FILOMAT (Niš) 9:2 (1995), 233–242

Filomat '94, Niš, October 22-24, 1994
Geometry. Computer Sciences

LINES OF CURVATURE OF FREE FORM SURFACES TRACING

Dušan M. Milošević and Ljubiša M. Kocić

ABSTRACT. *An level-line tracing algorithm, recently developed by the authors is used for Bézier surface interrogation. Namely, for Bézier triangular patches the algorithm is modified so as to trace the lines of constant Gaussian and mean curvature. The map of these lines can be used for better understanding the shape of these patches. The efficacy of the method is illustrated through several examples.*

1. Introduction

The aim of this paper is to obtain curvature level sets of Bernstein-Bézier triangle fragment. Particularly, it gives level sets of Gaussian and mean curvature. This problem is solved by using the algorithm for implicit function graph tracing. Since the, analytic form for Gaussian and mean curvature involve derivatives of two degree, it is necessary to have at least thread order Bézier's fragment.

As far as the applications is concerning, it is enough to mention Computer Aided Geometric Design and Data Visualization. In both topics, the sets of curvature level sets is applied for Bézier surface interrogation. From the curvature level sets one can easily seen the monotonicity, convexity, the existence of saddle points, locations of extrema and gradient intensity of curvature lines. This means having more information about surfaces them self. For example zero Gaussian curvature line share surface on three parts: elliptic (greater than zero) , hyperbolic (smaller) and parabolic (equal). The points of extrema of mean curvature is very important for example in industry because this point of surface is critical in mean of tension.

1991 *Mathematics Subject Classification.* Primary 65D17.

This work was supported in part by the Science Fund of Serbia under grant #0401F.

2. Gaussian end mean curvature

For parametric defined surfaces

$$\vec{x} = \vec{x}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}; \quad \vec{u} = \begin{bmatrix} u \\ v \end{bmatrix} \in [a,b] = T \subset \mathbb{R}^2,$$

where x, y, z are differentiable functions and $T = [a,b]$ is triangle in u, v plane, Gaussian (K) and mean curvature (G) are defined as

$$(1) \qquad K = \frac{LN - M^2}{EG - F^2}, \qquad H = \frac{NE - 2MF + LG}{2(EG - F^2)},$$

where L, M , N , E , F and G fit standard Gauss notation

$$\begin{aligned} L &= L(u,v) = \vec{n} \cdot \vec{x}_{uu}, \quad M = M(u,v) = \vec{n} \cdot \vec{x}_{uv}, \quad N = N(u,v) = \vec{n} \cdot \vec{x}_{vv} \\ E &= E(u,v) = \vec{x}_u \cdot \vec{x}_u, \quad F = F(u,v) = \vec{x}_u \cdot \vec{x}_v, \quad G = G(u,v) = \vec{x}_v \cdot \vec{x}_v. \end{aligned}$$

Bézier surfaces are defined implicitly $B(x,y) = 0$. To use (1) it was necessary to express Bézier surfaces in parametric form

$$\vec{x} = \vec{x}(u,v) = \begin{bmatrix} u \\ v \\ B(u,v) \end{bmatrix}; \qquad (u,v) \in T \subset \mathbb{R}^2,$$

which makes

$$(2) \qquad \begin{aligned} E(u,v) &= \vec{x}_u \cdot \vec{x}_u = 1 + z_u^2 \\ F(u,v) &= \vec{x}_u \cdot \vec{x}_v = z_u z_v \\ G(u,v) &= \vec{x}_v \cdot \vec{x}_v = 1 + z_v^2. \end{aligned}$$

The normal vector $\vec{n}$ is

$$\vec{n} = \frac{\vec{x}_u \times \vec{x}_v}{||\vec{x}_u \times \vec{x}_v||} = \frac{\begin{bmatrix} -z_u \\ -z_v \\ 1 \end{bmatrix}}{\sqrt{1 + z_u^2 + z_v^2}},$$

and because of that

$$(3) \quad L = \frac{z_{uu}}{\sqrt{1 + z_u^2 + z_v^2}}, \quad M = \frac{z_{uv}}{\sqrt{1 + z_u^2 + z_v^2}}, \quad N = \frac{z_{vv}}{\sqrt{1 + z_u^2 + z_v^2}}.$$

Using (2), (3) and equalities $u = x$ and $v = y$ the analytic form for Gaussian and mean curvature for Bézier surfaces on triangular domain can be obtained.

$$\begin{aligned} K &= \frac{z_{xx}z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2}, \\ H &= \frac{z_{yy}(1 + z_x^2) - 2z_{xy}z_x z_y + z_{xx}(1 + z_y^2)}{(1 + z_x^2 + z_y^2)^{3/2}}. \end{aligned} \tag{4}$$

For finding $m - th$ derivative of Bézier's surface in $l - th$ direction, the following formula [1] is used:

$$\frac{\partial^m}{\partial l^m} B_n(f,t) = \frac{n!}{(n-m)!} \sum_{i+j+k=m} P_{ijk}^{n-m}(t) b_{ijk}^m(l). \tag{5}$$

For $m = 1$ one obtains from (5)

$$\frac{\partial}{\partial l} B_n(f,t) = n \sum_{i+j+k=1} P_{ijk}^{n-1}(t) b_{ijk}^1(l).$$

For $l = (-1, 1, 0)$ (which specifies x-axis direction) this yields

$$\frac{\partial}{\partial x} B_n(f,t) = n(P_{010}^{n-1} - P_{100}^{n-1}),$$

while in y-axis direction $l = (-1, 0, 1)$,

$$\frac{\partial}{\partial y} B_n(f,t) = n(P_{001}^{n-1} - P_{100}^{n-1}).$$

For $m = 2$ (5) becomes

$$\frac{\partial^2}{\partial l^2} B_n(f,t) = n(n-1) \sum_{i+j+k=2} P_{ijk}^{n-2}(t) b_{ijk}^2(l).$$

By using this formula one can find

$$\begin{aligned} \frac{\partial^2}{\partial x^2} B_n(f,t) &= n(n-1)(P_{200}^{n-2} + P_{020}^{n-2} - 2P_{110}^{n-2}), \\ \frac{\partial^2}{\partial y^2} B_n(f,t) &= n(n-1)(P_{200}^{n-2} + P_{002}^{n-2} - 2P_{101}^{n-2}), \end{aligned}$$

$$\frac{\partial^2}{\partial x \partial y} B_n(f,t) = n(n-1)(4P_{200}^{n-2} + P_{020}^{n-2} + P_{002}^{n-2} - 4P_{110}^{n-2} + 2P_{011}^{n-2} - 4P_{101}^{n-2}).$$

3. Curvature level-set

Algorithm for tracing graph of a function given implicitly by $f(x, y) = 0$ in some domain is important and attractive problem. Many authors have given important contribution to this problem (see references in [3]). Majority of these methods make use of two stages.

1. Fixing seed points;
2. Tracing the curve.

Fixing seed points

In this stage, the seed points (starting points) are determined for each branch of the curve. This can be done by solving the double sequence of equations $f(x_i, y) = 0,\ i = 0, \dots, N_x$ and $f(x, y_j) = 0,\ j = 0, \dots, N_y$ where x_i and y_j are uniformly distributed along the interval $[a, b]$. The density of "hunting mesh" is controlled by the numbers N_x and N_y. It is recommended to use a predictor-corrector method for solving of each equation above. First, the coarse subdivision of an interval is performed to locate the root and then an iterative method is applied (here modified Regula falsi method is used).

Tracing the curve

In this stage, starting from the seed points, the algorithm traces branches of the curve until some of them leaves the domain T, or until the branch closes up to form a loop. Tracing of each branch is performed by joining the sequence of points $(x_i, y_i),\ i = 0, \dots, m$, where (x_0, y_0) is the seed point for the corresponding branch. The problem of finding next point on the curve can be solved by using derivatives of $f(x, y)$ or without that. If we choose to use derivatives in tracing implicit graph function $f(x, y) = 0$, we must calculate derivatives of (K) and (H) in (4). It mean that we must calculate 3−th derivatives of Bézier's surfaces. It is possible to do provided that we have at least 4−th order Béziers fragment.

a) Algorithms without derivatives

- *Four-point algorithm*

For each point (x, y), the next point in the sequence is calculated by evaluating four neighbour points $(x, y \pm h)$ and $(x \pm h, y)$ and selecting this one which minimize $|f(x, y)|$. In the case when the branch of the curve is closed loop, lying entirely in D, the terminating criteria employes closeness to the starting point. So, each point in the sequence is tested whether or not it is in the ϵ_2 vicinity of the starting seed point. The accuracy may be controlled by testing the inequality $|f(x, y)| < \epsilon$ for each point. If it is not satisfied, the step h is halving until it is.

- *Eight-point algorithm*

This algorithm is similar to the previous one, except that the function is evaluated at eight points $(x, y+h), \ldots, (x+h, y+h)$, and the next point is choosing among them so that $|f(x,y)|$ is minimal.

b) Algorithms with derivatives

- *Algorithm with initial value problem solver*

This stage is consist of M iterations to product the sequence $\{(x_0, y_0), \ldots, (x_M, y_M)\}$. Connecting these points results in a polygonal line being an approximation of the implicit curve. Each point $p_i = (x_i, y_i)$ is tested for being in ϵ_1-vicinity of a singular point i.e.

$$|F'_x(x_i, y_i)| + |F'_y(x_i, y_i)| < \epsilon_1. \tag{6}$$

The logical value of (6) is the main switch in this stage of the algorithm. If it is true, i.e. if p_i is close enough to the singularity, the next point $p_{i+1} = (x_{i+1}, y_{i+1})$ calculates by linear extrapolation, i.e. $p_i = (p_{i-1} + p_{i+1})/2$. Of course, the case when the seed point $p_0 = (x_0, y_0)$ is also the singular point has to be considered separately. Since the preceding point, say p_{-1} is missing, it is taken $x_{-1} = x_0 \pm h$, $y_{i-1} = y_0 \pm h$ where $h > 0$ is the given step. The signs $\pm$ should be chosen arbitrarily if $p_0 \in intD$. But, if $p_0 \in \partial D$ (the border of D), signs should be chosen so that $p_{-1} \in extD$, which gives $p_1 \in intD$.

If (6) is false, p_{i+1} is found by two-stage predictor-corrector method. Then, one solves

$$y' + \frac{F'_x(x_i, y_i)}{F'_y(x_i, y_i)} = 0, \quad y(x_0) = y_0, \tag{7}$$

whenever

$$F'_y(x_i, y_i) \geq F'_x(x_i, y_i),$$

or

$$x' + \frac{F'_y(x_i, y_i)}{F'_x(x_i, y_i)} = 0, \quad x(y_0) = x_0, \tag{8}$$

otherwise. (Note that it can not be $F'_x(x_i, y_i) = 0$ and $F'_y(x_i, y_i) = 0$ at the same time as the consequence of the singular point being far enough). Equations (7) or (8) are solved by Euler method:

$$x_{i+1} = x_i + S_x h, \quad y_{i+1} = y_i - S_x h \frac{F'_x(x_i, y_i)}{F'_y(x_i, y_i)},$$

where $S_x = \text{sgn}(x_i - x_{i-1})$ when $\delta_i = |F'_x(x_i, y_i)| - |F'_y(x_i, y_i)| < 0$, and

$$y_{i+1} = y_i + S_y h, \quad x_{i+1} = x_i - S_y h \frac{F'_y(x_i, y_i)}{F'_x(x_i, y_i)},$$

where $S_y = \text{sgn}(y_i = y_{i-1})$;

So, the point p_{i+1} is obtained and it is corrected by the Newton-Raphson method,

$$\begin{aligned} y_{j+1} =& y_j - \frac{F(x_j, y_j)}{F'_y(x_j, y_j)}, \quad x_{j+1} = x_j \ (\delta_j < 0), \\ x_{j+1} =& x_j - \frac{F(x_j, y_j)}{F'_x(x_j, y_j)}, \quad y_{j+1} = y_j \ (\delta_j \geq 0), \end{aligned}$$

until

$$|F(x_j, y_j)| < \epsilon_2.$$

This completed the algorithm.

Algorithm with initial value problem solver is better in aspect of accuracy and speed (see [4]), but in the case of 3-th order Bézier path it is necessary to use some of the previous algorithms.

4. Examples

The algorithm is tested through many examples and two of them will be presented here. The arrangement of the control points of n-th order Bernstein-Bézier polynomial is accepted to be

$$\begin{matrix} P_{n00} & \cdots & P_{0n0} \\ \vdots & \cdots & \\ P_{00n} & & \end{matrix}$$

Example 1. For the triangular patch given by the control points

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & \\ 0 & & \end{matrix} ,$$

the corresponding level-lines map is given in Figure 1.A (level-lines map is obtain by using algorithm develop in [2]). Figures 1.B and Figure 1.C presents the level-lines map of Gaussian and mean curvature respectively.

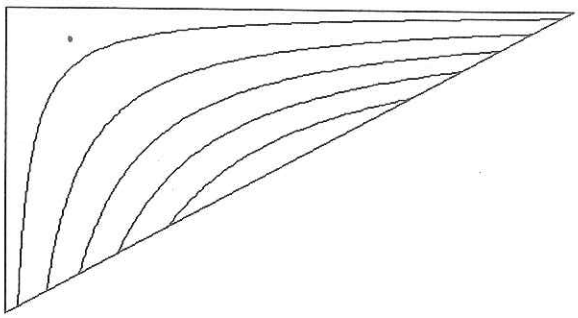

Figure 1.A

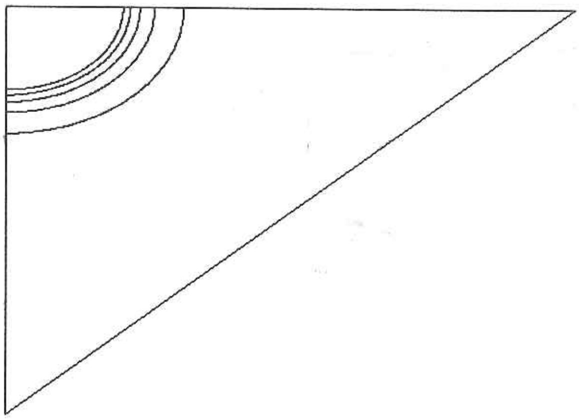

Figure 1.B

Example 2. For the control points

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \\ 0 & 0 & & \\ 0 & & & \end{matrix},$$

the level-lines map for the corresponding patch is given in Figure 2.A. As in the previous example, the Gaussian and mean curvature are shown by Figure 2.B. and 2.C respectively.

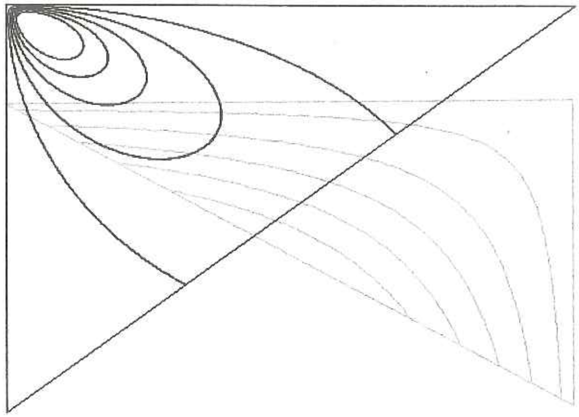

Figure 1.C

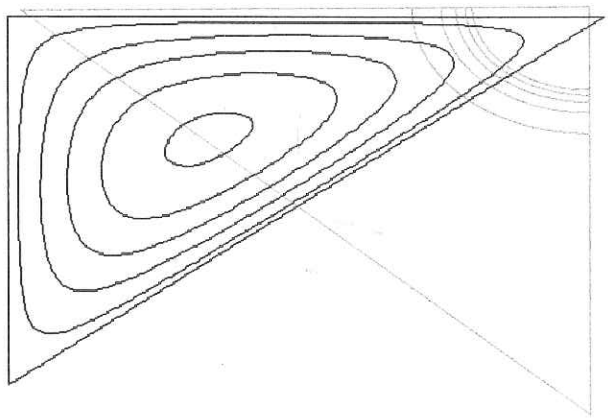

Figure 2.A.

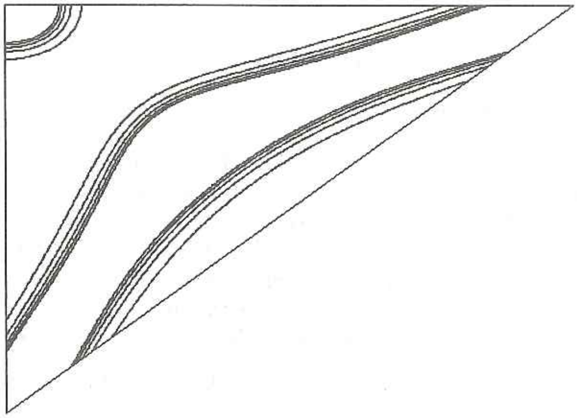

Figure 2.B.

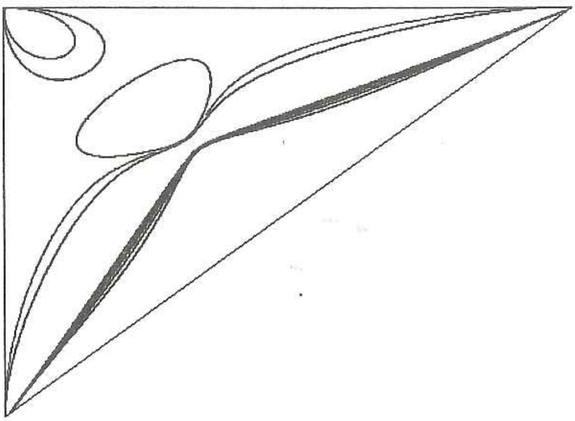

Figure 2.C.

References

[1] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design.*, Academic press, 1988.

[2] Lj. Kocić, D. Milošević, *On level sets of Bernstein-Bézier operators.*, Zbornik radova Filozofskog fakulteta u Nišu, Serija Matematika **6** (1992), 19–25.

[3] Lj. Kocić, D. Milošević, *Numerical Characteristics of Algorithm for Implicit Curve tracing*, Facta Univ. Ser. Mathematics and Informatics **8** (1993), 97–109.

[4] D. Milošević, Lj. Kocić, *Comparison of some algorithms for implicit function graph tracing.*, IX Conference on applied mathematics, Budva, 30 May – 1 Jun 1994, (D. Herceg, Lj. Cvetković, eds.), Institut of Mathematics, Novi Sad, 1995, pp. 65–70.

Department of Mathematics, Faculty of Electronic Engineering, P.O.Box 73, 18000 Niš