

# FITTING A CONIC A-SPLINE TO CONTOUR IMAGE DATA

V. Hernández Mederos<sup>1</sup>, D. Martínez Morera<sup>2</sup> y J. Estrada Sarlabous<sup>3</sup>  
Instituto de Cibernética, Matemática y Física, ICIMAF, La Habana, Cuba

## ABSTRACT

In this paper an algorithm for constructing contours from image data by means of a  $G^1$  conic A-spline is presented. A conic A-spline is a piecewise smooth chain of connected real quadratic algebraic curves meeting  $G^1$  at the junction points. Once the contour of the image data has been extracted, the algorithm computes the breakpoints of the conic A-spline, i.e the junction points for the conic curves make up the curve. Inflection points are also added to the set of junction points of the A-spline. Tangent lines at the junction points are computed using a weighted least square linear fit instead of the classical divided difference techniques. The conic A-spline interpolates the junction points along with the tangent directions and least-squares approximates the given data between junction points. We discuss and compare our experience with the approaches reported in the recent literature. Additionally, we propose some improvements.

**Key words:** fitting data, A-spline

## RESUMEN

En este trabajo se presenta un algoritmo para aproximar los contornos de imágenes digitalizadas usando un A-spline cónico  $G^1$ -continuo. Un A-spline cónico es una cadena de curvas algebraicas cuadráticas, reales y conexas, que se pegan de forma  $G^1$  en los puntos de unión. Una vez que se han extraído los puntos del contorno de la imagen, se calcula el conjunto de los puntos de unión del A-spline, al cual se agregan los puntos de inflexión. En lugar de utilizar las técnicas clásicas de diferencias divididas, las rectas tangentes en los puntos de unión se calculan usando un ajuste lineal mínimo cuadrático pesado. El A-spline interpola los puntos de unión y las direcciones tangentes prefijadas en estos y aproxima en el sentido mínimo cuadrado los datos contenidos entre dos puntos de unión consecutivos. Se discuten y comparan nuestras experiencias con los enfoques reportados en la literatura reciente y adicionalmente se proponen mejoras.

**Palabras clave:** datos dignos, A-spline

MSC: 65Y25, 51N35

## 1. INTRODUCTION

The problem of approximating a contour has been extensively treated in the literature Bajaj-Xu (1994), Bajaj-Xu (1999), Bookstein (1979), Gander *et al.* (1994), Odensaya *et al.* (1993), Pavlidis (1983), Ray-Ray (1994), Sampson (1982), Spaeth (1996), Shumaker (1990). The most common solutions use parametric splines Gander *et al.* (1994), Odensaya *et al.* (1993), Spaeth (1996), Lozover-Preiss (1983) which requires the introduction of an artificial parametrization of data. Unfortunately, the accuracy of the fitting process strongly depends on the selected parametrization.

In recent years new solutions to the problem has been proposed using implicitly defined algebraic splines (A-splines) Bajaj-Xu (1999), Bajaj-Xu (1994), Pavlidis (1983), Taubin (1991). Compared to parametric splines, more local control is obtained from A-splines, since after imposing the continuity conditions, the remaining free parameters of the A-spline only depend on the data to be fitted for each section of the A-spline curve. Moreover, in order to compute the coefficients of the parametric spline fitting a set of data points, we have often to solve a linear system of equations which involves all the coefficients of the spline. In this case, if some data vary, more than a section of the parametric spline should be computed again.

In this paper we expect to show, that several important steps of the procedure to construct an A-spline curve fitting the contour of a digital image may be improved. In section 2 we give some short preliminaries. Section 3 is devoted to conic A-splines. There, we describe how to select the break points of the A-spline, how to detect

**E-mail:** <sup>1</sup>vicky@cidet.icmf.inf.cu  
<sup>2</sup>dimas@cidet.icmf.inf.cu  
<sup>3</sup>matdis@cidet.icmf.inf.cu

the position of the inflection points and finally how to compute the tangent direction at each breakpoint. Therefore, after section 3 we know how to construct the control polygon of the A-spline curve. In section 4, we compute each segment of the A-spline fitting the data inside the corresponding control triangle. We study the problem of selecting a good approximation of the Euclidean distance from a point to a conic, in order to obtain a precise fitting of the data. Finally, in section 5 we present the results of applying the proposed algorithm to fit the contours of several digital images that represent cross sections of a human head.

## 2. PRELIMINARIES

The main purpose of this paper is to construct a smooth curve approximating the contour of a digital image. It is well known González-Woods (1992) that the term image refers to a two-dimensional light-intensity function  $I(x, y)$ , where the value of  $I(x, y)$  at spatial coordinates  $(x, y)$  is the intensity (brightness) of the image at that point. If the continuous image  $I(x, y)$  is approximated by equally spaced samples in the form of an  $N \times M$  matrix  $\tilde{I}(i; k) = I(X_i; Y_k)$ ,  $i = 1, \dots, N$ ,  $k = 1; \dots; M$  we obtain what is commonly called digital image. Each element of the matrix  $\tilde{I}$  is referred to as pixel. Commonly, the values of  $N$ ;  $M$  and the number of gray levels are integers powers of two.

The original data of our problem represents a digital image, i.e we have a matrix  $\tilde{I}$  where the value of  $\tilde{I}$  at a pixel is the corresponding grey level. This image is converted into a binary image by thresholding. Several classical techniques (segmentation, conditional dilatation and erosion) of image processing González-Woods (1992), Sahoo *et al.* (1988), Dougherty (1992) are used to improve the quality of the image in order to obtain a set of points describing its boundary.

The set of points  $q_j = (x_j, y_j)$ ,  $j = 1, \dots, m$  on the boundary of the image, where  $x_j = X_i$  and  $y_j = Y_k$  for some  $i$  and  $k$  is computed using an edge detection method. The data points  $q_j$ ,  $j = 1, \dots, m$  obtained by this method are sorted in the direction of motion of the boundary curve.

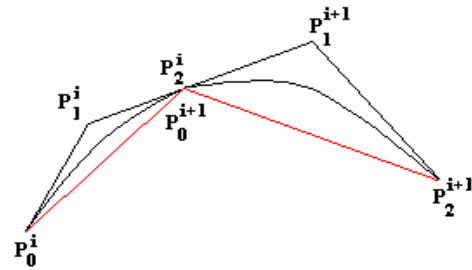
Once we have computed the data points representing the boundary of a digital image, the main steps of the algorithm to construct an A-spline curve approximating them are the following:

1. Determine the breakpoints of the A-spline, i.e. the points where two consecutive sections of the piecewise curve are joined.
2. Compute the tangent directions at each breakpoint. Determine the control polygon of the A-spline curve.
3. Fit the data inside each triangle of the control polygon by means of a conic curve.
4. Display the A-spline curve.

## 3. CONIC A-SPLINES

**Definition 1** (see Figure 1) Given a set of  $n$  triangles  $T_i$  with vertices  $P_0^i, P_1^i, P_2^i, i = 1, \dots, n$  satisfying  $P_0^{i+1} = P_2^i, i = 1, \dots, n-1$ , a conic A-spline  $S$  is a piecewise algebraic curve that satisfies the following conditions:

- i) Inside of each triangle  $T_i$ ;  $S$  is a conic segment  $S_i$ , i.e  $S|_{T_i} = S_i$  where  $S_i = \{(x, y) \in T_i; f_i(x, y) = 0\}$  and  $f_i$  is a polynomial of degree 2.
- ii)  $S_i$  interpolates the vertices  $P_0^i$  and  $P_2^i$  of the  $i$ -th triangle.
- iii) The tangent lines to  $S_i$  at  $P_0^i$  and  $P_2^i$  are the sides  $P_0^i P_1^i$  and  $P_1^i P_2^i$  of the triangle  $T_i$  respectively.



**Figure 1:** Two sections of a conic A-spline.

The breakpoints  $P_0^i, i = 2, \dots, n$  are also called knots and the sequence of triangles  $T_i, i = 1, \dots, n$  control polygon of the A-spline. If the tangents to the A-spline are continuous at the knots, we say that the A-spline

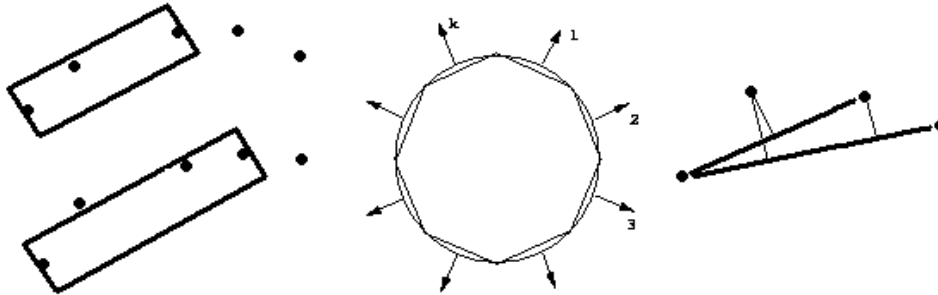
is  $G^1$ -continuous curve. According to the previous definition, a conic A-spline  $S$  is  $G^1$ -continuous if the points  $P_1^i, P_2^i, P_2^{i+1}, i = 1, \dots, n - 1$  are collinear. If  $P_2^n = P_0^1$ , then the A-spline  $S$  is a closed curve.

After the previous definition, the first step to compute a conic  $G^1$  A-spline, fitting the contour of a digital image, is to define the control polygon, i.e. a sequence of  $n$  triangles  $T_i$  with vertices  $P_0^i, P_1^i, P_2^i$  meeting at the vertices, containing the data and such that the points  $P_1^i, P_2^i, P_1^{i+1}, i = 1, \dots, n - 1$  are collinear. The breakpoints  $P_0^i = P_2^{i-1}, i = 2, \dots, n - 1$  of the A-spline are conveniently chosen from the set of contour data points. Then, we must determine the tangent line at each breakpoint and finally we compute the vertices  $P_1^i$  of each triangle  $T_i$  as the intersection point of the tangent line at  $P_0^i$  and  $P_2^i$  respectively.

#### 4. COMPUTATION OF JUNCTION POINTS

Several techniques have been used in the past to select the junction or break points of a spline (defined in parametric or implicit form). In this section we mention some of them and explain why some of them could not be successfully used to obtain the breakpoints of the A-spline curve.

In Bajaj-Xu (1994) the authors propose a curvature adaptive scheme to choose the breakpoints. They consider a regular polygon of  $k$  sides inscribed in the unit circle and numbers the normal directions to the polygon boundary with integers from 1 to  $k$ : Then, a line segment of the contour boundary is numbered with the integer  $i$  if it has the largest dot product of its normal with the  $i$ -th normal of the regular polygon. In this way, the  $k$  normal directions on the circle subdivides the data contour into groups, where each group consists of a connected sequence of line segments having the same assigned number. The endpoints of the groups are the breakpoints. It is clear that the sequence of breakpoints in this method doesn't depend on which is the initial point analyzed. Nevertheless, the method fails for data points representing the boundary of a digital image, where frequently the data are in form of a stair. In this case, the groups of points having the same assigned number have few points (frequently only one), and therefore the final sequence of breakpoints contains too many points .



**Figure 2:** Selection of the breakpoints.

Left: using a mask.

Center: discretizing normal directions.

Right: non-parametric sequential method.

The method considered in Odensaya **et al.** (1993) is an enhanced version of a similar method proposed by Lozover and Preiss Lozover-Preiss (1983). They employ a tolerance window or "mask" to identify the junction points. Initially, the mask is placed so that its centerline agrees with the segment passing through the first and the third points in the set. Then, the end of the mask is extended one point at a time, until one intermediate data point between the mask endpoints falls outside the mask. Backtracking one step, the corresponding mask endpoints are marked as junction points. From our point of view, this method has one disadvantage: the final sequence of junctions points depends on the first point where the mask is placed. Odesanya **et al.** propose to investigate each data point as a potential starting point for the masking process and to construct the sequence of junction points and the corresponding (parametric) spline interpolating at junction points, in order to determine the "optimal" spline approximation. This method can be very expensive if the amount of data points is large. With respect to the "optimal" mask width there are not any comments. Obviously, as the mask width grows the number of junction points decrease.

The method proposed in Ray-Ray (1994) to compress digital image contour can be also used to select the junction points. This algorithm constructs a polygonal approximation of digital curves looking for the longest possible line segments with the minimum possible errors. Starting from an arbitrary point  $P_i$  on the contour of the digital image, the algorithm constructs the line  $r_{ij}$  passing through  $P_i$  and  $P_j$  with  $j > i + 1$  and increases  $j$  while the function  $F_j := L_j - E_j$  grows, where  $L_j$  is the length of the segment  $P_i P_j$  and  $E_j$  is the sum of the squares of the distances from the intermediate points to the line  $r_{ij}$ . Unlike other methods, this algorithm does not require to specify a priori the allowable error. After our experience, this apparent advantage may lead to get very coarse approximations, if we deal with contour images possessing some regions where the details are very important while the remaining are very smooth.

According to the previous analysis, in order to determine the position of the breakpoints it is convenient to use a method which allows to approximate the image contour by a polygonal, introducing some control of the allowable error. In this sense, we propose to employ the method Odensaya *et al.* (1993) which gives good results, if we select a mask width capable of ignoring the noise of data, but preserving the important details of the contour. For instance, in the reconstruction of the human head (see section 5) the breakpoints of the A-splines approximating the contours of all cross sections were computed using a mask width equal to 5 pixels, except for the corresponding to the ears, where the mask width was selected equal to 3 pixels. On the other hand, after our experiences, it is expensive and worthless (from the point of view of the quality of the fitting) to compute the "optimal" spline over different breakpoint sequences, if we use a good approximation of the Euclidean distance from a point to a conic, to solve the least squares problem (5).

Since conics don't have inflection points, any inflection point of the A-spline must be a breakpoint. Therefore, we need a method for automatically determining the position of inflection points from the data describing the contour of the image. The method should be flexible enough to disregard noise in the data points and computationally efficient, since in order to locate the inflection points, we have to check a large number of data points.

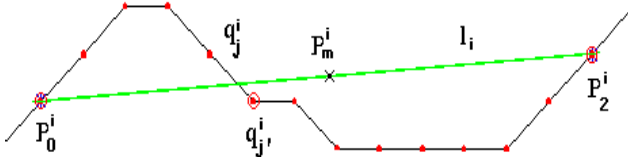
Let's suppose that  $P_0^i$  and  $P_2^i$  are two consecutive vertices of the polygonal that approximates the contour, then we introduce a new vertex between them, if the amount of data that is on the same side of the line  $l_i: l_i(x, y) = 0$  passing through  $P_0^i$  and  $P_2^i$  is not greater than 80 percent of the data between both points. More precisely, let  $q_j^i, j = 1, \dots, n_i$  be the points between  $P_0^i$  and  $P_2^i$  and denote by

$$\Gamma_i = \{q_j^i \mid l_i(q_j^i)l_i(q_{j+1}^i) < 0 \text{ or } l_i(q_j^i)l_i(q_{j-1}^i) < 0\}$$

the set of points where the sign of the evaluation of  $l_i$  changes. Then, the new breakpoint (corresponding to an inflection point) is  $q_{j'}^i$  such that

$$\|q_{j'}^i - P_m^i\|_2 = \min \{\|q_j^i - P_m^i\|_2, q_j^i \in \Gamma_i\}$$

where  $P_m^i$  is the middle point of the segment  $P_0^i P_2^i$  (see Figure 3).



**Figure 3.** Locating the position of an inflection point. The point  $q_{j'}^i$  is the new

knot  
(inflection point).

Summarizing, we choose the breakpoints of the A-spline  $S$  as the union of the set of junction points computed with the algorithm proposed by Odensaya *et al.* (1993) with the set of inflection points.

## 5. TANGENT LINE AT JUNCTION POINTS

In order to compute a  $G^1$  A-spline it is necessary to define a control polygon, such as it was introduced in Definition 1. After computing the breakpoints, which are vertices  $P_0^i$  and  $P_2^i$  of the control polygon, we must define the corresponding tangent lines at any of the selected breakpoints. The intersection point of the tangent lines at two consecutive breakpoints is the vertex  $P_1^i$  of the control triangle  $T_i$  for the  $i$ -th section of the A-spline.

For the computation of the tangent lines, we found several methods in the literature. Bajaj and Xu, Bajaj-Xu (1994), compute a polynomial interpolating a breakpoint and its closest neighbors, in the set of data points, to the right and to the left and assign as tangent direction at the breakpoint, the tangent direction corresponding to this polynomial approximation (parabola). We also tested out the fourth degree polynomial interpolating a breakpoint and its two closest neighbors to the right and to the left. In all cases, since contour data arising from a digital image are very noisy, the approximations to the tangent direction based on few neighbors of a breakpoint provide unaccurate results (see Figure 4). On the other hand, it is well known that numerical differentiation based on interpolating polynomials is basically an unstable process, and good accuracy can not be expected, even when the original data are known to be accurate (see Conte-de Boor (1980)).

In order to overcome these problems, we propose the following procedure. Recall that  $q_j^i$ ,  $j = 1, \dots, n_i$  are the contour points inside of the  $i$ -th triangle of the control polygon. Let  $\Lambda_i = \{q_{[n_{i-1}, 2]+1}^{i-1}, \dots, q_{n_{i-1}}^{i-1} = q_1^i, \dots, q_{[n_i, 2]}^i\}$  be the set of "neighbors" of  $P_0^i$ . In order to simplify the notation, denote by  $p_j^i = (x_j^i, y_j^i)$  with  $j = 1, \dots, m_i$  the set of points in  $\Lambda_i$ . Now, consider that  $p_j^i$  is associated to the parameter value  $t_j^i$ , representing the cord length parametrization, where

$$t_1^i = 0$$

$$t_{j+1}^i = t_j^i + \|p_{j+1}^i - p_j^i\|_2, \quad j = 1, \dots, m_i - 1$$

Note that  $P_0^i = p_{j^*}^i = (x_{j^*}^i, -y_{j^*}^i)$ , where  $j^* = \left\lfloor \frac{n_{i-1} + 1}{2} \right\rfloor$ . We are interested in computing the parametric line  $r(t) = (x(t), y(t))$ ,

$$x(t) = a_1(t - t_{j^*}^i) + x_{j^*}^i$$

$$y(t) = a_2(t - t_{j^*}^i) + y_{j^*}^i$$

passing through the point  $P_0^i$  and fitting in the least squares sense the points in the set  $\Lambda_i$ . More precisely  $a_1$  and  $a_2$  are free parameters that we determine solving respectively the linear least square problems,

$$\min_{a_1} \sum_{j=1}^{m_i} w_j^2 (w_{j^*}^i - x(t_j^i))^2 \quad (1)$$

$$\min_{a_2} \sum_{j=1}^{m_i} w_j^2 (y_{j^*}^i - y(t_j^i))^2 \quad (2)$$

where the weights  $w_j$  are given by  $w_j^2 = 1/\|p_j^i - p_{j^*}^i\|_2$ ,  $j = 1, \dots, m_i$ ,  $j \neq j^*$ ,  $w_{j^*} = 1$ . Observe that the selected weights are greater for the points closer to the breakpoint. Therefore, the fitting line not only passes through the breakpoint, but also fits better the points closest to the breakpoint. It is easy to prove that the solution of the problems (1) and (2) are given respectively by,

$$a_1^* = \sum_{j=1}^{m_i} w_j^2 (x_j^i - x_{j^*}^i) (t_j^i - t_{j^*}^i) / \sum_{j=1}^{m_i} w_j^2 (t_j^i - t_{j^*}^i)^2 \quad (3)$$

$$a_2^* = \sum_{j=1}^{m_i} w_j^2 (y_j^i - y_{j^*}^i) (t_j^i - t_{j^*}^i) / \sum_{j=1}^{m_i} w_j^2 (t_j^i - t_{j^*}^i)^2 \quad (4)$$

Therefore, the vector  $[a_1^*, a_2^*]$  is selected as tangent vector at the breakpoint  $P_0^i$ .

Figure 4 shows the tangent line obtained using different techniques. Observe that the approximations based on interpolating polynomials happen to be unsatisfactory for data arising from digital images. On the other hand, the fitting line passing through the breakpoint provides a nice approximation to the tangent line.

Finally, if  $P_0^i$  is an inflection point, then the previous tangent direction must be corrected, since the tangent line at an inflection point should cross the polygon joining the breakpoints in order to obtain a feasible control polygon. The new tangent direction is given by the line passing through  $P_0^i$  and a data point  $q_j$  with  $\hat{j} \neq j^*$  located between  $P_0^{i-1}$  and  $P_0^{i+1}$ . The point  $q_j$  is selected maximizing the angle between the segments  $P_0^i q_j^{i-1}$  and  $P_0^{i-1} P_0^i$  or between  $P_0^i q_j^i$  and  $P_0^{i+1} P_0^i$  (see Figure 5).

## 6. FITTING WITH CONIC A-SPLINES

### 6.1. Distance from a point to a conic

Several methods are available in the literature for fitting data by conics Bookstein (1979), Bajaj-Xu (1999), Sampson (1982), Gander *et al.* (1994), Pavlidis (1983). Given a set of  $n$  data points  $q_j$  in the plane, the classical fitting approach consists on determining a conic  $C$  that minimizes the mean square distance,

$$\frac{1}{n} \sum_{j=1}^n d^2(q_j, C) \quad (5), \text{ where } d(q_j, C) \text{ is the Euclidean distance from } q_j \text{ to } C.$$

Define the implicit equation of the conic  $C$  as

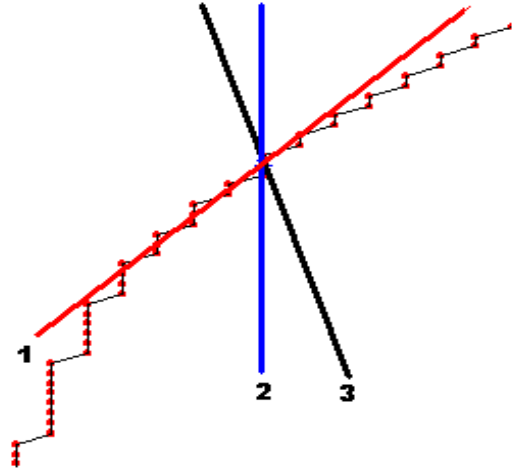
$$f(x; y) = 0 \quad (6), \text{ where } f(x; y) \text{ is a polynomial of degree 2.}$$

Since it is not possible to give a closed expression for the Euclidean distance from a point to a conic, the most common approach to solve (5) is to use the approximation,

$$d(q_j, C) = |f(q_j)| \quad (7)$$

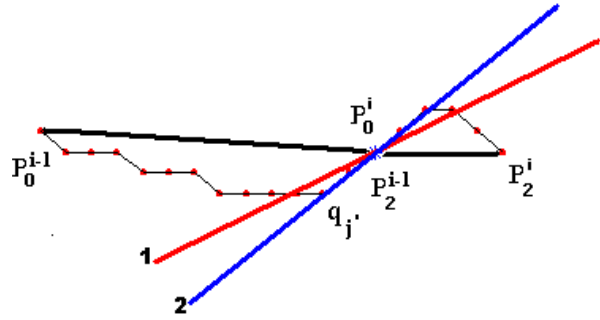
Hence the problem (5) gives rise to the problem of minimizing,

$$\frac{1}{n} \sum_{j=1}^n f^2(q_j) \quad (8)$$



**Figure 4.** Selecting the tangent vector using:

1. Fitting line,
2. Interpolation "parabola",
3. Fourth degree interpolation polynomial.



**Figure 5.** Correcting the tangent direction at an inflection point:

1. Fitting line,
2. Corrected tangent line.

This formulation of the problem is very popular because (8) is a linear least squares problem for the coefficients of the conic. Nevertheless, since the implicit equation (5) is homogeneous, a constraint on the coefficients of the conic has to be imposed in order to obtain a nontrivial solution.

Some authors have considered linear constraint changing the minimization of (8) into a linear regression problem, whereas others have proposed quadratic restrictions converting (8) into an eigenvalue problem. Moreover, depending on the constraints, the fitting methods may possess special properties, for instance Taubin's generalized eigenvector fit Taubin (1991) and the Bookstein algorithm Bookstein (1979) are invariant under affine transformations. Unfortunately, any constraint that we impose introduces bias into the selection of the fitting conic and one should be aware that the bias may eliminate some candidates for best fitting conic.

The approximation of the Euclidean distance (7), so called *algebraic distance*, happens to be a good approximation only if the data are very close to the conic. Therefore, some authors to use better approximations given by a closed formula Taubin (1994) or an iterative method Ponce **et al.** (1992), Hernández **et al.** (2001) but in that case the corresponding least squares problem is no longer linear. The more precise the approximations or the iterative method are, the more expensive is the computation of the best fit, hence a good compromise must be found. In this paper, we use the approximation of the Euclidean distance given by G. Taubin in Taubin (1994),

$$d(q_j, C) = \frac{|f(q_j)|}{\|\nabla f(q_j)\|} \quad (9)$$

and impose constraints that bias the solution against the selection of undesirable degenerate conics (see section 6.2).

Finally, let's mention the parametric approach. Since every conic may be represented in parametric form, the problem (5) can be solved introducing one parameter value for each data point. Thus, it becomes nonlinear and its dimension grows with the number of data points. Some previous works using this approach are Spaeth (1996), Gander **et al.** (1994).

## 6.2. Fitting a conic inside of each triangle

It is well known from the literature Farin (1992), that inside each triangle  $T_i$  it is possible to construct a family of conic sections interpolating the vertices and the prescribed tangent directions. This family depends on a single parameter that we call  $w_1^i$ , which may be used to fit the data in the interior of the corresponding triangle. In what follows, we concentrate ourselves in the selection of the value  $w_1^i$ , corresponding to the conic section  $C_i$  that best fits the data  $q_j^i = (x_j^i, y_j^i)$ ,  $j = 1, \dots, n$  inside a fixed triangle.

In order to handle any triangle in a similar way, we write the conic section in barycentric coordinates with respect to the vertices of the triangle,

$$C_i : g(u, v) = a_i v^2 - b_i u(1 - u - v) = 0$$

where

$$u = \frac{\begin{vmatrix} x & P_{1x}^i & P_{2x}^i \\ y & P_{1y}^i & P_{2y}^i \\ 1 & 1 & 1 \end{vmatrix}}{A}, \quad v = \frac{\begin{vmatrix} P_{0x}^i & x & P_{2x}^i \\ P_{0y}^i & y & P_{2y}^i \\ 1 & 1 & 1 \end{vmatrix}}{A}$$

$P_0^i = (P_{0x}^i, P_{0y}^i)$ ,  $P_1^i = (P_{1x}^i, P_{1y}^i)$ ,  $P_2^i = (P_{2x}^i, P_{2y}^i)$  and  $A$  is the area of the corresponding triangle.

We impose the constraints  $a_i \neq 0$  and  $b_i \neq 0$  which bias the selection of the fitting conic against the degenerate conic sections  $C_i : g_i(u, v) = u(1 - u - v) = 0$  and  $C_i : g_i(u, v) = v^2 = 0$ , respectively. Under these restrictions, the implicit equation of the conic section may be rewritten as,

$$C_i : g_i(u, v) = v^2 - (w_1^i)^2 u(1 - u - v) = 0$$

with  $w_1^i \neq 0$ . Therefore, if we use the algebraic distance as approximation of the Euclidean distance, the corresponding least squares problem is given by,

$$\min_{w_1^i} G(w_1^i) := \frac{1}{n_i} \sum_{j=1}^{n_i} g_i^2(u_j^i, v_j^i) \quad (10)$$

where  $(u_j^i, v_j^i)$  are the barycentric coordinates of  $q_j^i$  with respect to the vertices of the triangle. This is a linear problem, whose solution is explicitly given by,

$$(w_1^i)^2 = \frac{\sum_{j=1}^{n_i} (v_j^i)^2 u_j^i (1 - u_j^i - v_j^i)}{4 \sum_{j=1}^{n_i} (u_j^i)^2 (1 - u_j^i - v_j^i)} \quad (11)$$

As we previously mentioned, if the points  $q_j^i = (x_j^i, y_j^i)$  are not very close to a conic, the optimal parameter (11) gives rise to a conic far from the conic that best fits the data in the Euclidean distance. Therefore we recommend the approximation of the Euclidean distance (9). If  $f_i(x, y) = 0$  is the implicit equation of the curve  $C_i$  then the corresponding least squares problem may be written as,

$$\min_{w_1^i} G(w_1^i) := \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{f_i^2(x_j^i, y_j^i)}{\|\nabla f_i(x_j^i, y_j^i)\|^2} \quad (12)$$

Since  $f_i(x_j^i, y_j^i) = g_i(u_j^i, v_j^i)$  and

$$\nabla f_i(x_j^i, y_j^i) = \left( \frac{\partial g_i}{\partial u} \frac{\partial u}{\partial x} + \frac{\partial g_i}{\partial v} \frac{\partial v}{\partial x}, \frac{\partial g_i}{\partial u} \frac{\partial u}{\partial y} + \frac{\partial g_i}{\partial v} \frac{\partial v}{\partial y} \right) (u_j^i, v_j^i)$$

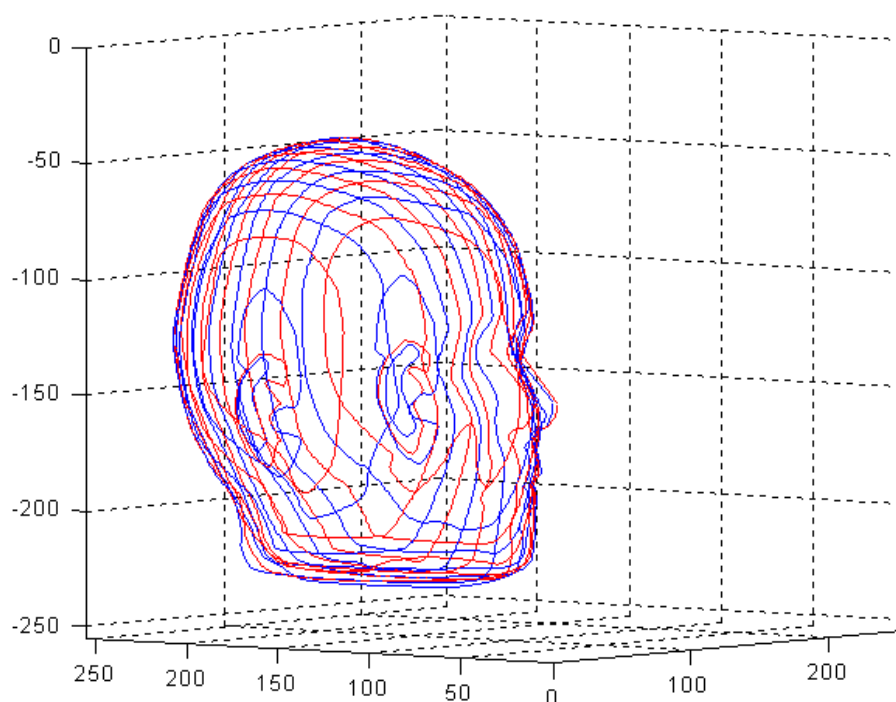
the function  $G(w_1^i)$  may be written in terms of the barycentric coordinates  $(u_j^i, v_j^i)$  of the data points  $q_j^i$  and the parameter  $w_1^i$ . The corresponding least squares problem is nonlinear in  $w_1^i$ , with the restriction  $w_1^i > 0$ . Since (9) is a better approximation (of the Euclidean distance from a point to a conic) than the algebraic distance, the corresponding fitting conic, approximates better the contour image.

## 7. NUMERICAL EXAMPLES

The algorithm proposed in this paper was successfully applied to the approximation of the contours of magnetic resonance images (MRI) of a human head (downloaded from Rendering test data set, North Carolina University, ftp.cs.unc.edu). Figure 6 shows, in the same plot, the conic A-splines which approximate 25 contours. Each contour was obtained from a previous processing of a digital image that corresponds to a cross section of the human head. The results were obtained from a MATLAB (version 5.3) program that constructs and displays the A-spline curve approximating the contour data. In order to display each conic section we use the classical Bernstein-Bezier parametrization of rational curves, Farin (1992). As we previously pointed out, we used a mask width equal to 5 pixels for all contours, except for the corresponding to the ears, where the mask width was selected equal to 3 pixels. In the computation of the A-splines fitting



the data inside each triangle, we use the approximation (9) of the Euclidean distance from a point to a conic and solved the nonlinear least square problem (12).



**Figure 6.** Approximation of the cross sections of a human head by conic A-splines.

## ACKNOWLEDGEMENTS

The results contained in this paper were partially obtained under the TWAS Research Grant 98-195 RG/MATHS/LA. We would like to express our gratitude to R. Patterson for his long support. We also would like to the members of the Group of Image Processing from ICIMAF, for their advice on the techniques of Digital Image Processing and the preprocessing of the MRI of a human head.

## REFERENCES

- BAJAJ, C. and G. XU (1999): "A-Splines, Local Interpolation and Approximation using  $G^k$ -Continuous Piecewise Real Algebraic Curves", **Computer Aided Geometric Design**, 16, 557-578.
- \_\_\_\_\_ (1994): "Data fitting with cubic A-splines", Proceedings: **Computer Graphics International**, CGI94, Melbourne, Australia.
- BOOKSTEIN, F.L. (1979): "Fitting conic sections to scattered data", **Comput. Vision, Graphics and Image Processing**, 9, 56-71.
- CONTE, S.D. and C. de BOOR (1980): **Elementary Numerical Analysis**, Mc Graw-Hill, New York.
- DOUGHETY, E. (1992): **An Introduction to Morphological Image Processing**, SPI-The International Society for Optical Engineering, Washington.
- FARIN, G. (1992): **Curves and Surfaces for Computer Aided Geometric Design**, Academic Press, New York.
- GANDER, W.; G.H. GOLUB and R. STREBEL (1994): "Least squares fitting of circles and ellipses", **BIT** 34, 558-578.
- GONZALEZ, R.C. and R.E. WOODS (1992): **Digital Image Processing**, Addison-Wesley, Massachusetts.

- HERNANDEZ MEDEROS, V.; J. ESTRADA SARLABOUS and P. BARRERA SANCHEZ (2201): "A new algorithm to compute the Euclidean distance from a point to a conic" (accepted for publication, **Investigación de Operaciones**.
- LOZOVER, O. and K. PREISS (1983): "Automatic construction of a cubic B-spline representation for a general curve", **Comput. and Graphics**, 7(2), 149-153.
- ODESANYA, O.S.; W.N. WAGGENSPACK and D.E. THOMPSON (1993): "Construction of biological surface models from cross sections", **IEEE Transactions on Biomedical Engineering**, 40(4), 329-334.
- PAVLIDIS, T. (1983): "Curve fitting with conic splines", **ACM Trans. on Graphics**, 2(1), 1-31.
- PONCE, A.; J. HOOGS and D. KRIEGMAN (1992): "On using CAD models to compute the pose of curved 3D objects", **Comp. Vision Graphics and Image Processing** 55(2), 184-197.
- RAY, B. KR. and K.S. RAY (1994): "A non-parametric sequential method for polygonal approximation of digital curves", **Pattern Recognition Letters** 15, 161-167.
- SAHOO, P.K.; S. SOLTANI; A.K. WONG and Y.C. CHEN (1988): "A survey of thresholding techniques", **Computer Vision, Graphics, and Image Processing**, 41, 233-260.
- SAMPSON, P.D. (1982): "Fitting conic sections to very scattered data: an iterative refinement of Bookstein algorithm", **Comp. Vision Graphics and Image Processing**, 18, 97-108.
- SHUMAKER, L.L. (1990): "Reconstructing 3D objects from cross sections", **Computation of Curves and Surfaces**, Kluwer Academic Publishers, Dordrecht, 275-309.
- SPAETH, H. (1996): "Orthogonal squared distance fitting with parabolas", **Numerical Methods and Error Bounds, Mathematical Research** 89, Akademie Verlag, 261-269.
- TAUBIN, G. (1991): "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation", **IEEE Trans. on Pattern Anal. Machine Intell.**, 13, 1115-1138.
- TAUBIN, G. (1994): "Distance approximations for rasterizing implicit curves", **ACM Trans. on Graphics**, 13, 3-42.