A Rendering Method for Curved Tubular Objects by Using Scan Line Algorithm

Tomoyuki Nishita and Henry Johan
Dept. of Information Science, Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
e-mail: {nis,henry}@is.s.u-tokyo.ac.jp
Tel: +81-3-5841-4106, Fax: +81-3-5803-7288

Abstract

Parametric surfaces and implicit surfaces are generally used for representing curved surfaces in CAD/CG Systems. This paper discusses a curved tubular object (or generalized cylinder) which is a surface swept by a sphere/circle moving along a curve. For the trajectory curve, a 3D Bézier curve is employed, and its radius can be varied along the curve. In general, its surface cannot be defined by a closed form, while a high degree of polynomial must be solved for ray/surface intersection. This paper proposes an effective rendering method which uses a scan line algorithm for detecting curved tubular objects on the projection plane. The calculation of the distance from a point to a curve plays an important role in our algorithm; *Bézier Clipping Method*[11] is employed for this calculation.

Keywords: Curved tubular objects, Swept volume, Bézier curve, Projection for curve, Closest point, Bézier Clipping

If the paper is accepted, one of the authors will present the paper at the Pacific Graphics '99 conference.

1 INTRODUCTION

In many CAD/CG systems, simple primitives such as cylinders and spheres are generally used. The curved tubular objects, a relatively new primitive also plays an important role in modeling/rendering systems. This primitive is useful for representing pipes, poles, ropes, 3D fonts, brass instruments, or internal organs. For modeling a human body, organs of animals or sculptural designs, metaballs are widely used. The problem with this primitive is that it is difficult to represent long thin objects. To model such objects, many metaballs have to be used along the skeleton and ripples appear on the surfaces in most cases. Even though ellipsoidal metaballs are used in some cases, it cannot represent curved surfaces such as banana. A generalized cylinder is a surface swept by a sphere/circle moving along a curve (envelope of swept ball/circle). That is, this surface is an iso-distance surface with respect to the skeleton curve. It is easy to calculate the points on the iso-surface from the points and its normal on the curve. It is, however, difficult to calculate the distance from a point in 3D space to the curve (called *projection for curve*). For line segments or planar faces, the distance calculation is easy, but it is hard to calculate the distance to 3D curves. For metaballs, the iso-density field is used for surface definition. In this case, its surface is implicit, but it cannot be represented by a closed form. For curved tubular objects, it is very hard to represent them by a closed form. In most CAD systems, polygonization is done before rendering. Precise rendering of generalized cylinders without polygonization is desired. This paper discusses the curved tubular objects which satisfy the following:

1. A 3D Bézier curve is used for a skeleton (or trajectory) element; this paper employed Bézier curves as a representative parametric curve.

2. The radius of the cylinder varies along the curve (three types: constant, linear, Bézier function).

3. Two types of surface definition; iso-distance surface (union of spheres on the curve), and iso-density surface such as metaballs (the density function is defined by the distance from the curve).

The generalized cylinder proposed here has the following advantages :

1. Input is easy because only control points and radii are needed.

2. An effective rendering method which uses a scan line algorithm for detecting tubular objects on the projection plane.

In this paper, the main subject is the rendering method, not a modeling method. The key technique here is an effective method for finding ray/surface intersections by clipping the projected skeleton curves by a scan line. As a preprocessing, an effective calculation method for finding the closest point on the clipped curve from a point on the screen is employed. The closest point on the projected curve is used for the initial guess for the calculation of ray/surface intersections.

The calculation of the distance from a point to a curve is in general difficult. We have to solve degree $(2n-1)$ of the polynomial to find the closest point on a degree n curve, so we have problems with computational cost and robustness. We employed an effective algorithm for calculating the closest point on a Bézier curve from a point by using the Bézier Clipping Method [11] which is also used for clipping the curves on the scan line; the Bézier Clipping method uses the convex hull property of Bézier curves and is an iterative method using linear equations with a small number of iterations.

Though the main subject of this paper is a rendering method for 3D generalized cylinders, for easy understanding we discuss the scanning method (scan version) of planar Bézier curves with width (we refer to these as *Fat Curves*).

In this paper, we discuss the calculation method of the closest point on a curve in Section 2, and the rendering method for shaded *Fat Curves* and curved tubular objects in Section 3 and 4, respectively. Lastly, we demonstrate the usefulness of the proposed method by using various examples.

2 BASIC IDEA

2.1 Swept Volume

This paper discusses display methods for 2D Fat Curves and 3D generalized cylinders. These are implicit area and surfaces, respectively. For the latter, a swept volume is defined as the geometric space occupied by a moving object along a trajectory in a given time interval. Sweeping can be generally defined by $S = \bigcup_{t=0}^{1} G(t)$, where S is a swept volume, G is a moving primitive, $\cup$ is the union operator of all instances of $G(t)$ for sweeping parameter t. In this paper, a sphere is used for the moving primitive which is called a generator. The time interval can be normalized to [0, 1]. This is equivalent to the parameter of the Bézier curve. Convolution surfaces are defined by the distance to the base surface (or skeleton). Distance surfaces are surfaces defined by the distance to skeletal elements such as a point, line segments, or polygons. In this paper, a Bézier curve is employed for the skeletal element. The curved tubular objects are the iso-distance surface from the Bézier curves.

There has been a lot of work on swept volume [19, 4, 1, 2, 20, 22, 7, 5, 3, 15]. Recently, Hu and Ling have developed a modeling method for it, but rendering is performed after polygonization [8]. Bloomenthal [2] have proposed a modeling method for convolution surfaces, and eliminated the bulge of swept volume. He solved this by representing blend instead of the union of moving objects [2]. In his method and Crespin's method[4], unfortunately, skeleton curves are approximated by polylines (piecewise linear approximation). Snyder [20] proposed the method, in which a surface is rendered as a collection of triangles after sampling the surface. Neulander[10] proposed more efficient method of dynamically tessellating generalized cylinders (i.e., multi resolutions). The disadvantage for polygonizing (or sampling/meshing) methods is that a large amounts of memory is used in storing the precomputed polygons, and the modeler must choose the sampling densities before the shape is visualized. Polygonizations are typically not guaranteed and may not accurately detect disconnected or detailed sections of the surface. Recently, Maekawa et all. developed the parameterization method for pipe surfaces (skeleton is Bezier curve) without polygonization[15]. But they display the pipe surfaces with line drawings not shaded surfaces. Our purpose is to display realistic shaded surfaces.

We use curves without any approximation, defining basically a swept volume by the union of moving spheres, and also try to represent a swept volume by a blend of spheres.

Though the skeleton curve is expressed by a Bézier curve, its surface cannot be defined by a closed form, while the high degree of polynomial must be solved for ray/surface intersection; for degree n Bézier curve degree $(4n-2)$ polynominal should be solved for intersection test (see Section 4)[9]. Solving this is very hard, so we propose a pre-scanning method of curved tubular objects on the projection plane. That is, the region it occupies is detected by using the distance from each pixel to the projected skeleton curve. This region is similar to the bounding box (or region) used in ray tracing algorithms. For effective extraction of this region, the subdivision of curves overlapped on the scan line is performed.

2.2 Projection for Curves

A Fat Curve and a curved tubular object are defined by the distance from the curve. So the calculation method for projection for curves (i.e., finding the closest point on the curve) plays an important role. There are some previous methods to find the closest point on curve C defined by parameter u to a specified point Q.

1. The subdivision method into sub-spans: The method evaluates points at equally spaced parameter values on a curve, computes the distance of each point from Q, and chooses parameters to be the value yielding the point closest to Q.

2. The numerical method (Newton iteration): The distance from Q to curve $C(u)$ is minimum when the dot product of tangent vector $C'(u)$ and the vector $QC(u)$ is zero (see [17]). That is, when the vector from Q to a point P on the curve is perpendicular to the tangent vector C', the distance PQ is minimum (see Fig.1(a)). That is, the distance from Q to $C(u)$ is minimum when $C'(u) \cdot (C(u) - Q) = 0$ is satisfied. This equation is solved by Newton iteration. For Newton iteration, however, we need an initial guess, and it is not robust for multiple roots (minimal points).

3. Clipping method: The closest point is effectively and precisely solved by using Bézier Clipping which is an iterative method using linear equations. The method converges to the closest points by clipping away the intervals which have no solutions[16].

The first two methods are not sufficient with respect to computational cost. We need a quick method. So we employed the third method. For easy understanding, we would like to describe the basic idea of the method.

Lets consider degree n Bézier curve defined by

$$C(u) = \sum_{i=0}^{n} P_i B_i^n(u), \tag{1}$$

where P_i is control point and B is Bernstein basis polynominal. We define a function $g(u)$ which is given by

$$g(u) = C'(u) \cdot (C(u) - Q). \tag{2}$$

We can obtain the minimum distance when $g(u) = 0$ is satisfied (see function g in Fig.1(b)). C' is degree $(n-1)$, so we have to solve degree $(2n-1)$ polynomial. These cannot be solved analytically in the case of such a high degree curve. There are also computational cost and robustness problems. For higher degree curves, the problems are more serious. The method overcomes them by using the Bézier Clipping Method which was developed for ray tracing Bézier patches [11]. The method converges to the roots by clipping away the intervals which have no solutions. The intervals (see $[u_{min}, u_{max}]$ in Fig.1) are extracted by using the geometric characteristics of the convex hull property of Bézier curves. In this problem we could get the solution through several iterations (3 to 8). Intervals containing a root or roots are extracted by using function g which is converted from C and C' (g can be represented by a degree $(2n-1)$ Bézier function), and curve C is clipped by using the interval.

The basic algorithm is as follows:

1. Convert to function $g(u)$ from $C(u)$.

2. By using convex hull properties of g, extract interval $[u_{min}, u_{max}]$ containing a root or roots (see Fig.1).

3. Clip $C(u)$ by using the extracted interval.

By repeating this process, intervals containing solutions become small, and then the solution can be obtained.

3 DISPLAY OF FAT CURVES

The final purpose of this paper is the rendering of 3D curves (curved tubular object), but for simple description we discuss the display method for planar Fat Curves. This is equivalent to painting with brush with variable width depending on the pressure.

Lets consider a degree n Bézier curve $C(u)$ as a trajectory curve with width $r(u)$; the curve is given by equation (1). The boundary curve Q is defined by shifting the point along normal vector N. That is, offset curve Q by a radius $r(u)$ is defined by

$$Q(u) = C(u) + r(u)N(u), \tag{3}$$

where N is unit normal at point $C(u)$. Though the trajectory curve is a Bézier curve, the boundary curve is not a Bézier curve in most cases [6].

Lets consider a point Q on a scan line. The radius $r(u)$ is calculated by using the closest point on the curve. The closest point on the curve to Q can be obtained when the following equation satisfies zero (i.e., $g(u,t) = 0$).

$$g(u,t) = C'(u) \cdot (C(u) - Q(t)), \tag{4}$$

where $Q(t)$ is a point on the scan line $y = y_s$, that is $Q = (t, y_s)$, t is x-coordinates in this case (see Fig.3). The square distance is expressed by, $(C(u) - Q(t)) \cdot (C(u) - Q(t))$, which should be $r^2(u)$. Then the intersection points also satisfy $h = 0$; h is defined by

$$h(u,t) = r^2(u) - (C(u) - Q(t)) \cdot (C(u) - Q(t)). \tag{5}$$

The inner points should satisfy $h(u,t) > 0$. As shown in Fig.2, lets consider coordinates system (u,v) within the region, where u is parameter at the closest point and v is distance from the closest point. In the region, $v < r(u)$ is satisfied. The evaluation method of u and v are described in Section 2. The intersections between the boundary of the Fat Curve and the scan line (see Q_1 and Q_2 in Fig.3) can be solved by both of equations (4) and (5) (degree $(2n-1)$ for eq.(4), degree $2n$ for eq.(5)). In general, these points can be obtained by the Newton method.

We employ a subdivision method for finding the intersection sections. The scanning process is performed by finding points within distance $r(u)$ to the curve. We can use the following conditions :

1. The scanning range of the Fat Curve is determined by the bounding box of the control points of the Bézier curve because of the convex hull properties (see $[y_{min}, y_{max}]$ in Fig.2).

2. After finding the closest point on the curve, radius $r(u)$ can be obtained by the parameter u at the closest point. If the distance is less than r, the point is inside the region.

3. The intersections between the boundary and the scan line are points whose distances are $r(u)$ to the curve (see Fig.2).

Though radius r is variable, lets consider constant r for simple description. For scanning we introduce a scanning band $B(y_s)$ with height $2r$. B is the horizontal band whose range is $[y_{s1}, y_{s2}]$, where $y_{s1} = y_s - r$, $y_{s2} = y_s + r$ for scan line $y = y_s$ (see Fig.3).

The main problem is how to minimize ranges which have solutions on the scan line. The Bézier curve is clipped by the scanning band. The curve C becomes short which is denoted by C^s. That is, C becomes a straight line segment and so the distance calculation becomes easy. The algorithm is as follows :

1. Clip C by the scanning band B.

2. Obtain a Fat Line of clipped curve C^s.

3. The intersections (see Q_1, Q_2 in Fig.3) exist within the intersection (x_{min} and x_{max} in Fig.3) between the scan line and *Extended Fat line* of C^s.

The Extended Fat line expands the Fat line by r as shown in Fig.3. The definition of Fat Line is described in Sederberg's paper[18].

The clipping method using the scanning band is as follows. The scanning band is defined by

$$y_{s1} \leq y \leq y_{s2}. \tag{6}$$

By substituting equation (1) to the above equation, we can get

$$\begin{aligned} f_1(u) &= \sum_{i=0}^{n} y_i B_i^n(u) - y_{s1} = \sum_{i=0}^{n} f_i^1 B_i^n(u) \geq 0, \\ f_2(u) &= \sum_{i=0}^{n} y_i B_i^n(u) - y_{s2} = \sum_{i=0}^{n} f_i^2 B_i^n(u) \leq 0, \end{aligned} \tag{7}$$

where $f_1^1 = y_i - y_{s1}, f_1^2 = y_i - y_{s2}$.

The interval of parameter u, which satisfies equation (7), can be extracted by the Bézier Clipping method [13]. By using this interval, curve C is clipped. After clipping, the curve becomes short. So the closest point can be easily obtained by the method described in Section 2.

In this paper, intensity variation or texture mapping can be taken into account by using (u, v) parameters.

4 RENDERING CURVED TUBULAR OBJECTS

For the intersection test between the viewing ray and the surface, we can employ a 3D version of equation (4) and (5). In these equations, $Q(t)$ is a 2D point on the scan line, but here $Q(t)$ is a 3D point on the ray. $Q(t)$ is expressed by

$$Q(t) = P_v + tV, \tag{8}$$

where P_v is the viewpoint, V the viewing unit vector, t the distance from the viewpoint.

Leipelt handled the swept surface generated by sweeping a sphere along a path. The sweep path is represented by a polynominal $C(u)$ of degree n, and the radius of the sweep sphere is defined by the polynominal $r(u)$ of degree m [9]. Parameter t in equation (5) can be solved by substituting the ray equation (8) in equation (5). The root t gives the intersections of all sweep spheres. We need minimal t. To get it, he used the first derivative of the function t. The polynominal of the first derivative is of a degree less than or equal to max$\{\ 4n-2, 4m-2\ \}$. That is, the outline of his process is : (1) Calculate the first derivative of a given function, (2) Find all roots of the derivative, thereby locating the global extrema. (3) Evaluate the function at the roots and choose the smallest value of t.

For the case of degree 3 curve with the constant radius, the method has to find all roots of degree 10 polynominal. On the other hand, our method is very effective because we employ degree $(2n-1)$ Bézier function for the intersection test.

For rendering curved tubular objects, we can employ an extension of the method described in Section 3. We handle 3D Bézier curves. The process is mainly done on the projection plane. The difference between the process for 2D curves (Fat Curve) and 3D curves (curved tubular object) is as follows :

1. On the screen, projected Bézier curves are defined by rational Bézier curves.
2. On the screen, the projected area is determined by the distance from the projected curve, but the projected radius is determined by the depth from the viewpoint.
3. The closest point on the projected curve and the closest point on the 3D curve are not always the same.

4.1 Outline of the Procedure

The outline of the procedure for each curve is as follows :

1. Project control points of the 3D curve onto the projection plane (perspective transformation).
2. Determine the scanning range for the projected control points (y_{min} and y_{max} in Fig.2).

3. For each scan line $y = y_s$, detect the overlapped parametric range u between the scanning band B and curve C.

4. Clip curve C by the scanning band B.

5. Extract Fat Line of clipped curve C^s.

6. For each pixel within the intersected interval $[x_{min}, x_{max}]$ between the scan line and the extended Fat Line (see Fig.3),

   (a) Calculate the distance from the curve on the screen(see Fig.4(a)).

   (b) Find the intersection point P_s, between the ray and the surface on 3D space (see Fig.4(b)).

   (c) Calculate normal vector and intensity at the intersection point P_s.

In step 3., the height of the scanning band is determined by the radius from the curve. For the initial radius, we can use the maximum radius which is used for the previous scan line.

After clipping a curve by the scanning band, the clipped curve is rotated so that the viewing ray lay on the Z-axis and is reprojected onto the new projection plane (see Fig.4(b)). By this process, the error between the closest point on the 2D curve and the closest point on the 3D curve become less.

In step 6., there is a possibility of having multiple points which satisfy equation (4). In this case, we have to find all intersections with the ray, and choose the closest point to the viewpoint (see Fig.5).

There is a special case that the projected curve is an exact arc and point Q is located on the center of the arc. Since the number of the closest points is infinite, we employ point-sampling method for detecting the closest points to the ray in 3D space.

4.2 Detection of the Closest Point to Projected Curve on the Screen

Lets consider a Bézier curve defined in an eye-coordinates system (i.e., the viewpoint is the origin of the coordinates system). After a perspective transformation of a Bézier curve having control points $P_i(X_i, Y_i, Z_i)$, the projected curve on the projection plane is defined by a rational Bézier curve. That is, the curve having control points $P_i(x_i, y_i)$ is defined by

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i B_i^n(u)}{\sum_{i=0}^{n} w_i B_i^n(u)}, \tag{9}$$

with control point coordinates, $x_i = R\frac{X_i}{Z_i}, y_i = R\frac{Y_i}{Z_i}, w_i = \frac{Z_i}{R}$, where w_i are the weights of the control points, and R is the distance between the viewpoint and the center of the projection plane. By substituting the coordinates of projected control points into equation (7) and applying Bézier Clipping method, we can get the clipped curves by the scanning band. Though the curve is a rational Bézier curve, the clipping works well [12].

We use an approximation method for using function g in equation (4). In order to avoid calculation error, the coordinate system is rotated so that the vector from the viewpoint to Q coincides with the Z-axis as mentioned before. We discuss here an approach for finding the closest point to a projected 3D curve on an imaginary screen. After rotation, equation (4) is expresed by

$$x'_c(u)x_c(u) + y'_c(u)y_c(u) + z'_c(u)(z_c(u) - z) = 0, \tag{10}$$

where (x_c, y_c, z_c) is the coordinates of a point on C, (x'_c, y'_c, z'_c) is the first derivative at $C(u)$.

If $z'_c < \varepsilon$ we can ignore the third term in the above equation. The rest of it means the condition for finding the closest point to the origin on the screen. We assume the curve as a 2D Bézier curve with control points (x_i, y_i) in equation (9), and substitute C and C' into function g.

If $z'_c > \varepsilon$, after subdividing the curve for getting small z'_c, the finding processes of the closest point are done for each curve. By this method we can get an initial guess of u for the closest point. By using the parameter u at the closest point, we can estimate the closest point on the 3D curve from the viewing ray.

4.3 Intersection Test between Ray and Surfaces

The swept volume is defined by the union of spheres on the curve. We can find the closest point P_c on the projected curve from a current pixel on the screen by using the method described in the previous subsection. We can assume that $P_c(u)$ on the 3D curve is the center of the sphere (see Fig.4). Then we can get the intersection point P_s between the sphere centered at P_c(radius $r(u)$) and the viewing ray V (see Eq.(8)). By setting these points as initial values, we can find the real intersection points by using the Newton method. In this problem, we have to solve two variable degree $2n$ polynomials.

The intersections can be also obtained by finding the minimum t by moving the center of sphere around P_c. That is, we have an alternative method which is effective. Lets consider a sphere whose center is at P_c on the curve and whose radius is $r(u)$. The intersection P_s between the viewing ray and the surface can be obtained as a intersection between the ray and the sphere. The z value of the P_s is obtained by

$$z = z_c(u) - \sqrt{d(u)} \tag{11}$$

where $d(u) = r^2(u) - x^2(u) - y^2(u)$.

The intersection point is projected onto the tangent vector $C'(u)$ at P_c.

If the distance between its projected point and P_c is not short enough, the sphere is moved along the curve by using the distance which is obtained from the inner product of C' and the vector $(P_s - P_c)$. The parameter Δu for moving is obtained by

$$\Delta u = \frac{C'(u) \cdot (P_s - P_c)}{|C'(u)|^2}, \tag{12}$$

where the denominator of (12) equals equation (4) and (10). If Δu is not small ($\Delta u > \varepsilon$), the new parameter is

$$u = u + \Delta u. \tag{13}$$

After repeating this process, a real solution (parameter u_c) can be obtained.

If equation (12) equals zero, equation (4) is satisfied. $|P_s - P_c|$ is $r(u_c)$ because P_s is on the sphere with radius $r(u_c)$. That is P_s is the real root because equations (4) and (5) are satisfied.

In step 6.(c), the unit normal vector at the intersection point P_s can be obtained by (see Fig.4(b))

$$N_s = \frac{P_s - C(u_c)}{r(u_c)}. \tag{14}$$

4.4 Rendering Blended Curved Tabular Objects

In the previous section, we have discussed iso-distance surfaces. In this section, we discuss iso-density (iso-potential) surfaces such as metaballs [21, 14]. The Bézier curve is the source of potential field. In this case, the density d at a point P is defined by the distance from the source (i.e., Bézier curve).

$$d(P) = \sum_{k=0}^{N} f(\frac{v_k}{r_k(u)}) - T = 0, \tag{15}$$

where N is the number of curves, v_k is the distance from the closest point $C(u)$ on curve k, $r_k(u)$ the radius at $C(u)$, and f the density function. In this paper, we employed the degree 6 density function which is developed by Wyvill[21]. T is the threshold. For $v_k > r_k, f = 0$. The surface is defined by points satisfying equation (15).

In order to display iso-surfaces of the curved tubular objects, we have to calculate the intersection between the ray and their surfaces. The density at the intersection point is a given threshold value. The density at point Q on the ray can be determined by the distance from the curve. This distance can be obtained by finding the closest points on the curves. Note that we have to find the closest point for every curve around a point on the ray and the density at the point can be obtained by summing the density due to every curve. For finding the closest points, we can also use Bézier Clipping method.

5 EXAMPLES

We have coded our method in Java language so that we can display same examples interactively by using the Internet. We can easily edit 3D curves. By using the technique of finding the closest points on the curve, we can pick one of the 3D curves and deform it on the screen viewed from arbitraly viewpoint. (The Java applets of our system will be open.)

Fig.6 shows Java applet screen for the interactive input of 3D curves; this is an example for Fig.8(a).

Fig.7 shows examples of Fat Curves. This is an example of calligraphy (brush strokes); this is the character called "Hikari" (light in English). The trajectories are expressed by planar Bézier curves. The variation of density is taken into account. By using parameters (u, v) (see Section 3), the intensity(density) around the boundary is gray though the center is black.

Fig.8 shows examples of curved tubular objects. Fig.(a) is a complicated knot and the texture pattern is added by using the parameter u. Fig.(b) is a spring. Figs.(c) and (d) are examples of metalic pipes; Cook-Torrance shading model is used. Fig.(d) shows curved tubular objects with variable radius.

Fig.10 shows examples of curved tubular objects. Fig.(a) is 3D fonts of Pacific Graphics. Figs.(b), (c), and (d) contain rings. We constructed a ring by using 4 tubular objects. For Figs.(b) and (c), the texture patterns of marble and wood are added. Figs.(c) and (d) show curved tubular objects with variable radius. The rod with wood texture in Fig.(c) consists of 4 tubular objects (Fig.9 shows the changes in radius along the skeletons of the rod). We used Bézier functions as the radius functions. As for the ring, the radius changes linearly along the skeleton curves. The object in Fig.(d) is constructed by using 50 rings. Here, linear functions are employed for the radius calculation.

The numbers of Bézier curves and computation times for the examples in Fig.10 are shown in Table 1. The size of the images is 512x512 pixels. Our original program is written in Java language, but it was converted to C language for the timing test. We used SGI O2 (CPU R10000 195 Mhz) for rendering. In the table, the cpu time for shadowing/texturing is not included. As shown in the table, the proposed method is very effective for the images including many curves. We concluded that the computation time depends on two factors, the number of tubular objects and the area occupied by the tubular objects on the projection screen. When we do not use our clipping algorithm for each scan line, the cpu time for Fig.10(d) was 82.5 sec. That is, the proposed method reduced the calculation time to 16%.

Fig.11 shows iso-density surfaces defined by curved tubular objects. The intersected parts of objects are blended.

6 Conclusions

This paper has proposed a new rendering algorithm for curved tubular objects. As shown in the examples in the previous section, the usefulness of the proposed method is demonstrated. The rendering method proposed here has the following advantages :

1. The method can be applied to both of 2D curves and 3D curved cylinders.

2. Curved cylinders can be rendered without polygonization.

3. The definition of the curved cylinder is easy because only control points and radius are needed.

4. An effective rendering method is proposed, which uses a scan line algorithm for detecting curved cylinders on the projection plane.

5. We can calculate the closest points by using the Bézier Clipping method which uses the convex hull property of Bézier curves and is an iterative method using linear equations with a small number of iterations.

References

[1] J.Bloomenthal, K.Shoemake, "Convolution Surface," Computer Graphics, Vol.25, No.4, 1991, pp.251-256.

[2] J.Bloomenthal, "Bulge Elimination in Convolution Surfaces," Computer Graphics Forum, Vol.16, No.1, 1997, pp.31-41.

[3] W.F.Bronsvoort, P.R.Nieuwenhuizen, F.H.Post, "Display of profiled sweep objects," The Visual Computer,5, 1989, pp.147-157.

[4] B.Crespin, C.Blanc, C. Schlick, "Implicit Sweep Objects," EUROGRAPHICS96, Vol.15, No.3, 1996, pp.155-174.

[5] S.Coquillart, "A Control-Point-Based Sweeping Technique," IEEE CG & A, 1987-11, pp.36-45.

[6] G.Elber, I.Lee, M.Kim, "Comparing Offset Curve Approximation Methods," IEEE CG & A, 1997, pp.62-71.

[7] J.C Hart, "Sphere tracing a geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, 1996, pp.527-545.

[8] Z.Hu, Z.Ling, "Swept Volumes generated by The Natural Quadric Surfaces," Computers & Graphics, Vol.20, No.2, 1996, pp.263-274.

[9] A.Leipelt, "Ray Tracing a Swept Sphere," *Graphics GEMS V*, 1995, pp.258-267.

[10] I.Neulander, M.Panne, "Rendering Generalized Cylinders with Paintstrokes," Graphics Interface98, pp.233-242.

[11] T.Nishita, T.Sederberg, M.Kakimoto, "Ray Tracing Trimmed Rational Surface Patches," Computer Graphics, Vol.24, No.4, 1990-8, pp.337-345.

[12] T.Nishita, K.Kaneda, E.Nakamae, "A Scanline Algorithm for Displaying Trimmed Surfaces by Using Bézier Clipping," The Visual Computer, Vol.7, No.5, 1991-9, pp.269-258.

[13] T.Nishita, S.Takita, E.Nakamae, "A Display Algorithm of Brush Strokes Using Bézier Functions," CG International'93, 1993-6, pp.244-257.

[14] T.Nishita, E.Nakamae, "A Method for Displaying Metaballs by Using Bézier Clipping," Computer Graphics Forum, Vol.13, No.3, 1994-9, pp.271-280.

[15] T. Maekawa, N.M.Patrikalakis, T.Sakkalis, G. Yu, "Analysis and applications of pipe surfaces," CAGD, 15, 1988, pp.437-458.

[16] R.Matsuda, T.Nishita, "Modeling and Deformation Method of Human Body Model Based on Range Data," Proc. of International Conference on Shape Modeling and Applications, 1999, pp.80-87.

[17] L.Piegle, W.Tiller, The NURBS Book, 1995, pp.229-234.

[18] T.Sederberg, T.Nishita, "Curve Intersection Using Bézier Clipping," CAD, Vol.22, No.9, 1990-11, pp.337-345.

[19] A.Sourin, A.Pasko, "Function Representation for Sweeping by a Moving Solid," *IEEE Trans. on Visualization and Computer Graphics*, Vol.2, No.1, 1996, pp.12-18.

[20] J.M.Snyder, J.T.Kajiya, "Generative Modeling: A Symbolic System for Geometric Modeling," Computer Graphics, 26, 2, 1992, pp.369-378.

[21] B.Wyvill, C.McPheeters, G.Wyvill, "Data structure for soft objects," *The Visual Computer*, 2, 1986, pp.227-234.

[22] W.P.Wang, K.K.Wang, "Geometric Modeling for Swept Volume of Moving Solids," IEEE CG & A, 1986, pp.8-17.

Table 1: The numbers of Bézier curves and CPU times for Fig.10.

Figure	number of curves	cpu time(sec.)
(a)	47	4.62
(b)	56	7.25
(c)	8	5.28
(d)	200	13.03

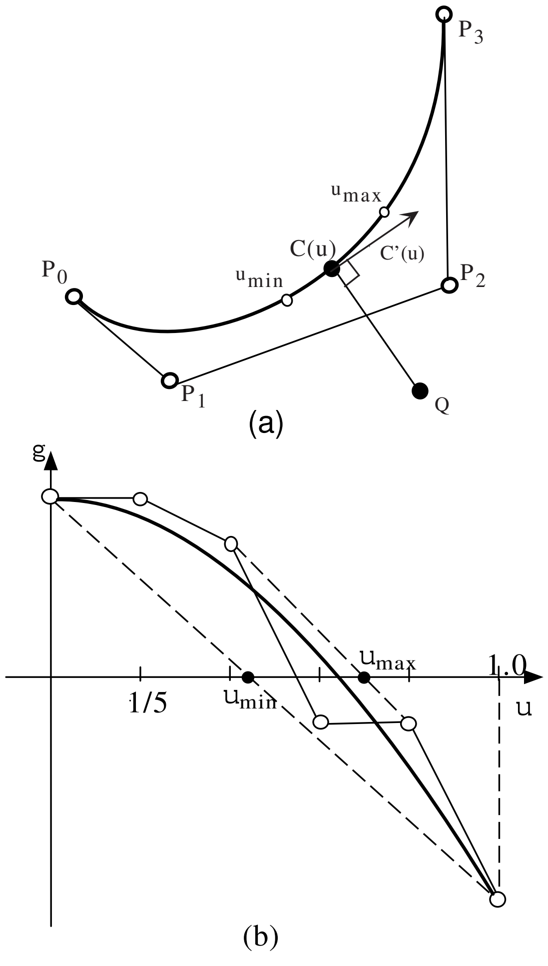


Figure 1: Finding the closest point on a curve. (g is degree 5 Bézier function for a cubic curve)

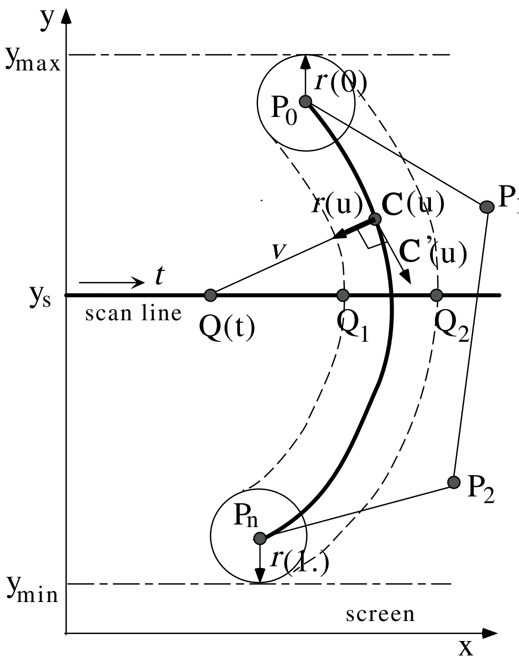


Figure 2: The distance from Q to curve C.

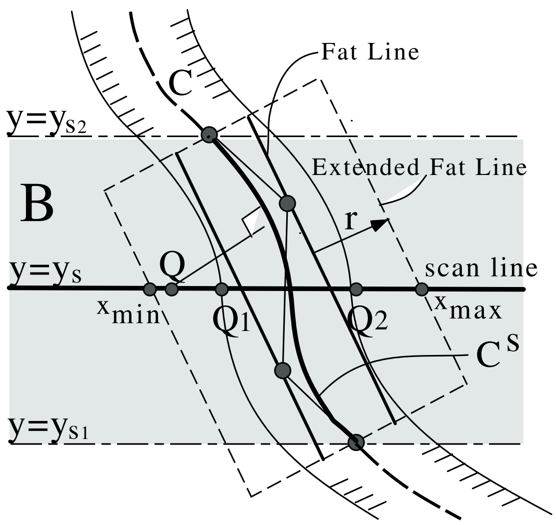


Figure 3: Subdividing curve C by scanning band.

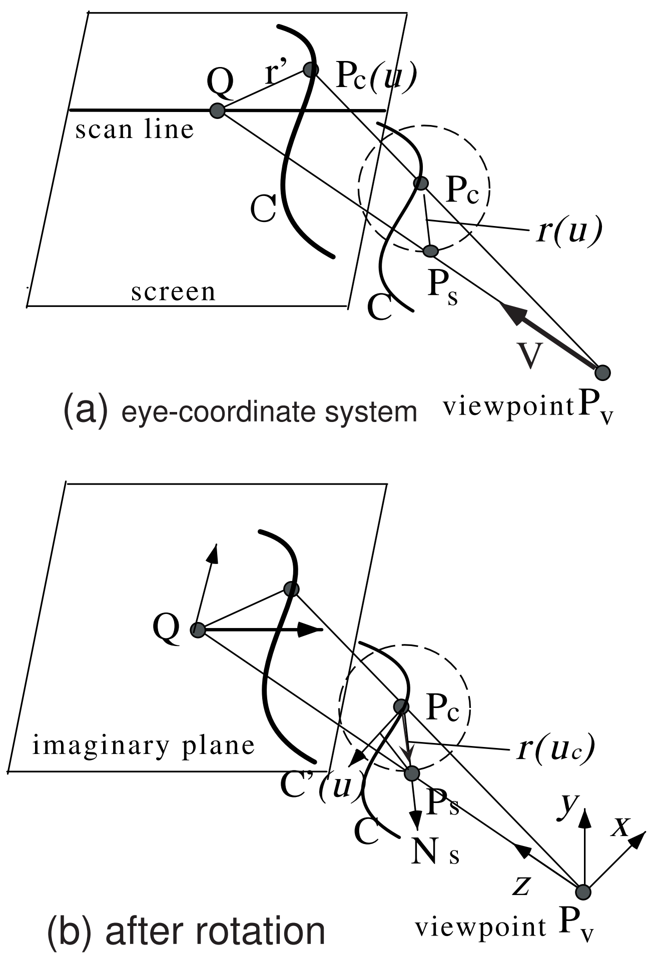


Figure 4: The closest point on the screen and the intersection with the ray.

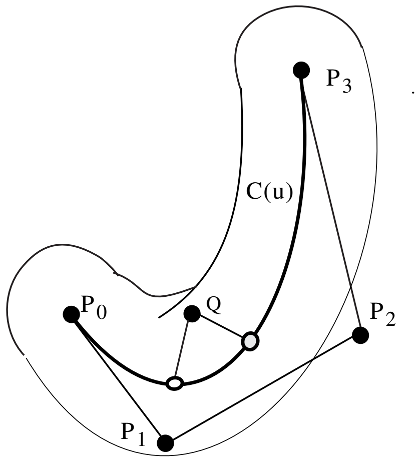


Figure 5: Multiple closest points to a curve on the screen.

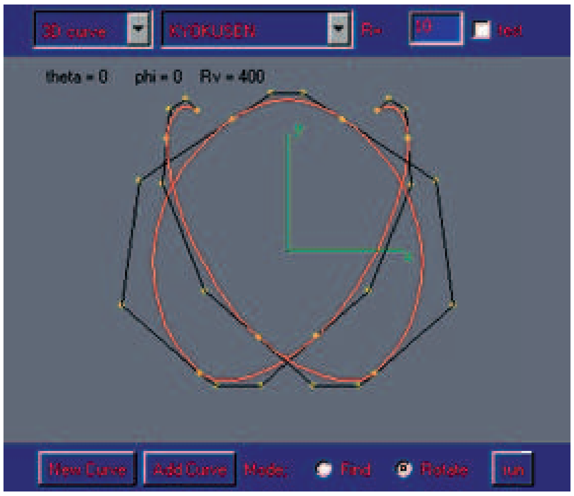
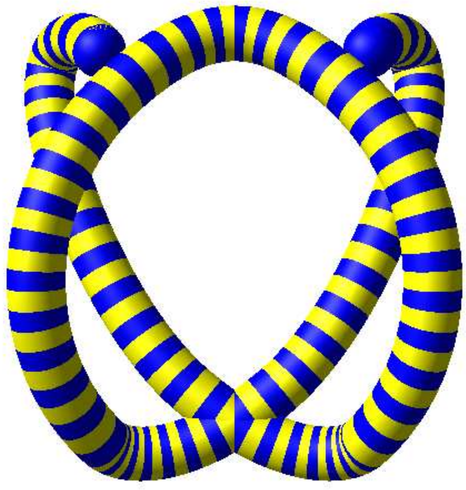
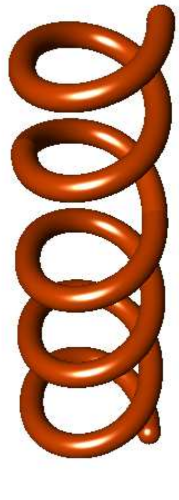


Figure 6: Java Applet for interactive input of Bézier curves.

Figure 7: An example of Fat Curves.



(a)



(b)

Figure 8: Examples of curved tubular objects (1).

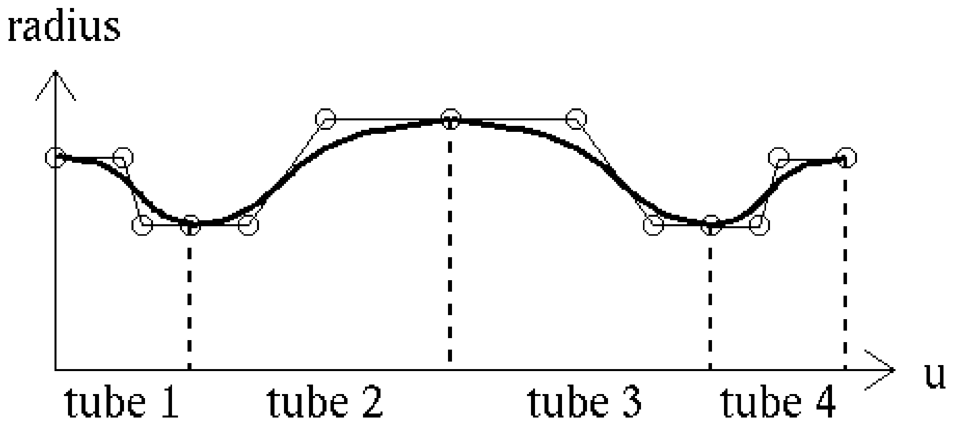


Figure 9: Changes in radius along the skeletons of the rod in Fig.10(c).