# A Display System for Bezier Surfaces and Metaballs using *Bezier Clipping*

Tomoyuki Nishita
Faculty of Engineering, Fukuyama University,
985 Higashimura-cho, Fukuyama, 729-02 Japan
E-mail: nis@eml.hiroshima-u.ac.jp
WWW: http://www.eml.hiroshima-u.ac.jp/~nis

**Abstract**

Displaying objects with high accuracy is important in CAGD (Computer-Aided Geometric Design) and for the synthesis of photo-realistic images. The representation of free-form surfaces can be classified into two: parametric surfaces, such as Bezier patches and implicit surfaces, such as metaballs (or blobs). This paper discusses display methods for both Bezier patches and metaballs. Traditionally, polygonal approximation methods have been employed to display parametric surfaces. This paper introduces various display methods for trimmed Bezier patches without polygonal approximation by using raytracing, scanline algorithm, or hidden line elimination. In most display systems, parametric patches and Metaballs can not be displayed by a single program. In commercial software, to display both of them polygonization is used. This paper introduces a display system for both of them without polygonization. In our system, for hidden surface removal and shadow detection for both types of surfaces, the same idea, called *Bezier Clipping,* is employed. The Bezier Clipping can be also applied to various lighting simulations such as radiosity method.

## 1. Introduction

Accurate and photo-realistic rendering methods for curved surfaces are discussed here. The representation of free-form surfaces can be classified into two categories: parametric surfaces and implicit surfaces. For the former, Bezier patches, B-spline patches, and NURBS are used. For the latter, algebraic surfaces and a set of density functions such as metaballs (or blobs) are used. This paper discusses the idea of Bezier clipping and its application to various rendering techniques.

Bezier clipping can be applied to hidden line/surface removal of Bezier patches. Bezier clipping can be also applied to raytracing of metaballs. Then both types of surfaces can be displayed by a single program. The Bezier clipping technique can be applied not only to hidden line/surface removal but also to various shading effects as described in section 2.1. Thus, the characteristics of the system described here are as follow:

1) Both of parametric and implicit surfaces can be displayed with high accuracy (i.e., without polygonization).

2) For parametric surfaces, Bezier patches, B-spline patches, and NURBS

can be handled even though rendering is performed after converting to Bezier patches.

3) Parametric surfaces are basically rendered by the scanline algorithm, even though raytracing is employed for shadow detection, reflection/refraction.

4) Raytracing is used to render metaballs.

5) Using metaballs allows effective display of various types of objects such as creatures, human bodies, liquid (water droplets), and natural objects such as clouds (volumetric objects).

6) Various shading effects, such as radiosity, can be simulated for parametric surfaces (e.g., various light sources such as cylindrical/curved light sources).

7) Optical effects, such as caustics on curved surface and light scattering due to cloud particles, can be simulated.

This paper discusses the basic idea of Bezier clipping in section 2, hidden line removal for Bezier patches in section 3, display of Bezier patches in section 4, illumination models in section 5, optical effects on surfaces in section 6, and finally demonstrates the effectiveness of the display system by depicting realistic images.

## 2. Basic Idea of Bezier Clipping

Bezier clipping is an iterative method which takes advantages of the convex hull property of Bezier curves, and iteratively clips away regions of the curve which don't intersect the line. Thus we can refer to this method as an interval Newton method. Bezier clipping converges more robustly with the polynomial's solution than does Newton's method. This method was first developed for raytracing Bezier patches[Nishi90].

The advantages of this technique are as follows:

(1) Applicable to a high order of polynomial (and rational functions)
(2) Robust
(3) No initial guess necessary
(4) All solutions within specified range
(5) Minimum/maximum root available if necessary
(6) Quick test for non-intersection
(7) High-degree Bezier function/curves solved by iterations using only linear equations (i.e., Bezier clipping uses only linear equations in each iteration).

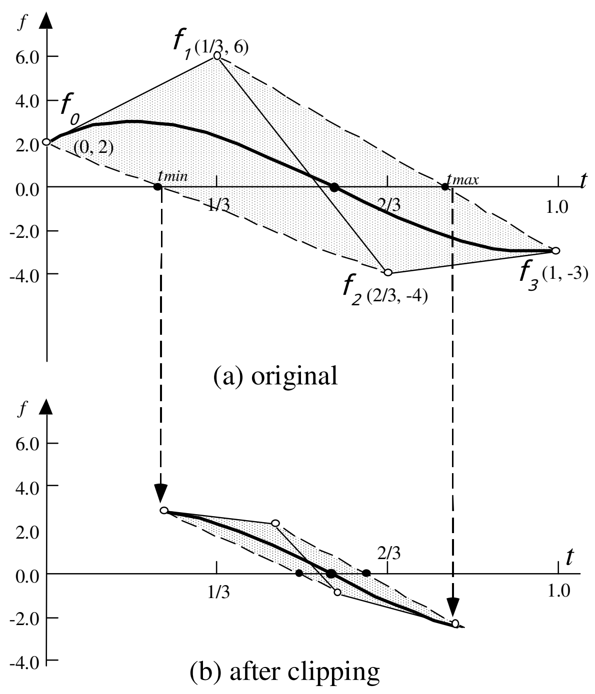
The Newton method is often used for numerical analysis, but it requires a suitable initial guess, and it is difficult to be sure of finding all solutions. Bezier clipping overcomes these problems.

### 2.1 Applications of Bezier Clipping

The following are applications of Bezier clipping.

(1) Root finder for polynomials
(2) Basic geometric problems: curve/curve intersection[Seder90], curve /surface or surface/surface intersection[Seder91]
(3) Hidden surface removal for parametric surfaces: raytracing[Nishi90], scanline algorithm[Nishi91a], hidden line algorithm[Nishi92b]
(4) Hidden surface removal for metaballs[Nishi94a]
(5) Shading models: cylindrical light sources[Nishi92a], curved light sources and radiosity[Nishi94b], optical effects on curved surfaces such as caustics[Nishi94b] or water drops[Kaneda96], natural phenomena such as clouds[Nishi96]
(6) 2-D computer graphics: scan conversion of curved regions, outline fonts [Nishi91b], brush strokes, watercolor painting[Nishi93a], morphing [Nishi93b]

As described above, Bezier clipping can be applied to many fields. This paper will focus on 3-D rendering, the details of (6) are omitted.



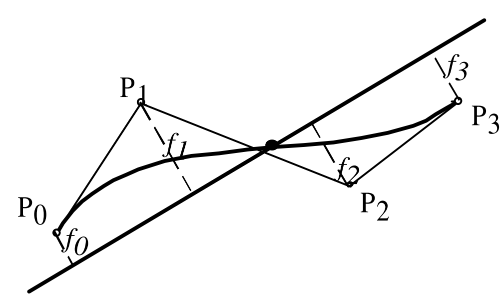




Fig. 1: Solving to polynomial by Bezier clipping Fig. 2: **Curve/line intersection**.

## 2.2 Solving to Polynomials

Polynomials can be converted to a Bezier curve. For example, a degree three polynomial can be converted to the cubic Bezier function. Fig. 1(a) shows the polynomial ($f(x) = 24x^3-42x^2+2x+2$) converted to the following a cubic Bezier function;

$$f(t)=\sum_{k=0}^{3} f_k B_k^3(t) \tag{1}$$

where $(k/3, f_k)$ (k=0,..,3) are control points of the Bezier function, and $B_k^3$ is the Bernstein function. The shaded region is the convex hull of the curve. The root of the curve always exists within the intersection between the axis and the convex hull; that is, the interval $t_{min}$ and $t_{max}$. By clipping the curve at $t_{min}$ and $t_{max}$, we can get a new curve with thinner convex hull as shown in Fig. 1(b). As the remaining part of the curve approaches a straight line, the intersection interval between the convex hull and the $t$ axis rapidly narrows at the next step. By repeating this process we can find the root.

Iteration terminates when the intersection interval between the convex hull and the t axis is smaller than the given tolerance. In the first step, in the example of Fig. 1, the interval is 0.55, but in the third iteration, the interval is only 0.0003. After three iterations we can find the intersection point.

If the intersection between the convex hull and the axis is relatively large, there is a possibility of multiple roots. In that case, the curve is subdivided at the mid point into two curves. And Bezier clipping can then be applied to each curve.

### 2.3 Curve-Line Intersection

Fig. 2 shows a line and a cubic Bezier curve. The Bezier curve with control points $P_k(x_k,y_k)$ is expressed by the following equation.

$$\begin{aligned} x(t) &= \sum_{k=0}^{3} x_k B_k^3(t) \\ y(t) &= \sum_{k=0}^{3} y_k B_k^3(t) \end{aligned} \tag{2}$$

And the distance from any point $(x,y)$ to the line is expressed by

$$d(x,y)=ax+by+c \tag{3}$$

By substituting $x$ and $y$ of the curve equation to the line equation, we can get the following Bezier function.

$$\begin{aligned} d(t) &= \sum_{k=0}^{3} f_k B_k^3(t) \\ f_k &= ax_k + by_k + c \end{aligned} \tag{4}$$

where $f_k$ is equivalent to the distance between the control point $P_k$ and the line. Equation (4) is called distance function. (a,b) is the unit normal of line ($a^2+b^2=1$). Fig. 1(a) shows the distance function expressed by the Bezier function. Parameter $t$ at the intersection with the $t$-axis gives us the intersection between the line and the curve. Bezier clipping solves this intersection.

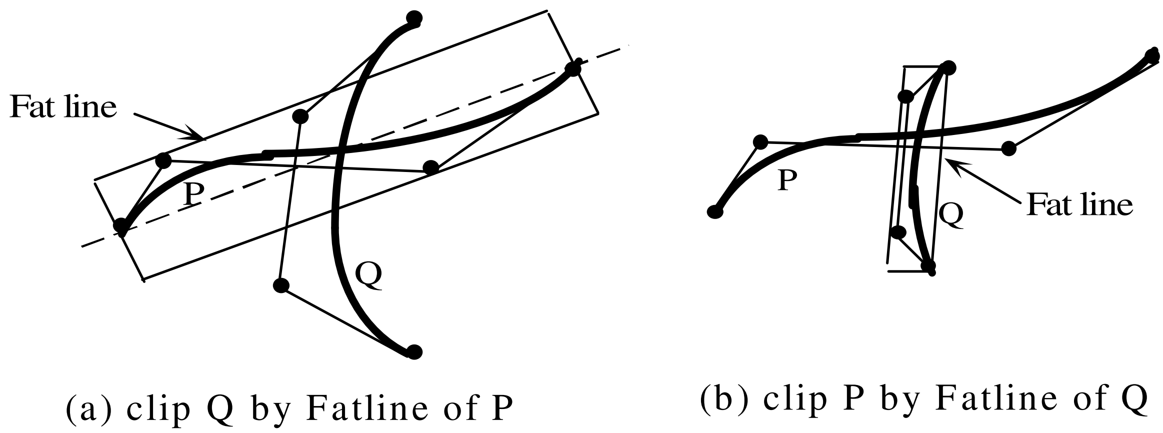



Fig. 3: Curve/curve intersection test by using Fatline.

### 2.4 Curve-Curve Intersection

For curve/curve intersection test, we can introduce the idea of *FatLine*, which is a bounding box of the curve[Seder90]. Let's consider curve P and curve Q shown in Fig. 3. Curve Q is clipped with the FatLine of curve P. This gives us the small curve of P. Curve P can be clipped with the FatLine of curve P. By repeating this process, we can find the intersection point. See reference [Nishi92] for details.

## 3 Hidden Line Removal for Trimmed Bezier Patches

The following problems must be solved for hidden line removal: (1) curve/curve intersection, (2) curve/surface intersection, (3) surface/surface intersection, (4) silhouette extraction.

Bezier Clipping can be used for all of the above problems[Nishi92b]. The curve/curve intersection is a major test. Our system can display trimming curves, curves on surfaces, and silhouette curves. Let's consider drawing iso-parametric curves of a cubic Bezier patch. The projected curve is expressed by a cubic rational Bezier curve, and the projected trimming curve (or a curve on a surface) is expressed by a degree 18 rational curve. Thus, a high degree of polynominal should be solved for hidden surface removal. For such high degree curves, Fatline method based on Bezier clipping is useful (see 2.4). See reference[Nishi92b] for details of problems (2), (3), and (4).

## 4. Display of Trimmed Bezier Patches

In this section, we discuss two hidden surface removal methods for trimmed Bezier patches.

### 4.1 Raytracing for Trimmed Patches

Let's consider previous work on hidden surface removal of parametric surfaces.

Solutions to the ray/patch intersection problem can be categorized as being based on subdivision or numerical techniques. Whitted[Whitt80] first developed the subdivision method. Kajiya's algorithm[Kajiya82] reduces the problem of intersecting a bicubic patch with a ray into one of finding the real root of a degree 18 polynomial. Our method[Nishi90] belongs to the subdivision method. After our paper was published, Fournier[Fourn94] used Chevyshev basis functions to speed up the ray/patch intersection test. The properties of Chevyshev polynomials result in the computation of better and tighter enclosing boxes. Kim[Kim95] has expanded our method. He reduced the amount of computation as much as possible by trying to find only the nearest point instead of computing them all. He built a BSP tree for each original patch in the preprocessing stage by doing adaptive subdivision over the surface. This binary tree allows us to find which part of the subdivided patch is likely to contain the nearest intersection from the viewpoint.

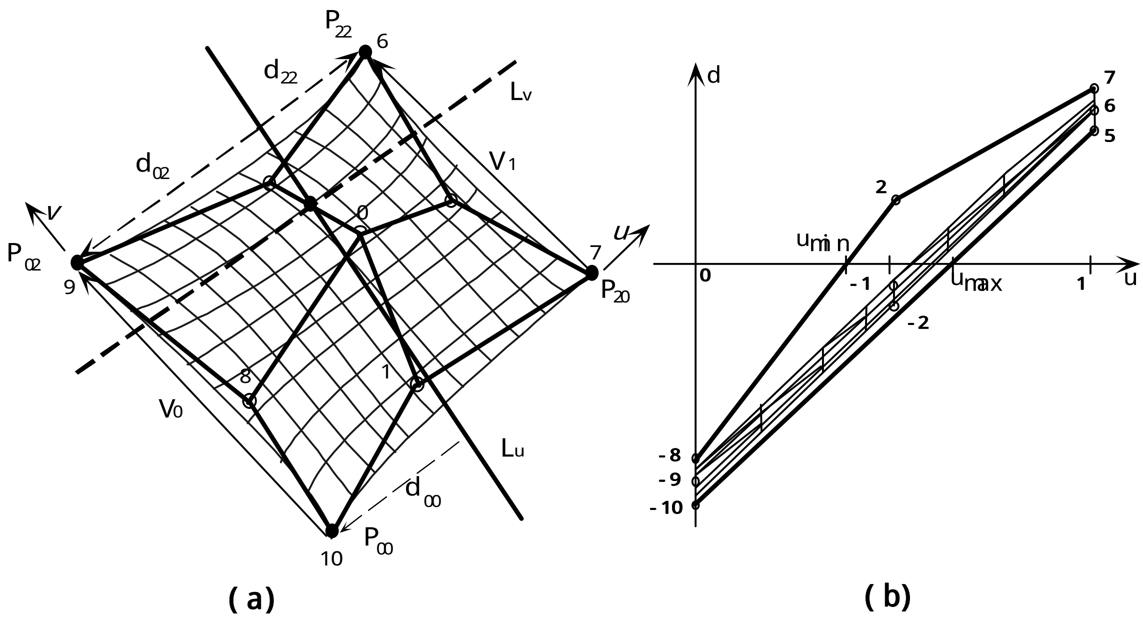



Fig.4: Ray/surface intersection.

Raytracing means to find *(u,v)* parameters from *(x,y)* coordinates on the screen. The viewing ray is the line intersecting two planes. After transforming the Bezier patch to be ray passing through the origin, the two planes become the lines, $L_u$ and $L_v$ (see Fig. 4(a)), passing through the ray (i.e., origin); the line equation passing through the origin is expressed by

$$L_u(x, y) = a_u x + b_u y. \qquad (5)$$

The projected cubic Bezier patch is expressed by

$$x(u,v)=\frac{\sum_{i=0}^{3}\sum_{j=0}^{3}w_{ij}x_{ij}B_i^3(u)B_j^3(v)}{\sum_{i=0}^{3}\sum_{j=0}^{3}w_{ij}B_i^3(u)B_j^3(v)}$$

$$y(u,v)=\frac{\sum_{i=0}^{3}\sum_{j=0}^{3}w_{ij}y_{ij}B_i^3(u)B_j^3(v)}{\sum_{i=0}^{3}\sum_{j=0}^{3}w_{ij}B_i^3(u)B_j^3(v)} \quad (6)$$

where $x_{ij}=X_{ij}/Z_{ij}$, $y_{ij}=Y_{ij}/Z_{ij}$, $w_{ij}=W_{ij}/Z_{ij}$, and $(x_{ij},y_{ij})$ is the projected control point of $P_{ij}(X_{ij},Y_{ij},Z_{ij})$; $W_{ij}$ is weight for the control point.

By substituting *x* and *y* equations in the line equation, we get the following equation.

$$d_u(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}d_{ij}B_i^3(u)B_j^3(v) \quad (7)$$

$$d_{ij}=a_u x_{ij}+b_u y_{ij}.$$

Fig. 4(a) shows the control point distances $d_{ij}$ ($d_{ij}$ for each control point is displayed in the figure). The function *d* can be represented as an explicit surface patch whose control points ($u_{ij}$, $v_{ij}$, $d_{ij}$); $u_{ij}=i/3$, $v_{ij}=j/3$. Even though *d* is function of $(u, v)$, Figure 4(b) is a side view of the $d(u, v)$ patch, the convex hull of the projected control points bounds the projection of the *d* patch. We can find the range having intersections (see $[u_{min}, u_{max}]$ in Fig. 4(b)) by this figure. This process of identifying values $u_{min}$ and $u_{max}$ which bound the solution set, and then subdividing off the regions $u<u_{min}$ and $u>u_{max}$. In a similar manner, we define the process of Bezier clipping in parameter *v*. Our ray-patch intersection algorithm consists of alternately performing Bezier clipping in *u* and *v*. By repeating this process, we can get the small patch which is the intersection point.

### 4.2 A Scan Line Algorithm for Trimmed Bezier Patches

Our hidden surface removal[Nishi91a] is an extension of Lane and Carpenter's algorithm[Lane80]: In their method curved surfaces are subdivided into polygons on each scanline, but small gaps arise between the approximated polygons. We solved this problem by subdividing curved surfaces on each scanline into nearly flat subpatches with curved edges, which are rendered by scanning (see Fig. 5).

These sub-patches are extracted by Bezier clipping method, and the intersection between the curved edge and the scan line can also be solved by Bezier Clipping (see section 2.3).

In our experiments, the scanline algorithm discussed here is 5 times faster than the raytracing algorithm described in 4.1.

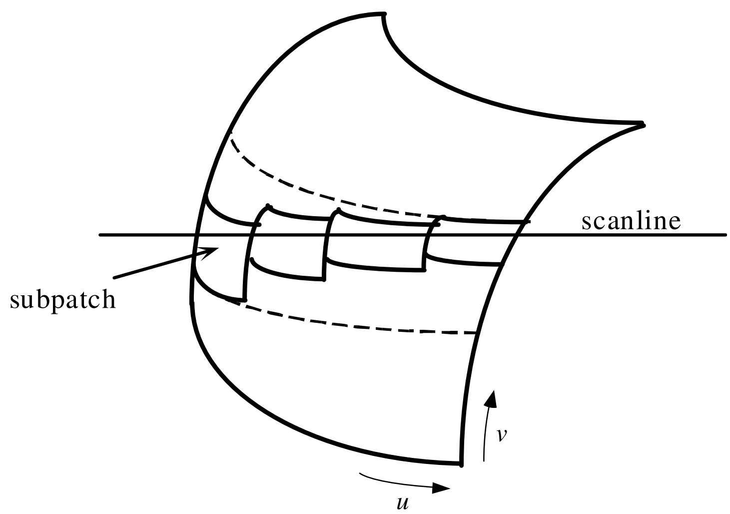



Fig. 5: Scanline algorithm for Bezier patches.

## 5. Displaying Metaballs

The features of metaballs are as follows: (1) the required data for metaballs is typically at least two to three orders of magnitude smaller than that modeled with polygons, (2) metaballs are suitable for use in the CSG model, (3) they are suitable for the representation of deformable objects, making them useful for animation. (4) they are well suited for modeling of human bodies, animals, organic models, and liquids. Because of such a usefulness, many commercial software packages implement metaball modeling techniques. The metaball technique has become an indispensable technique in 3-D graphics software. This modeling technique was first developed by Blinn[Blinn80] who called it *blobs*. In Japan, Nishimura et al.[Nishim85] developed it independently, and called it *metaballs*.

### 5.1 Field function

In the metaball technique, a free-form surface is defined as an isosurface (equi-potential surface) of a field function. The field value at any point is defined by distances from the specified points in space. We used the degree six field function proposed by Wyvill[Wyvill86]. If two balls are placed at the same location, it has twice the volume of the isosurface for a single ball. Thus, for geometric modeling, degree six polynomial function is useful expressed by

$$f_i(r) = -\frac{4}{9}\left(\frac{r}{R_i}\right)^6 + \frac{17}{9}\left(\frac{r}{R_i}\right)^4 - \frac{22}{9}\left(\frac{r}{R_i}\right)^2 + 1 \qquad (8)$$

where $R_i$ is the radius of metaball *i* and r is the distance from a point to the center $P_i(x_i, y_i, z_i)$.

For *n* metaballs, the shape of the curved surface is defined by the points satisfying the following equation.

$$f(x,y,z)=\sum_{i=1}^{n} q_i f_i - T = 0 \tag{9}$$

where $T$ is a threshold, $q_i$ the density values at the center of metaball $i$.

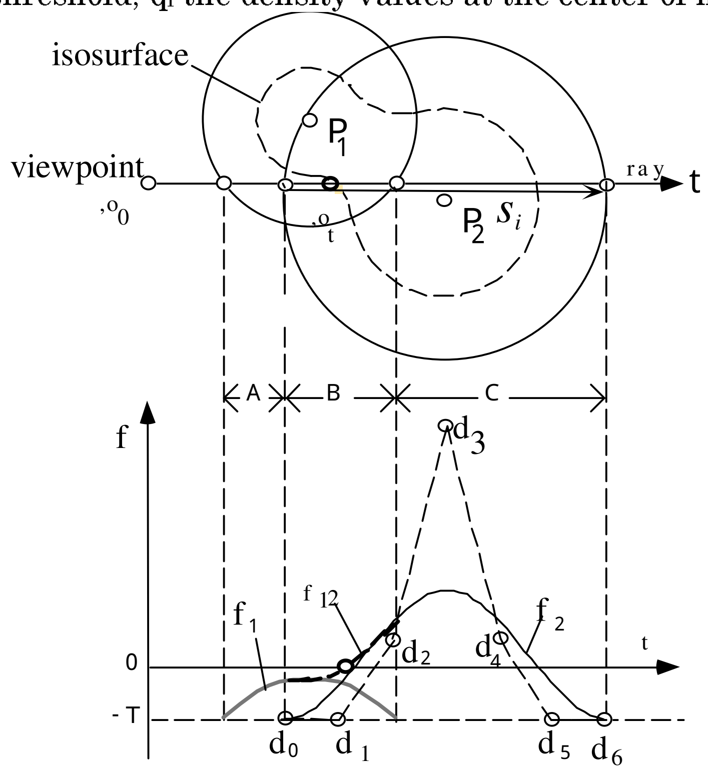



Fig. 6: Density distribution on the ray.

### 5.2 Intersection Test between Ray and Metaballs

The main task for rendering metaballs is intersection tests between rays and isosurfaces. In our algorithm[Nishi94a], the field function on the ray is expressed by Bezier functions, so the root of this function is effectively and precisely solved by Bezier clipping.

Let's discuss the intersection test between a ray and multiple metaballs. Fig. 6 shows a ray and an isosurface defined by two balls and shows the density distribution on the ray. By using parameter $s_i$ (0<s<1) on the intersected interval with ball $i$ (see Fig.6), the density is expressed by

$$d(s_i)=\sum_{k=0}^{6} d_k B_k^6(s_i) \tag{10}$$

where $d_0 = d_1 = d_5 = d_6 = 0$, $d_2 = d_4 = \frac{16}{27}a_i^2$, $d_3 = 8\frac{(8a_i+5)a_i^2}{45}$, and $a_i$ is (intersected length/$R_i$)$^2$ $(0 < a_i < 1)$.

As shown in Fig. 6, Bezier curves, $f_1$ and $f_2$, are clipped by the interval to be tested (i.e., section B in the figure), then both of the clipped curves are composited. This composting of the curves is very simple. It is performed by simply adding each control point $d_{ki}$ belonging to $f_1$ and $f_2$; After composting the curves, the new curve $f_{12}$ is also expressed by a degree six Bezier curve, then the root (e.g., $P_t$ in Fig. 6) is found by using Bezier Clipping.

For opaque objects, only one intersection point closest to the viewpoint is required. Once the first root (i.e., a minimum real root) is found, the process is stopped. In the examples in section 8, the average number of iterations for a single intersection is only 2.0.

## 6. Illumination Models for Curved Surfaces

In the most recently proposed methods, the shapes of light sources and objects are restricted to polygons or simple curved surfaces. Bezier clipping can be applied to many types of light sources such as cylindrical, curved light sources, and skylight. It is also applied to radiosities.

### 6.1 Shading Model for Cylindrical Light Sources

Many offices, classrooms, and factories are lit with multiple fluorescent lamps arrayed in parallel rows on the ceiling. For a cylindrical light source, the disks at the ends of the cylinder are generally quite small compared to the length of the cylinder. The light sources can then be treated as rectangles mapped onto the large rectangle. The illuminance due to each rectangle source can be calculated by contour integration. In general, a number of light sources will be colinear.

When there are obstacles between calculation P and the light sources, shadows must be calculated, for which the hidden surface removal scanline algorithm is employed[Nishi92a] (see Fig.7).

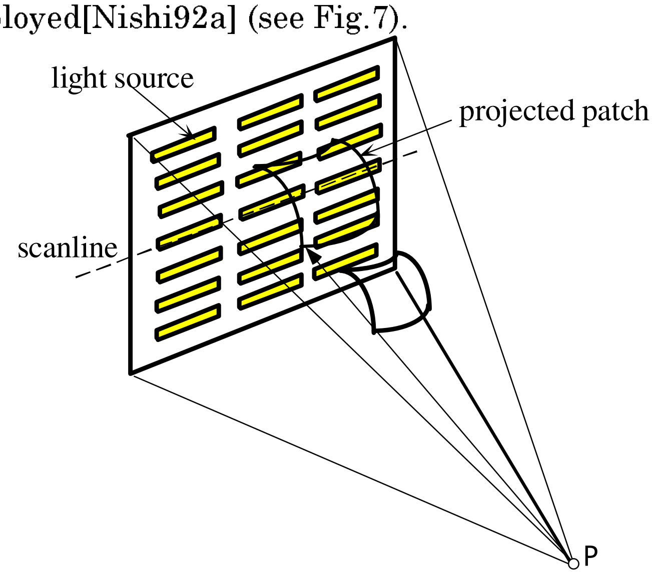



Fig. 7: Shadow detection for cylindrical light sources.

### 6.2 Shading Models for Curved Surface Light Sources

The method described here is developed for radiosity solutions of parametric patches[Nishi93d]. The illumination due to area source is determined by the fraction of the circle covered by projecting the elements onto the base of the hemisphere. If a light source with constant intensity consists of a polygon, the intensity can be obtained using the contour integration method (see Fig. 8). After the calculation of the approximated radiance, the source is subdivided into sub-elements in order to apply the contour integration for boundaries of the subelements.

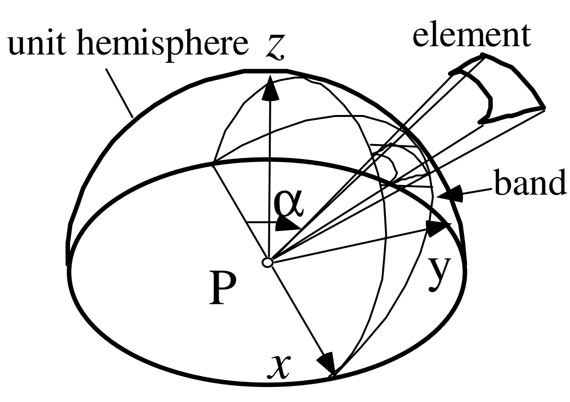
Fig. 8(b) shows the bounded area of a projected light source on the base circle. The radiance is calculated by employing this area. The band is determined by two planes, which contain the x-axis and bound every control point of the element: the interval of the band is obtained by the minimum and maximum angles $\alpha_{min}$ and $\alpha_{max}$ of control points from the $x$-axis.

The form-factor is approximated by the following equation (see Fig.8(b)).

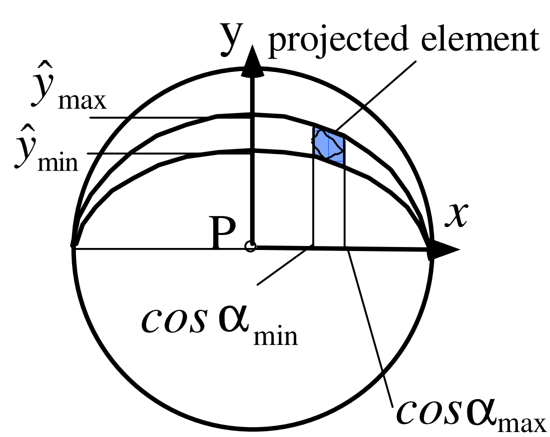
$$F = (\hat{y}_{max} - \hat{y}_{min})(E(\alpha_{min}) - E(\alpha_{max})), \tag{11}$$
$$E(\alpha) = (\alpha - \cos\alpha \sin\alpha)/2\pi .$$

Until the value of this equation becomes smaller than a given tolerance, the subdivision is performed recursively. After that, curved surfaces are polygonized (subelements) to apply contour integration. Then form-factor is calculated by multiplying the radiance of the subelements by the shadow ratio, which is the ratio of visible parts to the whole subelement area. The shadow ratio is obtained by the intersection test between the pyramid composed of the calculation point and the 4 corners of the subelement and each light source. This test is performed on a 2-D plane using Bezier clipping method.

Even though Bezier clipping was developed for ray/surface (or point/surface on 2-D plane) tests, we can employ this for the extraction of the overlapped area between the surface patch and a specified region; the interval of parameters, $u$ and $v$, which overlap the region are extracted, then the patch is clipped by these intervals.



(a) projection onto hemisphere



(b) projection onto circle

Fig.8: Radiance calculation for the curved light source.

## 7. Optical Effects on Curved Surfaces

For realistic rendering, various optical effects should be taken into account. In this section, caustics on curved surfaces, liquid, such as water drops, on curved surfaces, and clouds are described.

### 7.1 Caustics on Curved Surfaces

Some natural phenomena stand out in reflected/refracted light from surfaces of waves in water. Refracted light from water surface converges and diverges, and creates caustic. For these effects, the intensity and direction of incident light on particles plays an important role, and it is difficult to calculate them in conventional ray-tracing because light refracts when passing through waves. Therefore, pre-process tracing from light sources is necessary. Watt[Watt90] developed *backward beam tracing*. The method is a 2 pass solution and requires storage memory for an illumination map or caustic polygons. The method can only handle polygons. Our method[Nishi94], by using a scanline Z-buffer and accumulation buffer, can effectively calculate optical effects on curved surfaces such as caustics without such pre-processing. The visible space within the water at each pixel can be obtained as the front part of the depth of objects stored in Z-buffer.

Free-form surfaces are indispensable for displaying creatures in the water,, and the display of caustics on such curved surfaces is desirable. In our method, metaballs are employed for displaying curved surfaces. The intersection test between the viewing ray and iso-potential surface can be done through ray-tracing as described in section 2.2. And the depth at each pixel is stored in Z-buffer.

### 7.2 Liquids on Curved Surfaces

In many cases, liquids adhere to curved surfaces, e.g., milk in a cup, jelly on a spoon, water droplet on glasses or pots. Metaballs can be applied to express liquids, and parametric patches can be applied to model cups. That is, a display method using both parametric patches and metaballs is required.

In general, liquid adheres along the surfaces. We developed two systems which can models liquid on curved surfaces. One is an interactive system. In our system, the curved surfaces are defined by Bezier patches. The center of metaball is set at the surface of objects. The position of balls are set on the screen by using the mouse, and these position are converted to 3-D positions. Another one simulates flow of water droplets on curved surfaces[Kaneda96]

### 7.3 Clouds

The display of clouds is indispensable for the background images of buildings and flight simulators. Clouds are often displayed by mapping fractal textures onto ellipsoids. However, we developed a display method taking account of light scattering due to cloud particles illuminated by sky light[Nishi96]. The metaball technique is also useful for displaying translucent objects such as clouds/smoke. The intensity of clouds dependents on absorption/scattering effects due to cloud particles. Clouds are defined by density fields, which are modeled by the metaball technique. That is, the surface of a cloud is defined by the isosurfaces of potential fields defined by the metaballs. Shapes of clouds are modeled by applying the fractal technique to metaballs. Then, small metaballs on the surface of the cloud are generated recursively by using the fractal method to form the subtle fringe of the cloud.

In the rendering process for clouds and other forms of a non-uniform density, the numerical integration of the intensities along the ray are required. To detect the section to be integrated, the intersections between clouds surfaces and the viewing ray are required. The intersections of the isosurface (i.e., cloud surface) with a viewing ray are calculated by raytracing based on Bezier clipping as described in section 5.2.

## 8. Examples

Fig. 9(a) is an example of hidden curve elimination for trimmed parametric surfaces. In this drawing, the trimming curves, curves on surfaces are defined cubic Bezier curves in parametric spaces. That is, they are degree 18 Bezier curves in 3-D space. Fig. 9(b) illustrates raytracing for Bezier patches: The mapped maples are created as watercolor painting using Bezier clipping (see [Nishi93a] for brush strokes).

Fig. 9(c) is a car illuminated by many cylindrical light sources[Nishi92a]. The reflected images of light sources give us surface quality. The penumbrae on the floor are realistic. Fig. 9(d) shows a room which is illuminated by the two curved sources and several rectangle sources. In this figure interreflection of light (i.e., radiosity) is taken into account. Fig. 9(e) shows the killer wheels modeled by metaballs. This is one frame from the animation of HDTV size. This figure shows the optical effects in water such as caustics on the killer wheel and shaft of light[Nishi94b].

Fig. 9(f)-(g) show examples of two types of free-form surfaces in the scene. In Fig. (f), the teapot is modeled by 32 Bezier patches and the cup by 4 metaballs (balls with negative densities are used). Fig. 9(g) shows Bezier surfaces (car and tree) and metaballs (clouds and flog). Multiple scattering is taken into account for clouds[Nishi96a].

Fig. 9: Examples of curved surfaces.

## 9. Conclusion

We have introduced a display system for Bezier surfaces and metaballs using *Bezier Clipping.* The Bezier Clipping is a very powerful solver for geometric modeling and shading models. As shown in the examples, the system described here gives us photo-realistic images.

The advantages of the methods described here are as follows:

(1) Both of parametric and implicit surface can be displayed with high accuracy (i.e., without polygonization).

(2) Various shading effects for parametric surfaces can be simulated: cylindrical/curved light sources, radiosity.

(3) In the systems, parametric patches and metaballs can be displayed by a single program.

**Acknowledgment**

I would like to acknowledge Prof. Sederberg of Brigham Young University for his contribution to our discussion on Bezier clipping. The author wish to thank Prof. Nakamae in Hiroshima Technical collage and Prof. Yamashita in Hiroshima University for many valuable discussions. Part of this research is supported by Hiroshima Prefecture.

## References

[Blinn80] J.F.Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Transaction on Graphics*, Vol.2, No.3 (1980) pp.235-256.

[Fournier94] A. Fournier, J.Buchanan, "Chebyshev Polynomials for Boxing and Intersections of Parametric Curves and Surfaces," *Proc. of EUROGRAPHICS'94* (1994) pp.127-142.

[Kajiya82] Kajiya, J., "Ray Tracing Parametric Patches," *Computer Graphics*, Vol.16, No.3 (1982) pp.245-254.

[Kaneda96] K.Kaneda, Y.Zuyama, H.Yamashita, T.Nishita, "Animation of Water Droplet Flow on Curved Surfaces," *Proc. of Pacific'96* (1996)

[Kim95] J.Kim, D.Park, "Efficient Ray Tracing Trimmed Rational Surface Patches," *Proc. of Pacific Grapphics'95* (1995) pp.209-221.

[Lane80] J.Lane, L.Carpenter, T.Whitted, "Scan line Methods for Displaying Parametrically Defined Surfaces," *CACM*, Vol. 23, No.1 (1980) pp.23-34.

[Nishim85] H. Nishimura, M.Hirai, T.Kawai, T.Kawata, I.Shirakawa, K.Omura, "Object Modeling by Distribution Function and a Method of Image generation,", *Journal of papers given by at the Electronics Communication Conference'85* J68-D(4) pp.718-725 (in Japanese).

[Nishi90] T.Nishita, T.W.Sederberg, M.Kakimoto, "Ray Tracing Rational Trimmed Surface Patches," *Computer Graphics,* Vol.24, No.4 (1990), pp.337-345.

[Nishi91a] T.Nishita, K.Kaneda, E.Nakamae, “A Scanline Algorithm for Displaying Trimmed Surfaces by using Bezier Clipping," *The Visual Computer,* Vol.7, No.5 (1991) pp.269-258.

[Nishi91b] T.Nishita, S.Takita, E.Nakamae, “Scan Conversion of Regions Bounded by Bezier Curves," *Proc. of CG & CAD* (1991) pp.198-204.

[Nishi92a] T.Nishita, S.Takita, E.Nakamae, “Shading Model of Parallel Cylindrical Light Sources," *CG International'92* (1992) pp.429-445.

[Nishi92b] T.Nishita, S.Takita, E.Nakamae, “Hidden Curve Elimination of Trimmed Surfaces Using Bezier Clipping," *CG International'92* (1992) pp.595-619.

[Nishi93a] T.Nishita, S.Takita, E.Nakamae, “A Display Algorithm of Brush Strokes using Bezier Functions," *CG International'93* (1993) pp.244-257.

[Nishi93b] T.Nishita, K.Fujii, E.Nakamae, “Metamorphosis using Bezier Clipping," *Pacific Graphics'93* (1993) pp.162-173.

[Nishi93c] T.Nishita, T. Shirai, K.Tadamura, E. Nakamae, ”Display of The Earth Taking into Account Atmospheric Scattering,"*Proc. of SIGGRAPH'93* (1993) pp.175-182.

[Nishi93d] T.Nishita, E.Nakamae, “A New Radiosity Approach Using Area Sampling for Parametric Patches," *Proc. of EUROGRAPHICS'93* pp.385-393.

[Nishi94a] T.Nishita, E.Nakamae, ”A Method for Displaying Metaballs by using Bezier Clipping," *Proc. of EUROGRAPHICS '94*, Vol.13, No.3 (1994) c271-280.

[Nishi94b] T.Nishita, E.Nakamae, “Method of Displaying Optical Effects within Water using Accumulation Buffer," *Proc. of SIGGRAPH'94* (1994) pp.373-379.

[Nishi96] T.Nishita, Y.Dobashi, E.Nakamae, “Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light," *Proc. of SIGGRAPH'96* (1996) (to appear)

[Seder90] T.Sederberg, T.Nishita, “Curve Intersection using Bezier Clipping," *CAD*, Vol.22, No.9 (1990) pp.337-345.

[Seder91] T.Sederberg, T.Nishita, “Geometric Hermite Approximation of Surface Patch Intersection Curves," *CAGD*, Vol.8, No.2 (1991) pp.97-114.

[Watt90] M.Watt, “Light-Water Interaction using Backward Beam Tracing," *Computer Graphics*, Vol.24, No.4 (1990) pp.377-376.

[Whitt80] T.Whitted, “An Improved Illumination Model for Shaded Display,” *CACM*, 23, 6 (1980) pp.96-102.

[Wyvil86a] B.Wyvill, C.McPheeters, G.Wyvill, “Data Structure for Soft Objects," *The Visual Computer*, 2 (1986) pp.227-234.

[Wyvil86b] B.Wyvill, C.McPheeters, G.Wyvill, ”Animating Soft Objects," The *Visual Computer*, 2 (1986) pp.235-242.