa(u) \equiv 0$.

Hence $\kappa(u) = 0$ occurs only at points of inflection along the axis of the generalized cylinder or when the axis of the generalized cylinder is linear. The implementation issues for these special cases are described in the next section.

Using the definitions of the axis curve, the contour curve and the Frenet frame, the equation for the surface of the generalized cylinder (see Figure 1) is given by:

$$\mathbf{\Gamma}(u,v) = \mathbf{a}(u) + c_n(u,v)\hat{\mathbf{n}}(u) + c_b(u,v)\hat{\mathbf{b}}(u). \tag{1}$$

B Overview of Ray-Generalized-Cylinder Intersection Algorithm

The basic idea behind the Ray-Generalized-Cylinder (Ray-GC) intersection algorithm is to reduce the 3D intersection problem into a 2D intersection problem. This idea has been applied to solve for the intersection between a ray and restricted classes of generalized cylinders, namely generalized cylinders with constant cross-sectional contour [15] and straight axis generalized cylinders [17]. A direct extension of the idea to profiled generalized cylinders (i.e. generalized cylinders with cross-sectional contour which can be scaled in two orthogonal directions) has been reported [16]. In this paper, we describe the missing link that allows elegant computation of the intersection between a ray and a generalized cylinder with full generality except the restriction that the sweeping rule of the generalized cylinder must be invertible. This extension significantly widens the class of generalized cylinders that can be rendered.

The reduction in dimensionality of the Ray-GC intersection problem can be achieved by first considering the intersection of the ray with each normal plane along the axis of the generalized cylinder. The computed intersection points all lie on the 2D plane spanned by $\hat{\mathbf{n}}(u)$ and $\hat{\mathbf{b}}(u)$ (normal plane). These intersection points collectively form a 2D curve, $\boldsymbol{\rho}(u)$. By definition, the contour curve of the generalized cylinder lies on the normal plane too. Therefore, for a fixed normal plane (e.g. plane spanned by $\hat{\mathbf{n}}(u_j)$ and $\hat{\mathbf{b}}(u_j)$), if the ray intersects the surface of the generalized cylinder, the point $\boldsymbol{\rho}(u_j)$ will lie on the contour curve $\mathbf{c}(u_j, v)$. No intersection of the ray with the generalized cylinder occurs (for the particular normal plane) if the point $\boldsymbol{\rho}(u_j)$ does not lie on the contour curve $\mathbf{c}(u_j, v)$. Hence, the intersection of a ray with a generalized cylinder can be computed by finding the intersection points of the curve $\boldsymbol{\rho}(u)$ with the contour curve $\mathbf{c}(u, v)$. However, since the contour curve $\mathbf{c}(u, v)$ varies as the parameter u is changed, the problem is still inherently **3D** and the intersection points cannot be easily determined (see Figure 3).

In this paper, we describe an elegant solution for finding the intersection points between the curve $\boldsymbol{\rho}(u)$ and the contour curve $\mathbf{c}(u, v)$ when the deformation function (sweeping rule) of the contour curve is invertible. In this case, instead of finding the intersection points directly, the inverse sweeping rule is first applied to the curve $\boldsymbol{\rho}(u)$ to give another **2D** curve $\boldsymbol{\phi}(u)$. The intersection of the curve $\boldsymbol{\phi}(u)$ with the specified cross-sectional contour curve $\mathbf{cs}(v)$ is equivalent, to the intersection of the curve $\boldsymbol{\rho}(u)$ with the cross-sectional contour curve $\mathbf{c}(u, v)$. Since both $\boldsymbol{\phi}(u)$ and $\mathbf{cs}(v)$ lie on the normal plane, the **3D** problem has been reduced to a **2D** problem and the intersection points can be computed relatively easily (see Figure 4). In the next three subsections, we describe in detail how the intersection points are computed.

C Intersection of a ray with each normal plane along the axis

The equation of a ray, $\mathbf{r}(w)$, in the local coordinate system determined by the Frenet frame, $\hat{\mathbf{t}}(u)$, $\hat{\mathbf{n}}(u)$ and $\hat{\mathbf{b}}(u)$ is:

$$\mathbf{r}_l(w, u) = \begin{pmatrix} \hat{\mathbf{t}}(u) \\ \hat{\mathbf{n}}(u) \\ \hat{\mathbf{b}}(u) \end{pmatrix} \cdot (\mathbf{p} + w\mathbf{d} - \mathbf{a}(u)).$$

The point of intersection between the ray, $\mathbf{r}(w)$, and the normal plane, $\boldsymbol{\Pi}(u)$ (spanned by $\hat{\mathbf{n}}(u)$ and $\hat{\mathbf{b}}(u)$ and passing through $\mathbf{a}(u)$), can be found using the following equation:

$$\hat{\mathbf{t}}(u) \cdot (\mathbf{p} + w\mathbf{d} - \mathbf{a}(u)) = 0.$$

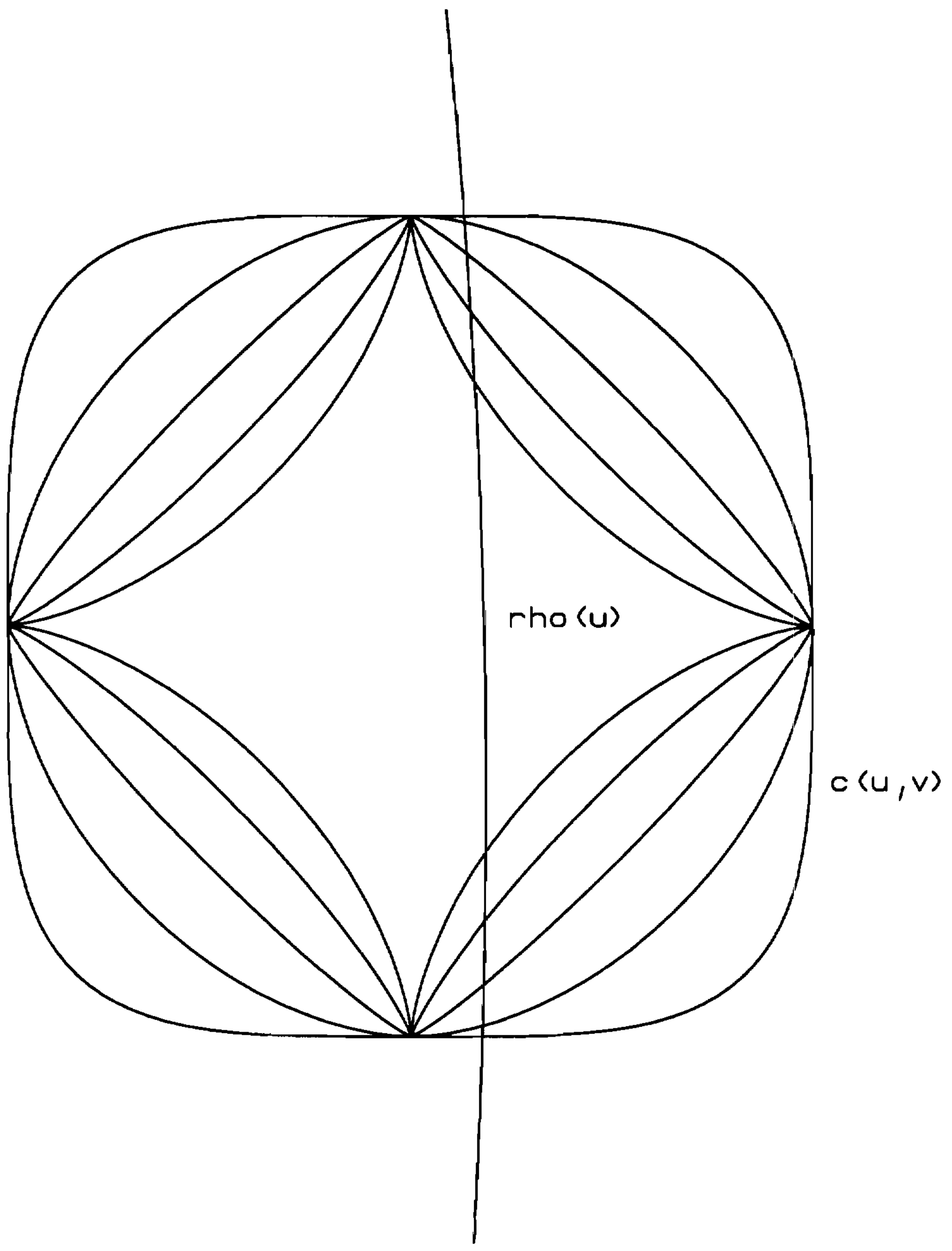


Figure **3:** Intersection of the curve $\boldsymbol{\rho}(u)$ with the contour curves $\mathbf{c}(u,v)$ for deformed object shown in Figure 8. The contour curves have u values varying from 0.5 to **3.0** (in steps of 0.625) from the outermost curve to the innermost one.

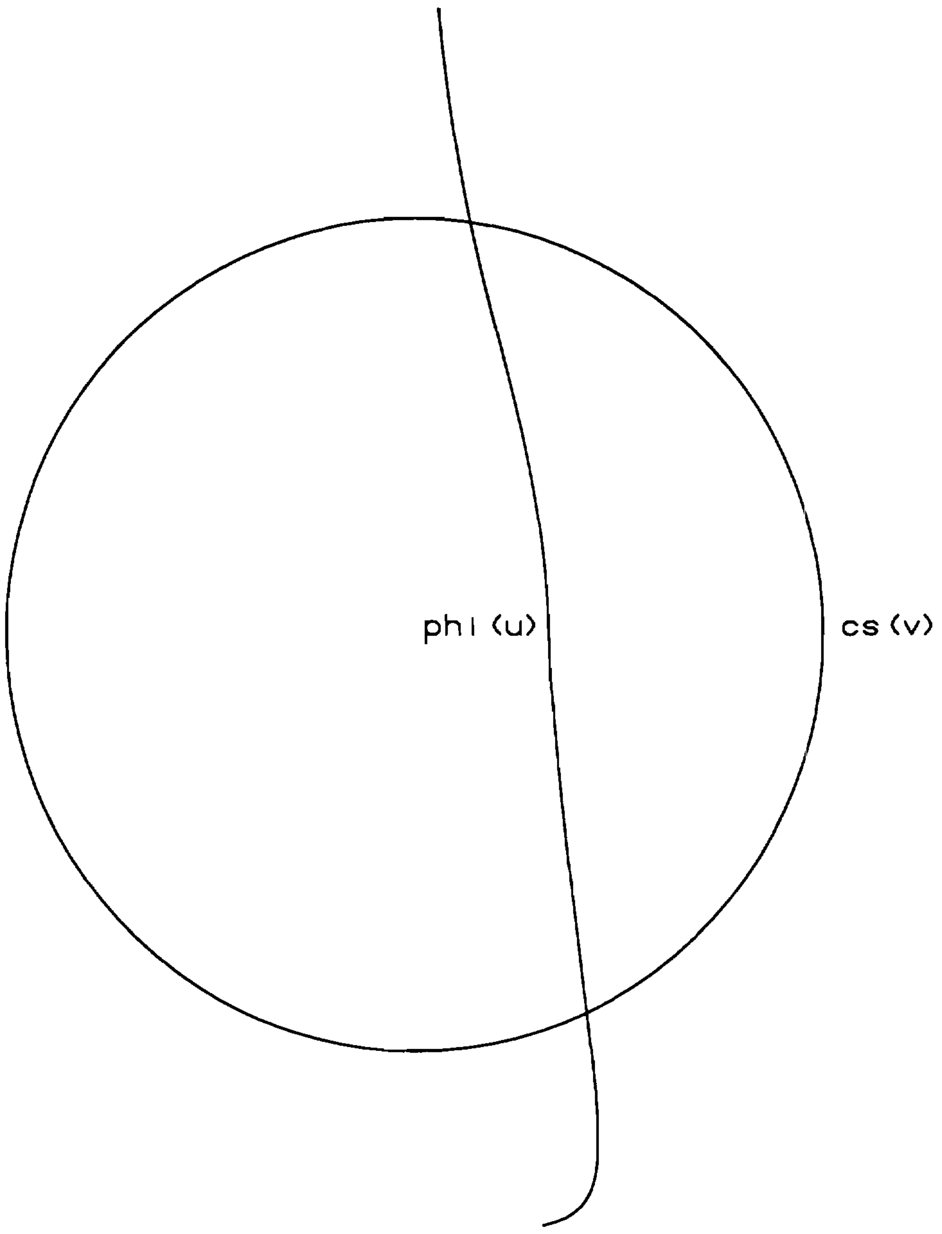


Figure 4: Intersection of the curve $\phi(u)$ with the specified contour curve $cs(v)$ for deformed object shown in Figure 8.

Substituting $\hat{\mathbf{t}}(u) = \frac{\mathbf{a}'(u)}{|\mathbf{a}'(u)|}$ and simplifying, we get:

$$\mathbf{a}'(u) \cdot (\mathbf{p} + w\mathbf{d} - \mathbf{a}(u)) = 0.$$

Therefore,

$$w = -\frac{\mathbf{a}'(u) \cdot (\mathbf{p} - \mathbf{a}(u))}{(\mathbf{a}'(u) \cdot \mathbf{d})}$$

provided $\mathbf{a}'(u) \cdot \mathrm{d} \neq 0$.

The case of $\mathbf{a}'(u) \cdot \mathrm{d} = 0$ corresponds to the case where the ray is parallel to the contour plane. This is treated as a special case and the implementation details can be found in the next section.

For normal cases, the value of w is substituted into $\mathbf{r}_l(w, u)$ to give $\boldsymbol{\rho}(u)$ which is the projection of the ray onto the normal planes along the axis.

$$\boldsymbol{\rho}(u) = \begin{pmatrix} \mathbf{n}(\mathbf{u}) \cdot (\mathbf{p} - \frac{\mathbf{a}'(u) \cdot (\mathbf{p} - \mathbf{a}(u))}{(\mathbf{a}'(u) \cdot \mathbf{d})}\mathbf{d} - \mathbf{a}(u)) \\ \mathbf{b}(\mathbf{u}) \cdot (\mathbf{p} - \frac{\mathbf{a}'(u) \cdot (\mathbf{p} - \mathbf{a}(u))}{(\mathbf{a}'(u) \cdot \mathbf{d})}\mathbf{d} - \mathbf{a}(u)) \end{pmatrix}$$

D Apply inverse sweeping rule to $\boldsymbol{\rho}(u)$

The intersection of $\boldsymbol{\rho}(u)$ with $\mathbf{c}(u, v)$ corresponds to the intersection of the ray with the generalized cylinder. However, as discussed in section II(B), the intersection points can be found much more readily by first transforming $\boldsymbol{\rho}(u)$ by the inverse sweeping rule and then intersecting the resulting curve $\boldsymbol{\phi}(u)$ with the specified cross-sectional curve $\mathbf{cs}(v)$, i.e. deformation of a ray rather than the object [19]. The inverse relationship between the specified cross-section and $\mathbf{c}(u, \mathrm{v})$ is given by $\mathbf{cs}(v) = \mathbf{s}_{inv}(\mathbf{c}(u, \mathrm{v}), u)$. Applying the inverse sweeping rule to $\boldsymbol{\rho}(u)$, we get $\boldsymbol{\phi}(u) = \mathbf{s}_{inv}(\boldsymbol{\rho}(u))$. The intersection of $\boldsymbol{\phi}(u)$ with $\mathbf{cs}(v)$ corresponds to the intersection of the ray with the generalized cylinder.

E Subdivision algorithm for solving intersection points between $\phi(u)$ and $cs(v)$

Solving for the intersection points between the curves $\phi(u)$ and $\mathbf{cs}(v)$ is not a straight forward problem since the curves, in particular $\phi(u)$, may be complex. A similar problem has been discussed in [15, 20]. We follow the basic approach of [20] in our work.. This approach avoids the problem in [15] of finding multiple intersection points when only one exists [21]. Points with horizontal or vertical tangents and points of inflection along the curve have to be computed. Due to the complexity of the curves, it is generally difficult to solve for such points anidytically. Numerical techniques of sampling are used instead.

An outline of the subdivision algorithm follows:

First, the curves $\phi(u)$ and $\mathbf{cs}(v)$ are split at inflection points and points with horizontal or vertical tangents (see Figure 5). This results in curve segments whose maximum and minimum coordinate values occur at the end-points. Hence, the entire curve segment is bounded within the box defined by the end-points of the curve segment. If such a curve segment is further subdivided, the subdivided curve segments are also bounded by their respective end-points boxes. A proof of this property can be found in [20].

After splitting the curves, the curve-intersect routine is called on all possible combinations of a curve segment from $\phi(u)$ and a curve segment from $\mathbf{cs}(v)$. The following is a pseudo-code algorithm of the curve-intersect routine:

```
procedure curve-intersect($curve_1$,$curve_2$:curveseg)

if overlap($curve_1$,$curve_2$)
    if linear($curve_1$)
        if linear($curve_2$)
            line-intersection($curve_1$,$curve_2$);
        else
            curve-divide($curve_2$,$curve_{21}$,$curve_{22}$);
            curve-intersect($curve_1$,$curve_{21}$);
            curve-intersect($curve_1$,$curve_{22}$);
    else
        if linear($curve_2$)
            curve-divide($curve_1$,$curve_{11}$,$curve_{12}$);
```

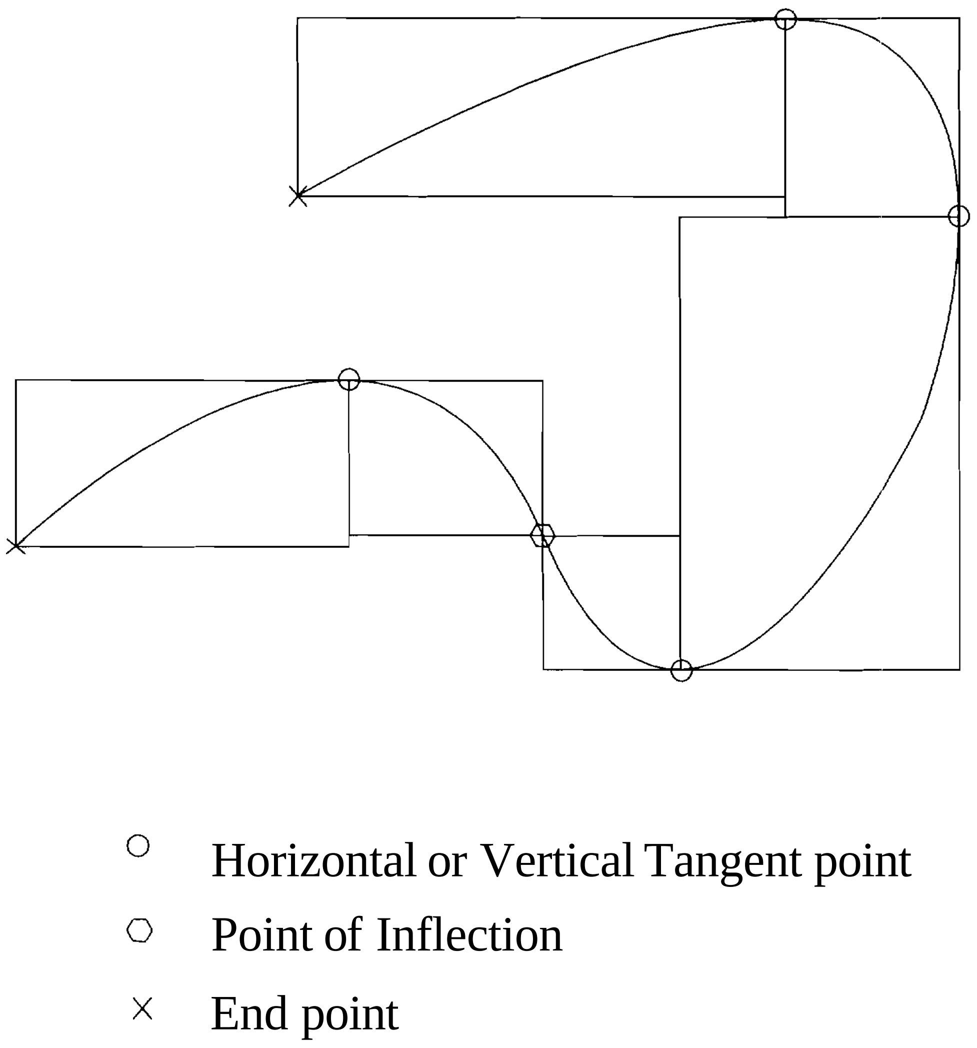


Figure 5: Subdivision of a curve.

```
        curve-intersect($curve_{11}$,$curve_2$);
        curve-intersect($curve_{12}$,$curve_2$);
    else
        curve-divide($curve_1$,$curve_{11}$,$curve_{12}$);
        curve-divide($curve_2$,$curve_{21}$,$curve_{22}$);
        curve-intersect($curve_{11}$,$curve_2$);
        curve-intersect($curve_{12}$,$curve_2$);
        curve-intersect($curve_1$,$curve_{21}$);
        curve-intersect($curve_1$,$curve_{22}$);
```

Note:

overlap($curve_1$,$curve_2$) detects whether the two curve segments have spatially overlapping bounding boxes.

linear($curve$) is a function that tests the linearity of a curve segment. Since each curve segment has the property that the maximum and minimum coordinate values occur at the end-points and there is no inflection point along any curve segment, a good test for the linearity of a segment is to find the perpendicular distance between the point of intersection of the tangent lines at the end-points and the chord joining the end-points of the segment (see Figure 6). Since the curve segment must lie between the triangle define by the two tangent lines and the chord, if the perpendicular distance found is less than ϵ, the curve segment is said to satisfy the linearity test.

If the x and y coordinates are separately considered and the deviation of each coordinate is less than ϵ, then the maximum deviation of the curve segment from the chord is less than $\sqrt{2}\epsilon$. Therefore, without loss of generality, let us consider the x coordinate. The following paragraph derives the formula for calculating the deviation:

Let $\mathbf{A} \equiv (u_0, x(u_0))$ and $\mathbf{B} \equiv (u_1, x(u_1))$ be the end-points of the curve segment and $\mathbf{X} \equiv$ (u*,x*) be the point where the two tangent lines at u_0 and u_1 intersect.

The equation of the lines **XA** and **XB** are:

$$x = x_0 + x'_0(u - u_0)$$

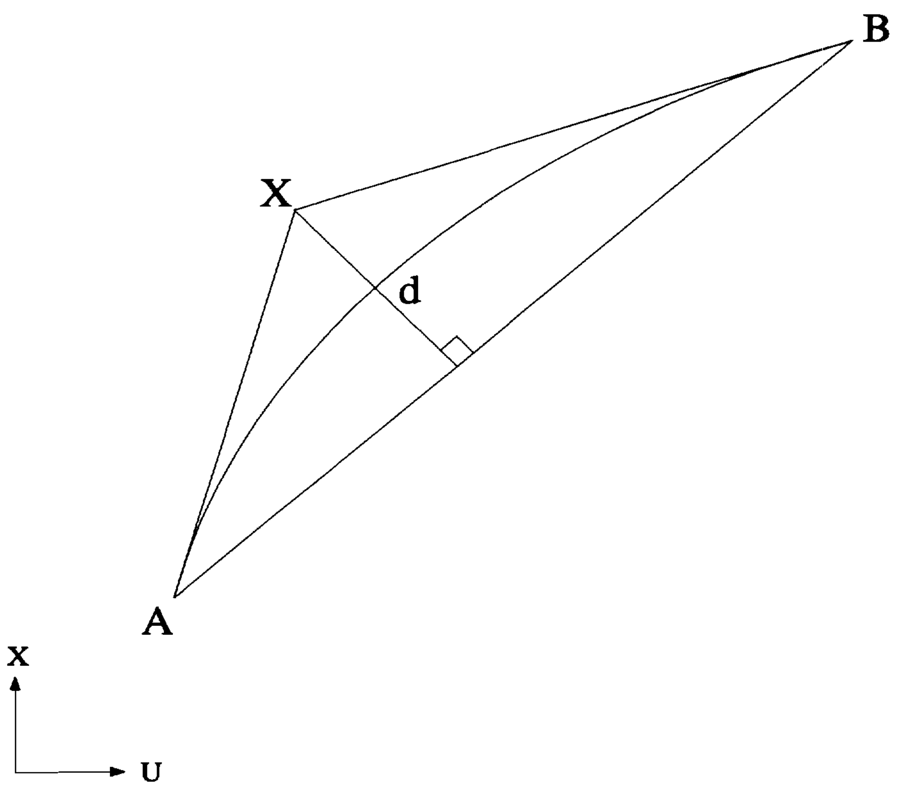


Figure 6: Testing for linearity of a curve segment.

$$x = x_1 + x_1'(u - u_1).$$

Solving, we get

$$u^* = \frac{(x_0 - x_1 + x_1'u_1 - x_0'u_0)}{(x_1' - x_0')}$$
$$x^* = x_0 + x_0'(u^* - u_0).$$

The equation of the chord is:

$$x(u_1 - u_0) - (x_1 - x_0)u + [(x_1 - x_0)u_0 - x_0(u_1 - u_0)] = 0.$$

Therefore, the perpendicular distance of (u*,x*) from the chord [22] is given by:

$$d = \left| \frac{(u_1 - u_0)x^* + (x_0 - x_1)u^* + [(x_1 - x_0)u_0 - x_0(u_1 - u_0)]}{\sqrt{(u_1 - u_0)^2 + (x_0 - x_1)^2}} \right|.$$

line-intersection($curve_1$,$curve_2$) detects whether the two almost linear curve segments intersect. If so, it returns the parameter values of the intersection point. Otherwise, it returns nil.

curve-divide($curve$,$curve_1$,$curve_2$) subdivides curve at the midpoint of the range of parameter values of curve into $curve_1$ and $curve_2$.

The curve-intersect routine returns the parameter values for the intersection points (if any). Using equation 1, the actual **3D** coordinates of the intersection points can be computed.

F End planes

So far, we have discussed the computation of the intersection points of a ray with the surface of a generalized cylinder. If the ray also intersects one of the end planes of the generalized cylinder, $\phi(u_i)$ or $\phi(u_f)$ will lie within the specified closed cross-sectional contour curve, **cs**(v). If so, the point of intersection can be computed by finding the intersection of the ray with the appropriate end plane. This ray-plane intersection can be easily computed using standard techniques [23].

G Normal calculation (for visible light shading)

To generate visible light images, it is necessary to know how to shade surfaces based on the position, orientation and characteristics of the surfaces and the light sources illuminating them. Most illumination models require the computation of the surface normal at the point under consideration. In our geometric modeling system, Phong's illumination model [23] (see Figure 7) is used. It should be noted that other lighting models can be easily incorporated into the system but they are not discussed here since illumination is not the main focus of this paper. Phong's illumination model is as follows:

$$\begin{aligned} I &= K_a + K_d cos\theta + K_s cos^n \phi \\ &= K_a + K_d(\hat{\mathbf{N}} \cdot \hat{\mathbf{L}}) + K_s(\hat{\mathbf{H}} \cdot \hat{\mathbf{N}})^n \end{aligned}$$

where

K_a = *ambient-reflection component of object*

K_d = *diffuse-reflection component of object*

K_s = *specular-reflection component of object*

n = *specular-reflection exponent of object*

$\hat{\mathbf{N}}$ = *surface normal at intersection point*

$\hat{\mathbf{L}}$ = *direction to the light source*

$\hat{\mathbf{E}}$ = *direction to the eye*

$H = \|\hat{\mathbf{E}} \oplus \hat{\mathbf{L}}\|$.

$\oplus$ denotes vector addition. $K_a, K_d, K_s, n, \hat{\mathbf{L}}$ and E are specified as input to the geometric modeling system and $K_a + K_d + K_s \leq 1$. H and N have to be computed. The computation of H is straight forward. The surface normal, N, of a point on the surface of a generalized cylinder can be computed as follows:

$$\hat{\mathbf{N}} = \eta(u,v) = \frac{\partial \mathbf{\Gamma}}{\partial v}(u,v) \times \frac{\partial \mathbf{\Gamma}}{\partial u}(u,v)$$

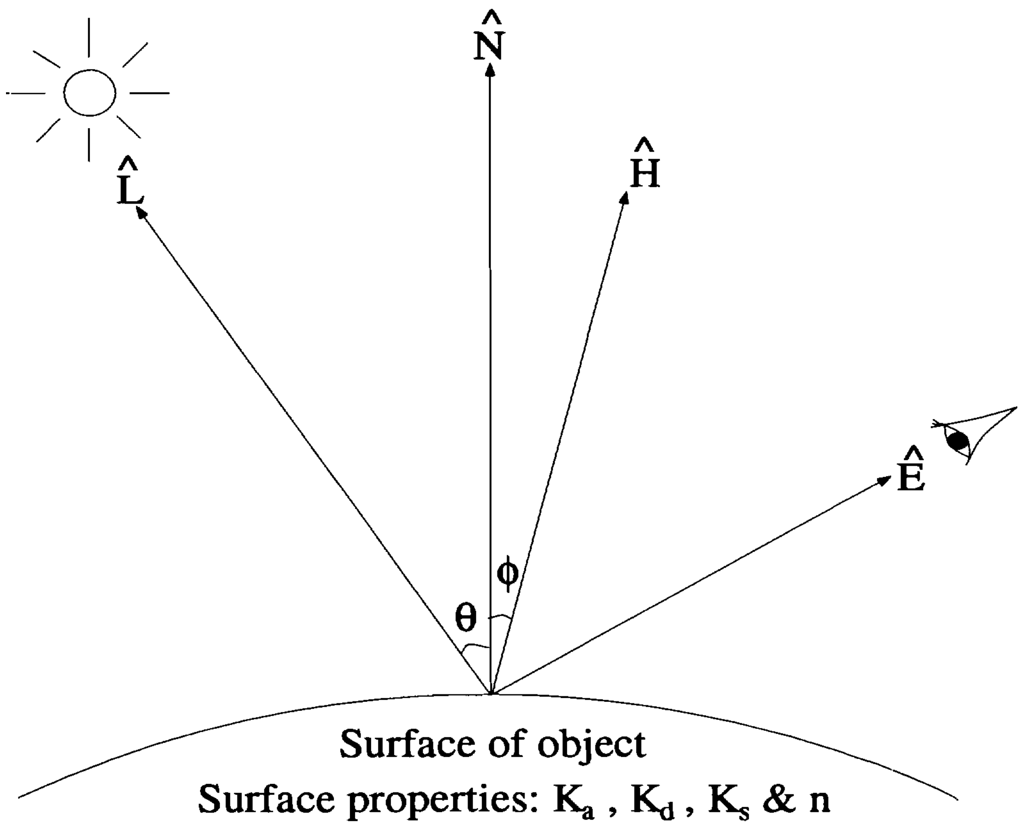


Figure 7: Phong's Illumination model.

where

$$\frac{\partial \mathbf{\Gamma}}{\partial u}(u,v) = \mathbf{a}'(u) + \hat{\mathbf{n}}(u)\frac{\partial c_n}{\partial u}(u,v) + c_n(u,v)\hat{\mathbf{n}}'(u) + \hat{\mathbf{b}}(u)\frac{\partial c_b}{\partial u}(u,v) + c_b(u,v)\hat{\mathbf{b}}'(u)$$

and

$$\frac{\partial \mathbf{\Gamma}}{\partial v}(u,v) = \frac{\partial c_n}{\partial v}(u,v)\hat{\mathbf{n}}(u) + \frac{\partial c_b}{\partial v}(u,v)\hat{\mathbf{b}}(u).$$

Using the parameter values (u, v) of the first (closest to eye) intersection point of the ray with the object, the value of N can be computed. If the intersection point lies on one of the end planes, N is instead given by $-\hat{\mathbf{t}}(u_i)$ or $\hat{\mathbf{t}}(u_f)$. For the visible light ray tracer, the computed value of **I** is the value recorded in the grid location (hit by ray) on the image plane.

H Distance calculation (for X-ray intensity)

To generate X-ray images, it is necessary to know the intensity of the X-ray that reaches the image plane. Beer's law for absorption of photons by radiodense materials is used to compute the amount of attenuation of the X-ray by the objects in the scene [24]:

$$N = N_0 e^{-\mu x}$$

where

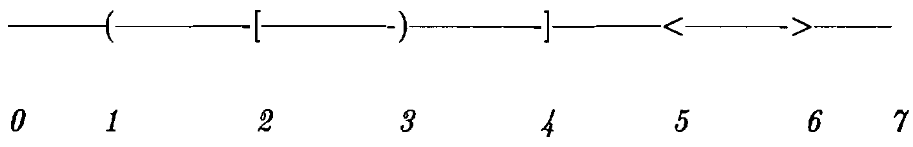
N = ***number of transmitted photons***

N_0 = ***number of incident photons***

μ = ***linear attentuation coefficient (object density)***

x = ***object thickness.***

Object densities are specified as input to the geometric modeling system. Object thickness is given by the distance that a ray passes through an object. It is computed by finding the intersection points of the ray with all objects along its path and then obtaining the distances between the intersection points in sequence. The distance between successive points can be easily (computed by the Euclidean distance. The number of photons reaching the image plane is recorded as the value for the pixel location. The following is a simple example that illustrates the computation:



The line denotes a ray passing through objects of various densities. Let $density_{ij}$ denote the density associated with the segment between point i and point j and $distance_{ij}$ denote the distance between point i and point j. For regions where objects overlap, such as between points 2 and 3, the density used can be the sum of the densities of the overlapping objects or the density of either one of the overlapping objects. More discussions on how objects are combined can be found in section III(D).

Assuming that $N_0 = 1$, we have

$$N_1 = N_0 e^{-density_{01} \times distance_{01}}$$

$$N_2 = N_1 e^{-density_{12} \times distance_{12}}$$

$$\cdot$$

$$\cdot$$

$$N_7 = N_6 e^{-density_{67} \times distance_{67}}$$

If point 7 is a point on the image plane, then the X-ray ray tracer records N_7 as the value of the grid location on the image plane.

I Contrast Stretching of Images

The visible light and X-ray images that are obtained (using the algorithm described above) contain values between 0 and 1. An additional step of contrast stretching [25] is necessary for visualization of these images. The images are linearly stretched so that the minimum and maximum values are mapped to the full color space. For 8-bit greyscale images, the minimum value is mapped to 0 and the maximum value is mapped to 255. For color images, the imager is independently scaled by the appropriate red, green and blue amounts to give the red, green and blue images. The minimum and maximum values of all 3 images are then used to compute the contrast stretch mapping. The same mapping is applied to the red, green and blue images to ensure color correctness. Since the mapping is single-valued and monatonically increasing, the order of gray levels is preserved and no :intensity artifacts are created in the processed image. This processed image is the output image from the ray tracer.

III IMPLEMENTATION ISSUES

In this section, we discuss implementation details for the special cases .mentioned in the previous section. In addition, we discuss some important pointers for efficient implementation and some assumptions that can be made to simplify the implementation. It should be noted that these assumptions are not essential, i.e. the theoretical discussion in the previous section holds even in the absence of these simplying assumptions.

A The Sweeping Rule

A problem that we encounter while implementing the generalized cylinder ray tracer is the computation of the inverse sweeping rule. Automatic computation of the inverse of an arbitrary function is, in general, not possible. If however, the sweeping rule is restricted to the class of linear functions, the inverse sweeping rule can be computed easily. In this case, the contour of the generalized cylinder can be defined as:

$$\mathbf{c}(u,v) = \begin{bmatrix} c_n(u,v) & c_b(u,v) & 1 \end{bmatrix} = \begin{bmatrix} cs_n(v) & cs_b(v) & 1 \end{bmatrix} S(u)$$

where $S(u) = \begin{bmatrix} a_1(u) & b_1(u) & 0 \\ a_2(u) & b_2(u) & 0 \\ a_3(u) & b_3(u) & 1 \end{bmatrix}$ is the linear sweeping function.

The inverse relationship between the initial cross-section and $\mathbf{c}(u,v)$ is then given by:

$$\mathbf{cs}(v) = \mathbf{c}(u,v)S^{-1}(u).$$

Since S is a 3x3 matrix, S^{-1} can be computed easily. $\phi(u)$ can then be obtained from $\boldsymbol{\rho}(u)$ by applying the inverse sweeping rule as follows:

$$\phi(u) = \begin{bmatrix} \boldsymbol{\rho}(u)^T & 1 \end{bmatrix} S^{-1}(u).$$

To avoid computation of the inverse sweeping rule, we provide an alternative implementation which requests the specification of both the sweeping rule and its inverse as input. This approach removes the linearity constraint placed on the sweeping rule and allows ray tracing of a greater class of generalized cylinders.

B The Frenet Frame

In section II(A), it was mentioned that the Frenet frame is not defined at points where the curvature, $\kappa(u)$, is zero. Such cases have to be given special consideration::

Linear axis: The Frenet frames along a linear axis is given by interpolation of the Frenet frames at both ends of the axis. If the Frenet frames at the ends of the axis are undefined, a unit vector orthogonal to the direction of the axis is chosen as $\hat{\mathbf{b}}(u)$. $\hat{\mathbf{t}}(u)$ and $\hat{\mathbf{n}}(u)$ have their usual definitions. The interpolation of Frenet frames between the two ends of the liinear axis ensures that the local coordinate system that is defined varies smoothly along the axis.

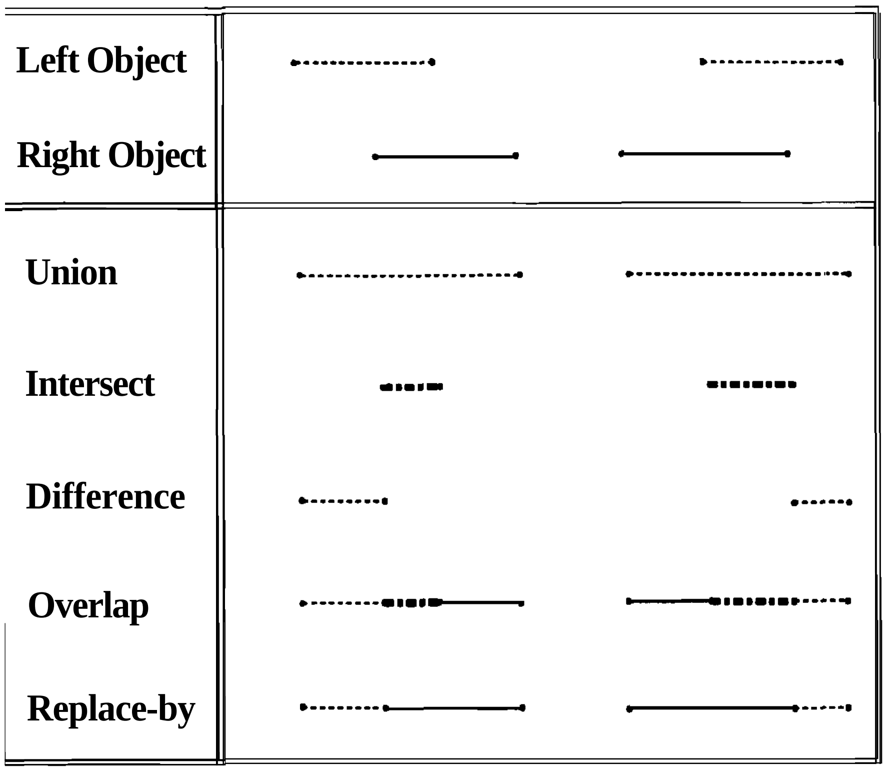
Points of inflection along the axis: If the axis is any arbitrary 3D space curve, the generalized cylinder may not have a smoothly varying surface at points of inflection along the axis. It has been proven that in order to have a smoothly varying local coordinate system, the axis has to be analytic [26]. For our implementation, the axis is not restricted to be analytic, so the surface of the generalized cylinder may not be smoothly varying. Since the Frenet frame is undefined at points of inflection, a neighboring Frenet frame along the axis is used instead.

C Ray Parallel to Contour Plane

As noted earlier, when the ray is parallel to the contour plane, i.e. $\mathbf{a}'(u) \cdot \mathrm{d} = 0$, it is not possible to find an intersection point between the ray and the contour plane. Such points have to be removed from consideration. This deletion process is done before the curves are split at points of inflection and points with vertical and horizontal tangents. If $\mathbf{a}'(u) . (\mathrm{p} - \mathbf{a}(u)) = 0$ also holds true, then the ray lies in the plane of the contour and the intersection points can be found by intersecting the ray with the contour. Otherwise, the ray does not intersect the contour plane with this parameter value and no additional processing is required.

D Representation of Complex Solids

It is not enough to be able to ray trace a single generalized cylinder. The Constructive Solid Geometry (CSG) method [2, 21, 27] is a mechanism by which voulme defining primitives can be assembled into more complex shapes. Primitive objects can be combined using the Boolean set operators Union, Difference and Intersection. In order to deal with X-ray imaging of objects, besides knowing whether a point is in or out of the combined solid, it is also necessary to know the density associated with the point. When two objects overlap, the density of the overlapping region can be taken as the sum of the densities of the objects or as the density of either one of the objects. To handle these possibilities, two extra boolean operators, Overlap and Replace-by, have been defined for the X-ray ray tracer. Table 1 defines the operations.



Line style represents the density of the segment.

Density of left object: -----------------
Density of right object: ————
Density of left object + Density of right object: ▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪

Table 1: CSG operations for the X-ray ray tracer.

E Efficiency issues

To improve the efficiency of the ray tracing algorithm, computations that remain unchanged for each ray are computed once and stored for later use. An example is the subdivision of the initial cross-sectional contour which remains the same even though different rays are cast through the scene. Another pre-processing step that is performed is the estimation of regions where ray-object intersections may occur. Ray tracing is only performed on these regions of the image. Region estimation is done by projecting the **3D** bounding volume of the generalized cylinder onto the image plane. Also, since the ray tracing algorithm is computationally intensive, z-buffer routines have been implemented to generate fast but coarse images for model review before actually rendering the model using our ray tracing technique.

IV EXAMPLE IMAGES

We have presented an algorithm to ray trace generalized cylinders. Besides generalized cylinders, our geometric modeling system can also allow spheres, quadrics (ellipsoids, elliptic cylinders, double cones etc.) and truncated quadrics (quadrics with associated cutting planes) as primitives. These primitives can be combined using the CSG operations as described in the previous section. Our system has the flexibility and versatility to model a wide class of o'bjects which previously had to be approximated by a large collection of simpler primitives [28] or were impossible to be accurately ray-traced. With our system, the user can exercise creativity, artistic and sculpturing talents to create all kinds of **3D** shapes and images. To give some ideas about the capabilities of our system, three example images will be presented.

A A Deformed Object

Figure 8 shows an object that is modeled with a single generalized cylinder. The axis of the generalized cylinder is a **3D** space curve and the cross section varies smoothly from a "star" to a circle and then to a square with rounded corners (see Figure 3). This figure shows the generality of the class of generalized cylinders that can be handled by our system. Our only restriction is that the sweeping rule of the generalized cylinder must be invertible.

B A Rotary Fan

Figure 9 shows a fan with 3 blades. The blades are transformed (rotated and translated) versions of one another. Hence, the model of the fan is easily built once the model of a blade is defined. Each blade is modeled as a generalized cylinder with a cross-section that twists along the axis. Since twist (rotation) can be modeled as a linear function, the sweeping rule is linear and the inverse sweeping rule can be automatically computed. 'The center of the fan is modeled by a truncated ellipsoid.

This example shows the applicability of our system to computer-aided design. Objects which can be decomposed into primitives can be rendered realistically. Since we have full control over the imaging process, it is possible to change the position of the source and generate a stereo pair of images so the designer can see his design in 3D.

C A Knee Joint

Figure 10 is a simulated X-ray image of a knee joint. A total of 6 primitives are used. The femur (upper bone) is represented by a generalized cylinder with subtraction of an ellipsoid. A single generalized cylinder is used to represent the tibia (lower main bone). The fibula (lower small bone) is represented by a generalized cylinder with an ellipsoid representing the head of the fibula. The patella (knee-cap) is represented by an ellipsoid.

Many hurnan organs can likewise be represented by a small number of generalized cylinders. With the ,generalized cylinder X-ray ray tracer, realistic X-ray images of human anatomy can be generated. These images, although not perfectly accurate, are of sufficient accuracy to be used as an inexpensive and convenient source of images for the evaluation of new medical display technology [29] and for the initial teaching & training of radiologists.

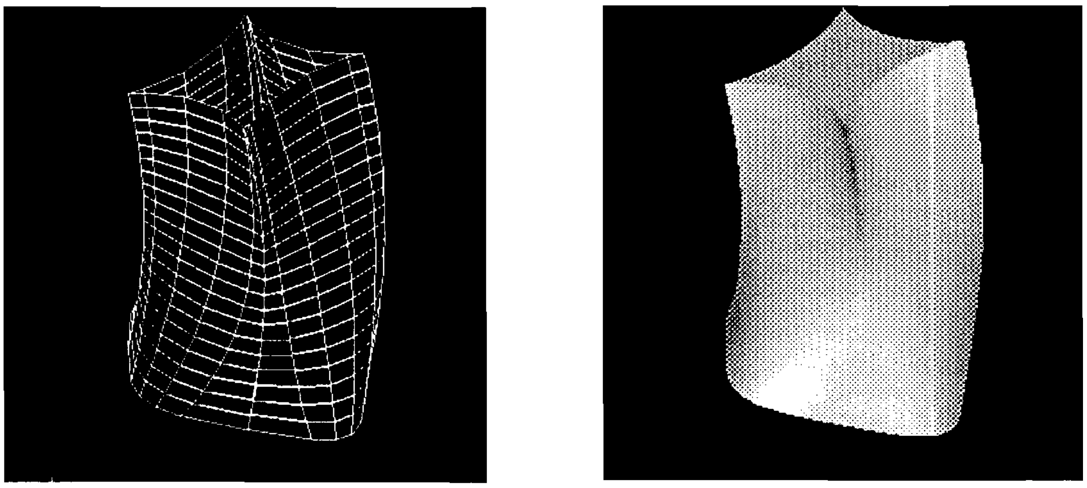


Figure 8: On the left is a wireframe image of a deformed object with no end-planes and on the right is the ray-traced image of the same deformed object with end-planes.

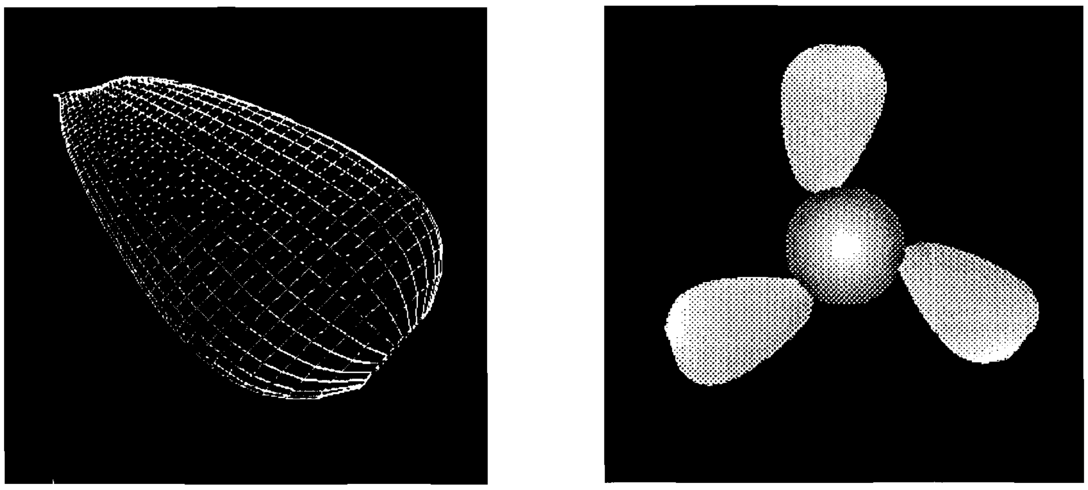


Figure 9: The left shows a wireframe image of a single blade and the right shows a ray-traced image of a rotary fan consisting of 3 blades.

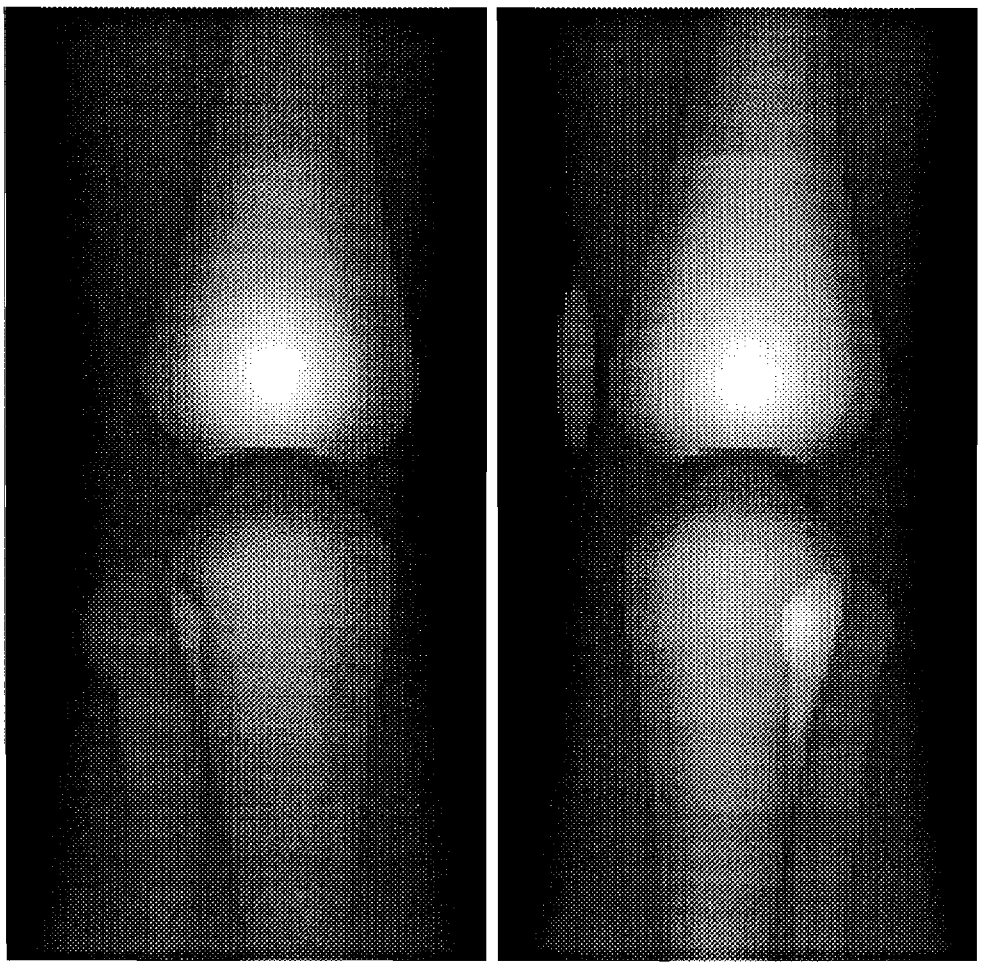


Figure 10: Simulated X-ray images of a knee joint from two views.

In the training of radiologists, the ability to view in different directions is an important aid to understanding the structure of the object. For example, in Figure 10(a), the knee cap (patella) cannot be easily seen. If however, the object is rotated by 90 degrees (Figure 10(b)), the knee-cap can be clearly seen. Using computer generated images, subtle (abnormalitiescan be added by the instructor to test if the trainee can locate and identify the abnormalities. Also, by modeling human anatomical structures using geometric models., a wide range of images corresponding to people of different height, weight and age can be generated for training purposes. The evaluation of medical display technology requires many images of known state (i.e. abnormal/normal). These images can be easily generated using the X-ray ray tracer.

V CONCLUSION

We have presented an algorithm for ray tracing generalized cylinders. The algorithm has been implemented in our geometric modeling system. Our examples show that the number and types of objects that can be ray traced is virtually limitless. The ability to ray trace generalized cylinders opens many new, useful and exciting applications for ray tracing, examples include CAD/CAM, medical education and training and the entertainment industry.

References

[1] R. Kuchkuda, "An introduction to ray tracing," in Theoretical Foundations of Computer Graphics and CAD, pp. 1039–1060, New York: Springer-Verlag, 1987. NATO AS Vol. F-40.

[2] A. S. Glassner, An Introduction to Ray Tracing. London; San Diego: Academic Press, 1989.

[3] A. Watt and M. Watt, Advanced Animation and Rendering *Techniques:* Theory and Praciice. New York, NY: Addison-Wesley, 1992.

[4] T. O. Binford, "Visual perception by computer," IEEE Conference on Systems and *Control*, December 1971, Miami. Invited Presentation.

[5] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," Proceedings Royal Society of London B, vol. 200, pp. 269–294, 1978.

[6] M. P. do Carmo, *Differential* Geometry of Curves and Surfaces. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1976.

[7] G. Farin, Curves and Surfaces for Computer Aided Geometric Design. San Diego: Academic Press, third ed., 1993.

[8] G. J. Agin and T. O. Binford, "Computer description of curved objects," IEEE Trans*actions* on Computers, vol. 25, no. 4, pp. 439–449, April 1976.

[9] R. Nevatia and T. O. Binford, "Description and recognition of curved objects," Artificial Intelligence, vol. 8, pp. 77–98, 1977.

[10] U. Shani, "A 3-d model-driven system for the recognition of adominal anatomy from ct scans," Proceedings of the 5th IJCPR, December 1980, Miami, pp. 585–591.

[11] B. I. Soroka, "Generalized cones from serial sections," computer *Graphics* and Image Processing, vol. 15, pp. 154–166, 1981.

[12] R. A. Brooks, "Symbolic reasoning among 3-d models and 2-d images," Artificial Intelligence, vol. 17, pp. 285–348, 1981.

[13] W. F. Bronsvoort, Direct Display Algorithms for Solid Modelling. PhD thesis, Delft University of Technology, The Netherlands, 1990.

[14] H. B. Bidasaria, "A method of ray tracing a wide class of generalized cylinders with straight line trajectories," CVGIP: Graphical Models and Image Processing, vol. 53, no. 2, pp. 101–107, March 1991.

[15] W. F. Bronsvoort and F. Klok, "Ray tracing generalized cylinders," *ACM* Transactions on Graphics, vol. 4, no. 4, pp. 291–303, October 1985.

[16] W. F. Bronsvoort, P. E. van Nieuwenhuizen, and frits H. Post, "Display of profiled sweep objects," ***The Visual Computer,*** vol. 5, pp. 147–157, 1989.

[17] J. Ponce and D. M. Chelberg, "Localized intersections computation for solid modelling with straight homogeneous generalized cylinders," ***Proceedings of the 4th IEEE Conference on Robotics and Automation,*** April 1987, Raleigh, NC, pp. 1481--1486.

[18] W. M. Boothby, ***An Introduction to Diflerentiable Manifolds and Riemannian Geometry.*** San Diego: Academic Press, Inc., second ed., 1986.

[19] A. H. Barr, "Global and local deformations of solid primitives," ***Computer Graphics,*** vol. 18, no. 3, pp. 21–30, July 1984. Siggraph 1984.

[20] P. A. Koparkar and S. P. Mudur, "A new class of algorithms for the processing of parametric curves," ***Computer-Aided Design,*** vol. 15, no. 1, pp. 41–45, January 1983.

[21] C. G. C. van Dijk, ***Ray Tracing Profiled Generalized Cylinders.*** PhD thesis, Delft University of Technology, The Netherlands, June 1989.

[22] I. D. Faux and M. J. Pratt, ***Computational Geometry for Design and Manufacture.*** Ellis Horwood, 1979.

[23] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, ***Computer Graphics: Principles and Practice.*** Addison-Wesley Publishing Company, second ed., 1990.

[24] E. E. Christensen, T. S. Curry, and J. Nunnally, ***An Introduction to the Physics of Diagnostic Radiology.*** Philadelphia: Lee & Febiger, 1972.

[25] R. C. Gonzalez and R. E. Woods, ***Digital Image Processing.*** Addison Wesley, 1992.

[26] M. M. Lipschutz, ***Theory and Problems of Diflerential Geometry.*** Schaum's outlines series, New York: McGraw-Hill Book Company, 1969.

[27] R. A. Goldstein and R. Nagel, "3-d visual simulation," ***Simulation,*** vol. 16, no. 1, pp. 25–31, January 1971.

[28] J. O'Rourke and N. Badler, "Decomposition of three-dimensional objects into spheres," ***IEEE Transactions on Pattern Analysis and Machine Intelligence,*** vol. 1, no. 3, pp. 295–305, July 1979.

[29] J. Hsu, C. F. Babbs, D. M. Chelberg, Z. Pizlo, and E. J. Delp, "A study of the effectiveness of stereo imaging with applications in mammography," ***Proceedings of the SPIE Conference on Human Vision, Visual Processing, and Digital Dispi'ay IV,*** vol. 1913, Feburary 1993. To appear.