Geometric Background of Bezier Clipping Method

Vladimír Palaj, Soňa Kudličková

Faculty of Mathematics, Physics and Informatics, Comenius University
Mlynská dolina, 842 48 Bratislava, Slovak Republic
email: palaj@fmph.uniba.sk, kudlickova@fmph.uniba.sk

Abstract. Finding the points of intersection of a curve and a line or two curves is based on the iterative method. The algorithms discussed in the paper indicate how the geometric properties of the Bezier representation, the distance function and the fat line are applied in the method of Bezier clipping.

Key words: Bezier curve, Bezier clipping method, fat line.

1 Introduction

Determining the points of intersection of two curves has become a complex problem. Many authors have dealt with this problem and there have been many related works [1], [6], [7] [8].

In our approach the problem of intersection is solved by a modified Bezier clipping method proposed by Nishita et al. [1998]. First (in the Section 2) we concentrate on the properties of the Bezier curves and their geometric background to explain all the items that are applied to the algorithms described in the Section 3. The Section 4 presents the future work.

2 Preliminary considerations

2.1 Parametric or functional Bezier curves, algorithms

Let $\mathbf{V}_0$, $\mathbf{V}_1$, … , $\mathbf{V}_n$ be given set of points in the space E (E^2, E^3) and $B_i^n(t)$, $t \in \langle 0,1 \rangle$, be the Bernstein polynomials. An n-th degree Bezier curve is defined by

$$\mathbf{B}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{V}_i \,. \tag{2.1.1}$$

This presentation works with four points $\mathbf{V}_i$, i = 0,1,2,3, called ***Bezier control points***. The points are placed in the plane $E = E^2$, so $\mathbf{V}_i$ (v_i^x ,v_i^y) and parametric representation of the Bezier curve has the form:

$$\mathbf{B}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{3} B_i^3(t) v_i^x \\ \sum_{i=0}^{3} B_i^3(t) v_i^y \end{bmatrix}, \; t \in \langle 0,1 \rangle \,. \tag{2.1.2}$$

In the algorithms of Bezier clipping we need work with ***the functional*** (non parametric) ***Bezier curve*** [2]. To write a Bezier curve in the functional form, we restrict one dimension to be a linear polynomial:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ g(t) \end{bmatrix} \tag{2.1.3}$$

and $g(t) = v_0 B_0^3(t) + v_1 B_1^3(t) + v_2 B_2^3(t) + v_3 B_3^3(t)$, where $v_i \in \mathbb{R}$, $i = 0,1,2,3$, are ***Bezier ordinates*** (scalar values). Parameter $t \in \langle 0,1 \rangle$ or $t \in \langle a,b \rangle$ is evenly spaced, it means $t_i = i/3$ or $t_i = a + i(b-a)/3$, $i = 0,1,2,3$.

The original approach to the Bezier curve (2.1.1) for $n = 3$ defines the curve that only approximates to the Bezier control points $\mathbf{V}_1$, $\mathbf{V}_2$. In our applications the Bezier curve that passes exactly through the given points is required.

Let the Bezier control points $\mathbf{V}_i[i/3, v_i]$, $i = 0,1,2,3$, be given. We need to determine the data points $\mathbf{R}_i[i/3, r_i]$ by the data points $\mathbf{V}_i$ on the Bezier curve. The functional form (2.1.3) is used and set: $r_i = g(i/3) = \sum_{i=0}^{3} \mathrm{B}_i^3(i/3) v_i$, $i = 0,1,2,3$. It means four equations are obtained and written in matrix notation $r_i = I_{CBP}.v_i$, where I_{CBP} denotes 4×4 matrix and it is called ***Inversion Bezier - Polynomic form algorithm*** (***IBP algorithm***).

In order to compute the Bezier control points $\mathbf{V}_i[i/3, v_i]$, more exactly the Bezier ordinates v_i, by the data points $\mathbf{R}_i[i/3, r_i]$, we determine the inverse matrix: $I_{CPB} = I_{CBP}^{-1}$ and the solution for computing Bezier ordinates is:

$$v_0 = r_0,\; v_1 = -15/18\, r_0 + 54/18\, r_1 - 27/18\, r_2 + 6/18\, r_3$$
$$v_2 = 6/18\, r_0 - 27/18\, r_1 + 54/18\, r_2 - 15/18\, r_3,\; v_3 = r_3. \qquad (2.1.4)$$

This step (2.1.4) is called ***Inversion Polynomic – Bezier form algorithm (IPB algorithm)***.

2.2 Convex Hull and Variation Diminishing Properties

The Bezier curves have got many important properties [2], [3], [4]. In the context of the presented algorithms the convex hull property and the variation diminishing property are the most useful.

Convex hull property (CHP): for all $t \in \langle 0,1 \rangle$, $\mathbf{B}(t) \in \{\mathbf{V}_0, \mathbf{V}_1, \ldots, \mathbf{V}_n\}$. It means that every point of a Bezier curve lies inside the convex hull of its defining control points. In the case, for the given points $\mathbf{V}_i \in E^2$, $i = 0,1,2,3$, the convex hull can create quadrangle, triangle, segment line or four-multiple point (more details in [5]).

Variation diminishing property (VDP): for a planar Bezier curve $\mathbf{B}(t)$, the VDP states that the number of intersections of a given line with $\mathbf{B}(t)$ is less than or equal to the number of intersections of that line with control polygon. This expresses the property that a Bezier curve follows its control polygon rather closely and does not wiggle more than its control polygon.

2.3 Distance Function

Suppose the line ℓ in the plane E^2 has the equation:

$$\ell: ax + by + c = 0, \qquad a^2 + b^2 = 1 \qquad (2.3.1)$$

with a unit normal vector. The signed distance from any point (x, y) to the line ℓ is expressed by

$$\mathrm{d}(x,y)=\frac{ax+by+c}{a^2+b^2}=ax+by+c \qquad (2.3.2)$$

By substituting the point on the Bezier curve (2.1.2) into (2.3.2) we get

$$\mathrm{d}(t)=a\sum_{i=0}^{3}B_i^3(t)v_i^x+b\sum_{i=0}^{3}B_i^3(t)v_i^y+c\sum_{i=0}^{3}B_i^3(t)=\sum_{i=0}^{3}B_i^3(t)d_i\,, \qquad (2.3.3)$$

where $d_i=av_i^x+bv_i^y+c$. The function $d(t)$ is called the ***distance function*** and the scalar value d_i represents the ***signed distance*** from the control point $\mathbf{V}_i=(v_i^x,v_i^y)$ to the line ℓ. The function $d(t)$ is a Bezier polynomial function.

2.4 Fat Line

Let $\mathbf{V}_0$, $\mathbf{V}_1$, … , $\mathbf{V}_n$ be the Bezier control points in the plane. The ***fat line*** of the Bezier curve is each strip in the plane with boundary lines to be parallel to the line $\mathbf{V}_0\mathbf{V}_n$. To restrict the number of fat lines of $\mathbf{B}(t)$ we suggest the fat line L of the cubic Bezier curve $\mathbf{B}(t)=\sum_{i=0}^{3}\mathrm{B}_i^3(t)\mathrm{V}_i$, $t\in\langle 0,1\rangle$, as the set of points $X\in E^2$ that satisfy $d_{\min}\le d_x\le d_{\max}$, where $d_{\min}$ = min{d_0, d_1, d_2, d_3}, $d_{\max}$ = max{d_0, d_1, d_2, d_3} and d_x, d_i, $i\in 0,1,2,3$, are the signed distances (2.3.3) from the points X, $\mathbf{V}_0$, $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ to the line $\ell=\mathbf{V}_0\mathbf{V}_3$. The line ℓ is represented by (2.3.1), so the boundary lines ℓ', ℓ'' of the fat line L are expressed as:

$$\ell' : ax + by + c + d_{\max} = 0,\ \ell'' : ax + by + c + d_{\min} = 0.$$

All the points of the convex hull belonging to the Bezier curve coincide with the fat line L = | $\ell'\ \ell''$|.

2.5 Iteration and Tolerance

The algorithms of iterations are based on an iterative process. The iteration terminates when the ***interval of interest*** **|*t*min, *t*max|** is smaller than a given ***threshold tolerance ε*** (Test C). If the interval of interest is relatively large, the Bezier curve is subdivided at the midpoint and the algorithm is applied to each segment.

3 Algorithms of Bezier clipping

The algorithms discussed in this section indicate how the properties of the Bezier representation can be applied to the problems of intersection curve and line or two curves in the plane.

3.1 PT algorithm (Polynom – *t* parametric axis algorithm)

To find the root of the polynomial function.

Input data: ■ suppose we have <u>a polynom</u> defined by a functional equation $y = v(t) = a_0 + a_1 t + \ ... \ + a_n t^n$, $t \in \langle a, b \rangle$

■ a <u>parametric axis *t*</u> by equation

$y = 0$ $(ax + by + c = 0$, where $a = c = 0)$. (Fig.1)

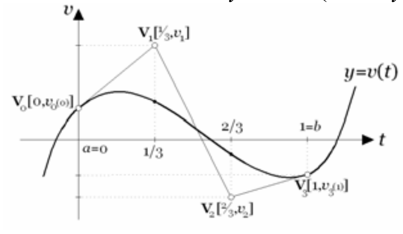

Fig. 1: PT algorithm.

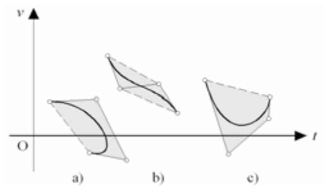

Fig. 2: Bezier curve convex hull with parametric axis *t* intersection.

Step 1. We convert the polynomial function $v(t)$ to the Bezier form, so we apply the IPB algorithm (2.1.4) to the given polynomial function $v(t)$ and get Bezier control vertices $\mathbf{V}_0 = [a, v(a)]$, $\mathbf{V}_1 = [(2a+b)/3, v_1]$, $\mathbf{V}_2 = [(a+2b)/3, v_2]$, $\mathbf{V}_3 = [b, v(b)]$ for evenly spaced parameter *t.*

Step 2. We determine the convex hull of the curve segment control points.

Step 3. Test A (for eliminating unsuitable cases)

Test to determine whether all Bezier curve's control points lie in one of two half-planes with the border line in t-axis; more details are in [5]. For example, in the Fig. 2 the eliminated case b) is illustrated.

Step 4. Test B

Test to determine whether a Bezier curve's convex hull intersects the t axis. In a positive case this gives us an interval of interest $[t_{\min}, t_{\max}]$. See Fig. 3a.

Step 5. For the values $t_{\min}$, $t_{\max}$ we have to determine the ordinates of the points on the Bezier curve $\mathbf{B}(t)$. This gives us the interval of interest on the curve which corresponds to the interval $[t_{\min}, t_{\max}]$. We conclude that the curve segments corresponding to the intervals $t < t_{\min}$ and $t > t_{\max}$ don't intersect the axis *t* and we clip them away (with respect to VDP [Chap. 2.2]). See Fig. 3b.

Step 6. Test C

to compare the change of new found interval $[t^i_{\min}, t^i_{\max}]$, $i = 1, 2, \dots, m$ to the original interval $[a, b]$, resp. to the previous interval $[t^i_{\min}, t^i_{\max}]$ because of $[t^m_{\min}, t^m_{\max}] \subseteq \dots \subseteq [t^i_{\min}, t^i_{\max}] \subseteq [t_{\min}, t_{\max}] \subseteq [a, b]$. Figure 3 shows the first clipping iteration. The Bezier clipping terminates when [Chap. 2.5]:

a) the convex hull of Bezier curve on the corresponding iteration does not intersect the *t* axis – this indicates that there is no intersection of Bezier curve and the parametric axis or the given line.

b) the length of the interval is smaller then a threshold value ε. The algorithm gives as a "return value" interval $[t_{\min}, t_{\max}]$ after each clipping iteration and the situation, when $[t_{\min}, t_{\max}] < \varepsilon$ we regard as a final iteration. Then the intersection is assumed to exist in the centre of the "final" interval of interest [$t^m_{\min}$, $t^m_{\max}$]. Note that we take the centre of interval for final, but approximate value of the root.

Output data: the root of the polynomial function

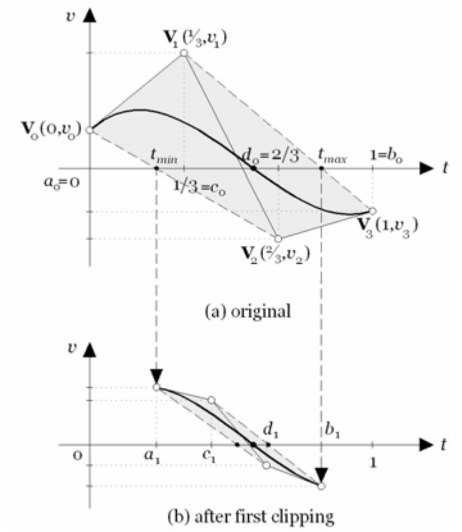

Fig. 3: PT algorithm. (a) *Original.* (b) *After first clipping*

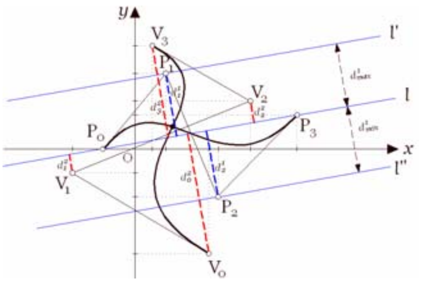

Fig. 4: CC algorithm

3.2 CL algorithm (Curve – Line algorithm)

To find an intersection(s) of planar Bezier curve and a line.

Input data: ■ Bezier curve $\mathbf{B}(t)=\sum_{i=0}^{3}B_{i,3}(t)\mathbf{V}_i$, $t\in\langle 0,1\rangle$.

■ line ℓ by its implicit equation ℓ: $ax + by + c = 0$, and the coefficients *a, b, c* are modified to the form: $a^2 + b^2 = 1$.

Step 1. We have to determine the signed distance $d_i \in \mathrm{R}$ from the control points $\mathbf{V}_i$ to the given line ℓ. [Chap. 2.3]

Step 2. We specify the "new" control points $\mathbf{D}_i = [i/3; d_i]$ for $i = 0,1,2,3$.

Step 3. We apply *PT algorithm* (3.1) for function $d(t)$

Output data: Intersection points of Bezier curve and the line ℓ.

3.3 CC algorithm (Curve – Curve algorithm)

To find an intersection of two planar Bezier curves.

Input data: ■ Two Bezier curves $\mathbf{V}(t)=\sum_{i=0}^{3}B_{i,3}(t)\mathbf{V}_i$, $t\in\langle a,b\rangle$,

$\mathbf{P}(s)=\sum_{i=0}^{3}B_{i,3}(s)\mathbf{P}_i$, $s\in\langle m,n\rangle$.

Step 1. Test D

Test to determine intersections of two Bezier curve convex hulls by the Minmax box method; more details in [6].
Step 2. We construct the fat line *ℓ'ℓ''* [Chap. 2.4] of the curve $\mathbf{P}(s)$, $s \in \langle m, n \rangle$ parallel to the line $\mathbf{P}_0\mathbf{P}_3$. See Fig. 4. *Remark: If the given Bezier curves are of second or third degree we can construct the Fat line to be narrower strip* [5].
Step 3. Using previous algorithms or their parts we find the intersection(s) of Bezier curve with lines *ℓ'*, *ℓ''*. We practice *CL algorithm* and modified *PT algorithm*.
Output data: Intersection points of two Bezier curves

4 Future work

We try to spread the method of Bezier clipping from an E^2 to E^3 coordinate system utilizing next curve representations. There are the main themes and details are in [5]: *1) Bezier clipping for non-Bezier curves*
2) Convergence of Bezier clipping

Acknowledgement

The research was supported by the VEGA Grant No. 1/3024/06.

References

[1] EFRENOV, A., HAVRAN, V., SEIDEL, H.-P.: *Robust and Numerically Stable Bezier Clipping Method for Ray Tracing NURBS Surfaces*. Proceedings of SCCG, Budmerice 2005. ISBN 80-223-2057-9. pp. 123-131.

[2] FARIN, G. – HANSFORD, D.**:** *The Essentials of CAGD*. Natick, MA: A K Peters Ltd., 2000.

[3] KMEŤOVÁ, M.: *Dynamické geometrické programy vo vyučovaní Bezierových kriviek*. Proceedings of the 25th GCG. Janov nad Nisou, 2005, ČR. ISBN 80-7015-013-0. pp. 103-111.

[4] KUDLIČKOVÁ, S.: *Working with Bézier Curves and Controlling Their Shape*. Proceedings of SCG'2004, vol. 13. Kočovce, SR. ISBN 80-227-2133-6. pp. 70-76.

[5] PALAJ, V.: *Algoritmy Bezierovho orezávania pre prieniky čiar*. Projekt dizertačnej práce. Comenius University, Bratislava, 2006.

[6] MARCH, D.: *Applied Geometry and Geometry Modelling*. Springer-Verlag, London, 1999.

[7] NISHITA, T.: *Application of Bezier Clipping Method and Their Java Applets*. Proceedings of SCCG. Budmerice, 1998. ISBN 80-223-0837-4. pp.3-15.

[8] SZARKOVÁ, D., ZÁMOŽÍK, J.: *Priesečníky čiar*. Proceedings of SCG'2004, vol. 13. Kočovce, 2004 , SR. ISBN 80-227-2133-6. pp. 119-125.