

# A Filtered B-spline model of Scanned Digital Images

Faramarz Famil Samavati\*      Nezam Mahdavi-Amiri†

January 16, 1998

## Abstract

We present an approach for modeling and filtering digitally scanned images. The digital contour of an image is segmented to identify the linear segments, the nonlinear segments and *critical corners*. The nonlinear segments are modeled by B-splines. To remove the contour noise, we propose a weighted least squares model to account for both the fitness of the splines as well as their approximate curvatures. The solutions of the least squares models provide the control vertices of the splines. We show the effectiveness of our approach with several representations constructed from various scanned images.

## 1 Introduction

Compact representations for digital images have practical applications. Geometric models can provide such representations. These models can be used effectively in Computer Aided Design (CAD), where the efficient construction and manipulation of geometric models are quite important. Recently, Several approaches have been proposed for converting digital images to geometric models. In [3], [11], a geometric model of an image is constructed by lines and arcs. B-spline Models have been proposed in [6], [7], [8]. We propose an alternative approach by first detecting the linear segments, the nonlinear segments and the *critical corners* of the image contour, and then modeling only the nonlinear segments by B-splines. This offers several advantages. Since the identification of the spline model constitutes most of the work in this process, we reduce the cost of the process by detecting and separating the linear segments. Moreover, separate modeling of each segment should allow for a more compact as well as a more desirable representation of the image.

Several features need to be emphasized. We present a weighted least squares model to account for both the fitness of the spline to a nonlinear segment and its approximate

---

\*Department of Mathematical Sciences, Sharif University of Technology, Azadi Avenue, Tehran, Iran. Present Address: Computer Graphics Laboratory, Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada.

†Department of Mathematical Sciences, Sharif University of Technology, Azadi Avenue, Tehran, Iran. Present Address: Department of Computer Science, York University, Toronto, Ontario, Canada.

curvature. The solution of the least squares problem yields the control vertices of the segment. By varying the weights in the least squares model, one can control the shape of the image. This also provides a useful capability for the removal of contour noise, commonly present in digital images. The effectiveness and utility of our approach are illustrated by various examples.

## 1.1 An Outline

Section 2 presents most of the preliminary notations for B-spline representations. It also outlines some desirable features of these models. In section 3, we discuss how to detect the linear and nonlinear segments along with the critical corners of the image contour. We explain these feature extractions in some detail making them convenient for an implementation. We devote section 4 to the presentation of the weighted least squares model and its solution. In section 5, We give an outline of an algorithm for computing the control vertices of the splines. We also discuss certain computational issues and alternatives. We end with several illustrative examples to show the effectiveness of the least squares model.

# 2 Geometric Models

There are various approaches to geometric modeling of data. We consider certain mathematical models as approximations to data. In section 2.1, we point out interpolation and introduce some necessary notations. We discuss the general B-spline approximation in section 2.2 and emphasize its attractive features for modeling.

## 2.1 Interpolating Models

Let  $P_0, P_1, \dots, P_N$  be points in  $\mathbb{R}^n$  and denote  $P_i = (p_{i1}, \dots, p_{in}), i = 0, 1, \dots, N$ . A parametric interpolating polynomial,  $Q(t)$ , corresponding to the points  $P_i$  is defined as

$$Q(t) = (Q_1(t), Q_2(t), \dots, Q_n(t)), \tag{1}$$

where  $Q_j(t)$ , for each  $j$ , is an interpolating polynomial for the points  $(t_i, p_{ij}), i = 0, 1, \dots, N$ . The parameters  $t_i$  are a collection of specified real numbers. It is well known that, when the  $t_i$  are distinct ( $t_i \neq t_j$ , for  $i \neq j$ ), there exists a unique interpolating polynomial of degree less than or equal to  $N$ . There are a number of methods for computing such polynomial (see [10]). It is also known that interpolation may not be suitable for graphical purposes, the main shortcomings being the lack of sufficient control over the shape of the curve and on the degree of the polynomial. Although polynomial interpolation does have certain applications, but, in general, is not practical for geometric modeling and computer graphics (see [2]).

An alternative interpolation approach is based on the use of spline models. A spline of order  $k + 1$  (degree  $k$ ),  $S(x)$ , over the joints (or knots)  $t_i$  ( $t_0 < t_1 < \dots < t_N$ ) is a piecewise polynomial of degree  $k$  so that the interpolating pieces  $S_i(t)$  on  $[t_i, t_{i+1}]$ , for

$i = 0, 1, \dots, N - 1$ , are so smoothly joined that the spline is continuous on  $[t_0, t_N]$  up to and including its  $(k - 1)$ th derivative. That is,

$$S_i^{(l)}(t_{i+1}) = S_{i+1}^{(l)}(t_{i+1}), \quad l = 0, 1, \dots, k - 1 \quad (2)$$

$$i = 0, 1, \dots, N - 2.$$

The above conditions along with  $(k - 1)$  additional ones (usually imposed on the endpoints of the interval) result in a unique representation for  $S$ . For graphical purposes, splines are more desirable than general interpolating polynomials. In fact, under certain conditions, the cubic spline has a minimal curvature (in some sense). However, the lack of a suitable control over the shape of the curve diminishes the utility of interpolating splines in geometric modeling.

To introduce more flexibility, interpolating splines under *tension* have been proposed [1]. More flexible spline models do not interpolate the data points, however. Instead, the points are used as a control device for the shape of the spline. Next, we discuss these models in terms of their basic polynomials, the B-splines.

## 2.2 B-spline Models

Most of the notations used here are adopted from [2]. The B-spline polynomials,  $B_{j,k}(u)$ ,  $j = 0, 1, \dots, m$ , form a basis for the space of all splines of order  $k$  (degree  $k - 1$ ) over  $[u_{k-1}, u_{m+1}]$  with respect to the knots  $\{u_j\}_0^{m+k}$ . The B-splines may be defined recursively as follows:

$$B_{j,1} = \begin{cases} 1 & \text{if } u_j \leq u < u_{j+1} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$B_{j,r}(u) = \frac{u - u_j}{u_{j+r-1} - u_j} B_{j,r-1}(u) + \frac{u_{j+r} - u}{u_{j+r} - u_{j+1}} B_{j+1,r-1}(u) \quad (4)$$

$$r = 2, 3, \dots, k,$$

where any term with a zero denominator is replaced by zero.

It can be shown that:

- $B_{j,k}(u)$  is nonnegative and is positive only on its *support*, the interval  $(u_j, u_{j+k})$ .
- $\sum_{j=0}^m B_{j,k}(u) = 1$  for all  $u \in (u_{k-1}, u_{m+1})$ .

A B-spline curve is defined as a linear combinations of the B-splines. Assume  $V_j \in \mathbb{R}^n$ ,  $j \in J$ , for some index set  $J$ , are given. The B-spline curve of order  $k$  corresponding to the *control vertices*  $V_j$  is defined to be

$$Q(u) = \sum_{j \in J} V_j B_{j,k}(u). \quad (5)$$

Figure 1: A B-spline Curve and its Control Vertices.

Since  $B_{j,k}(u) \geq 0$  and  $\sum B_{j,k}(u) = 1$ , we observe that for each  $u$ , the value of the spline is a linear convex combinations of the control vertices. Figure 1 shows a B-spline curve associated with the corresponding control vertices. B-spline curves have several important properties making them practical in geometric modeling. These are listed below.

- **Convex Hull Containment**

The B-spline is contained in the convex hull<sup>1</sup> of its control vertices. In fact, a stronger property holds. If we let  $Q_j(u) = Q(u)$  for  $u \in [u_j, u_{j+1})$  then  $Q_j(u)$  is contained in the convex hull of the vertices  $\{V_{j-k+1}, \dots, V_j\}$ .

This property prevents the spline values from having undue fluctuations.

- **Affine Invariance**

If  $T$  is an affine transformation on  $\mathbb{R}^n$  then

$$Q_T(u) = \sum_j B_{j,k}(u)T(V_j) = T(Q(u)). \quad (6)$$

This not only preserves the shape of the spline but also ensures a fast access to the new spline under an affine transformation (of vertices) such as *translation*, *rotation*, or *reflection*.

- **Local Control**

The change of the position of a control vertex affects only the shape of the B-spline in the vicinity of the control vertex (see Figure 2).

This property is quite useful in an interactive environment where the effect of a change to vertex positions can be demonstrated by performing only the necessary local computations.

In addition to the above properties, a B-spline representation of an image contour will also possess the following qualities.

---

<sup>1</sup>The smallest convex set containing the control vertices.

Figure 2: The Effect of Local Change to Control Vertices.

- **Compact Representation**

Since a B-spline curve is completely identified by its control vertices, the B-spline representation of a contour will require much less storage than its digital representation.

- **Analytic Features**

There are efficient ways to evaluate the B-splines and their derivatives. This will be useful in solving our weighted least squares problems for finding the control vertices.

### 3 Contour Features Extraction

A digital image is usually stored compactly with the use of certain formats. We use the *gif* and *pcx* formats (see [9]) to store our scanned images. We then convert these formats to their matrix representations, and apply a noise removal procedure to remove the so called *image noise* (see [5]), the noise not belonging to the original image and possibly introduced in the scanning phase (see Figure 3). Next, using the matrix representation, we trace the contour<sup>2</sup> of the image to extract features such as *segmentation points*, linear and nonlinear segments. The segmentation points partition the contour into consecutive segments each of which may be handled individually. We explain the approach through an example given in Figure 4.

Let  $P = \{P_i = (x_i, y_i), i = 0, 1, \dots, N\}$  be a set of points (obtained from a digital image) representing an image contour (the contour of  $\Psi$  in Figure 4). It is obvious that these points do not constitute a continuous curve. A continuous representation,  $C$ , may be constructed by the line segments adjoining the consecutive points, and can serve as an approximation of the contour. The vector function  $C$  may be defined in terms of a parameter  $s$ , a scaled representation of its contour length. More precisely, with the definitions

---

<sup>2</sup>The boundary set of points having the same pixel value.

Figure 3: A Digital Image Before (Left) and After (Right) Noise Removal.

Figure 4: A Contour of  $\Psi$  (Bottom) and its Features Diagram (Top).

$$l_i = \| P_i - P_{i-1} \|_2 \quad i = 1, 2, \dots, N, \quad (7)$$

$$L = \sum_{i=1}^N l_i, \quad (8)$$

$$s_i = \frac{\sum_{j=1}^i l_j}{L} \quad i = 1, 2, \dots, N, \quad (9)$$

the function  $C$  may be defined as follows:

$$C(s) = \begin{cases} P_i & \text{if } s = s_i \\ tP_i + (1-t)P_{i+1} & \text{if } s = ts_i + (1-t)s_{i+1}, \text{ for } 0 < t < 1. \end{cases} \quad (10)$$

We may now think of  $C$ , a continuous representation on  $[0, 1]$ , as an approximation of the original contour. We define approximations to the left and right derivatives of the original contour at  $s_i$  (corresponding to the point  $P_i$ ) as the left and right derivatives of  $C$ , respectively. These derivatives would simply be the derivatives of the line segments emanating from  $P_i$ . So, we have

$$D_i^- = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \quad i = 1, 2, \dots, N, \quad (11)$$

$$D_i^+ = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad i = 0, 1, \dots, N - 1. \quad (12)$$

The  $D_i$  can serve as approximations to the slopes of the tangent lines to the original contour at the points  $P_i$ . With the use of these approximate derivatives, we define the segmentation points,  $\mathbf{T}$ , to be the set of points which identifies the separated segments of the image contour. We define  $\mathbf{T}$  to be composed of two subsets of points from  $P$  as follows:

$$\mathbf{T} = \mathbf{LN} \cup \mathbf{SC}, \quad (13)$$

where  $\mathbf{LN}$  denotes the set of points at which a linear segment is turned into a nonlinear one, and  $\mathbf{SC}$ , the set of *critical corners*, is a set of points at which a segment is turned into another with a *sharp angle*, determined by a given angle  $\alpha$ ,

$$\mathbf{SC} = \{i | i \in \{1, 2, \dots, N - 1\}, |D_i^+ - D_i^-| > \tan(\alpha)\}. \quad (14)$$

The following comments may be considered for an implementation of the segmentation procedure.

- A linear segment may be identified pointwise (for a sequence of points  $P_i$  so that  $|D_i^+ - D_i^-| < \varepsilon$ , given a small predetermined tolerance  $\varepsilon$ ). A nonlinear segment is also identified for a sequence of points with the inequality being reversed. Figure 4 shows the segments of  $\Psi$  through its *features diagram*, the set of values of the angles

corresponding to the slopes  $D_i^+$  and  $D_i^-$  ( $-\arctan$  of the slopes). We realize that the linear segments are shown by horizontal lines. In Figure 4, some segmentation points are identified on  $\Psi$  with 3 belonging to **LN** and the other numbered points belonging to **SC**.

- For smooth turns at points in **LN** where a nonlinear segment is adjoined, it is advised that a few points nearby the joint be included with the nonlinear segment. With this inclusion, the B-spline model of the points corresponding to the nonlinear segment should join the connecting line segment smoothly.
- The case where a linear segment may join another has not been discussed here, but its consideration is a minor implementational issue.
- Having identified the segmentation points, an algorithm can generate the contour segment by segment handling each segment individually.
- We note that color images can also be handled, simply by a separate segmentation for each color.

Each nonlinear segment needs a special treatment to be explained in the next section.

## 4 The Least Squares and B-splines

Each nonlinear segment is modeled by a B-spline. A weighted least squares problem is introduced whose solution provides the control vertices of the B-spline. The idea of using least squares models for fitting curves to digital contours have been also considered in [6], [7], [8]. But we believe that our segmentation process in identifying and modeling only nonlinear segments should produce a more compact representation of the original contour with less computational effort. Moreover, our inclusion of the curvature in the weighted least squares model tenders a more desirable contour. The least squares model, accounting for curvature of the segment, has the effect of smoothing out the *contour noise*, the noise commonly present in a digital contour.

### 4.1 Modeling a Nonlinear Segment

Let  $P = \{P_i = (x_i, y_i), i = 0, 1, \dots, n\}$  be a set of points corresponding to a nonlinear segment. We are interested in finding a set of vertices  $V_j$  so that the spline

$$Q(u) = \sum_{j=0}^m V_j B_{j,k}(u) \quad (15)$$

is a good representation of  $P$  (preferably  $m \ll n$ ). In specifying the B-splines  $B_{j,k}(u)$ , we need to identify (1) the knot sequence  $\{u_j\}_0^{m+k}$ , and (2) the parameters  $\{U_i\}_0^n$  corresponding to the points  $P_i$  so that  $Q(U_i)$  serves as an approximation to  $P_i$ .



Since the use of a uniform knot sequence for B-splines is widely practiced, we let

$$\begin{cases} u_0 = u_1 = \dots = u_{k-1} = k - 1 \\ u_{m+1} = u_{m+2} = \dots = u_{m+k} = m + 1 \\ u_j = u_{j-1} + 1, & j = k, \dots, m. \end{cases} \quad (16)$$

The repetitions of the first and the last  $k$  knots with values as specified in (16) result in interpolating  $V_0$  and  $V_m$ .

We distribute the parameter values  $\{U_i\}$  proportional to the lengths of the appropriate contour segments of  $C$ , as given by (10). We let

$$\begin{cases} U_0 = k - 1 \\ U_i = U_{i-1} + (m - k + 2) \frac{l_i}{L} \equiv (m - k + 2) \frac{s_i}{L}, & i = 1, 2, \dots, n, \end{cases} \quad (17)$$

where  $l_i$ ,  $L$ , and  $s_i$  are respectively the lengths of the line from  $P_{i-1}$  to  $P_i$ , the contour  $C$ , and the contour piece from  $P_0$  to  $P_i$ , as given by (7)-(9). With the assignments as (17), the  $U_i$  are positioned so that  $U_0$  is at  $u_{k-1}$  and  $U_n$  is at  $u_{m+1}$ . Any other  $U_i$  is at a distance on  $[u_{k-1}, u_{m+1}]$ , with length  $m - k + 2$ , proportional to the length of the contour piece (from  $P_0$  to  $P_i$ ) relative to the length of the whole contour  $C$ .

It remains to specify the  $V_j$ . We would like the  $V_j$  so that the model  $Q$ , as in (15), predicts the points  $P_i$  as closely as possible. Moreover, we would also like the predictor model  $Q$  to have as minimal a curvature as possible (the model is to replace a nonlinear segment and this serves to remove the contour noise). Both of these objectives may be weighted and expressed in a linear least squares problem. We will do this in the next section.

## 4.2 Least Squares and the Control Vertices

We propose a weighted least squares problem whose solution supplies the control vertices  $V_j$ . We assign the same weights to the errors of the model at the points, but different weights to the curvature. Letting

$$V_j = (X_j, Y_j), \quad (18)$$

we can write  $Q$  as follows:

$$Q(u) = (Q_1(u), Q_2(u)), \quad (19)$$

where

$$Q_1(u) = \sum_{j=0}^m X_j B_{j,k}(u), \quad (20)$$

and

$$Q_2(u) = \sum_{j=0}^m Y_j B_{j,k}(u). \quad (21)$$

The curvature,  $\kappa(u)$ , of  $Q$  at  $u$  is known to be

$$\kappa(u) = \frac{\|Q''(u)\|_2}{\|Q'(u)\|_2} = \frac{\sqrt{\sum_j X_j B_{j,k}''^2(u) + \sum_j Y_j B_{j,k}''^2(u)}}{\sqrt{\sum_j X_j B_{j,k}'^2(u) + \sum_j Y_j B_{j,k}'^2(u)}}. \quad (22)$$

To simplify the least squares model (to make it linear), we ignore the denominator in (22) and use an approximation for the curvature as below:

$$\kappa(u) \approx \|Q''(u)\|_2 = \sqrt{(\sum X_j B_{j,k}''(u))^2 + (\sum Y_j B_{j,k}''(u))^2}. \quad (23)$$

Denoting

$$X = \begin{bmatrix} X_0 \\ \vdots \\ X_m \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} Y_0 \\ \vdots \\ Y_m \end{bmatrix}, \quad (24)$$

the least squares problem is hence expressed as:

$$\text{minimize } E(X, Y), \quad (25)$$

where

$$E(X, Y) = \sum_{i=0}^n \left\{ \left[ \sum_{j=0}^m X_j B_{j,k}(U_i) - x_i \right]^2 + \left[ \sum_{j=0}^m Y_j B_{j,k}(U_i) - y_i \right]^2 + \mu_X \left( \sum_{j=0}^m X_j B_{j,k}''(U_i) \right)^2 + \mu_Y \left( \sum_{j=0}^m Y_j B_{j,k}''(U_i) \right)^2 \right\}. \quad (26)$$

The objective function  $E$  measures a weighted deviations of the model  $Q$  from the points  $P_i$  as well as its approximate curvatures at the parameters  $U_i$  (representative of the actual points  $P_i$ ). The weights are set the same for the errors of the model at the points  $P_i$ , to  $\mu_X (\geq 0)$  and  $\mu_Y (\geq 0)$  for the two components of the curvature. Of course, the change in the values of  $\mu_X$  or  $\mu_Y$  affects the weights for all the components of  $E$ . The unknowns  $X_j$  and  $Y_j$  are separable in (26), and  $E$  can be written as:

$$E = E_X + E_Y, \quad (27)$$

where

$$E_X = \sum_{i=0}^n \left\{ \left[ \sum_{j=0}^m X_j B_{j,k}(U_i) - x_i \right]^2 + \mu_X \left( \sum_{j=0}^m X_j B''_{j,k}(U_i) \right)^2 \right\}, \quad (28)$$

and

$$E_Y = \sum_{i=0}^n \left\{ \left[ \sum_{j=0}^m Y_j B_{j,k}(U_i) - y_i \right]^2 + \mu_Y \left( \sum_{j=0}^m Y_j B''_{j,k}(U_i) \right)^2 \right\}. \quad (29)$$

The minimization of  $E$  is achieved by separately minimizing  $E_X$  and  $E_Y$ . The procedure is the same for both cases, so we discuss the details only for  $E_X$ . To minimize, we must have

$$\frac{\partial E_X}{\partial X_l} = 0, \quad l = 0, 1, \dots, m. \quad (30)$$

Differentiating  $E_X$  and setting it to zero, we get

$$\sum_{i=0}^n B_{l,k}(U_i) \left[ \sum_{j=0}^m X_j B_{j,k}(U_i) - x_i \right] + \mu_X \sum_{i=0}^n B''_{l,k}(U_i) \sum_{j=0}^m X_j B''_{j,k}(U_i) = 0, \quad (31)$$

or

$$\sum_{i=0}^n \sum_{j=0}^m [B_{l,k}(U_i) B_{j,k}(U_i) + \mu_X B''_{l,k}(U_i) B''_{j,k}(U_i)] X_j = \sum_{i=0}^n x_i B_{l,k}(U_i), \quad (32)$$

or

$$\sum_{j=0}^m X_j \sum_{i=0}^n [B_{l,k}(U_i) B_{j,k}(U_i) + \mu_X B''_{l,k}(U_i) B''_{j,k}(U_i)] = \sum_{i=0}^n x_i B_{l,k}(U_i), \quad (33)$$

$$l = 0, 1, 2, \dots, m.$$

The above equations are the so-called *normal equations* for solving the least squares problem with  $E_X$  as the objective function. Similar equations are obtained for the minimization of  $E_Y$ . The normal equations are also displayed in matrix notation. Let  $B$  and  $B''$  denote matrices with respective  $j$ -th columns shown below

$$B_j = \begin{bmatrix} B_{j,k}(U_0) \\ \vdots \\ B_{j,k}(U_n) \end{bmatrix} \quad \text{and} \quad B''_j = \begin{bmatrix} B''_{j,k}(U_0) \\ \vdots \\ B''_{j,k}(U_n) \end{bmatrix}, \quad (34)$$

and let

$$x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}, \quad (35)$$

then the objective function  $E_X$  (28) is the same as

$$E_X = \| (BX - x) \|_2^2 + \mu_X \| B'' X \|_2^2, \quad (36)$$

and the corresponding normal equations (33) can be written as

$$\sum_{j=0}^m (B_l^T B_j + \mu_X B_l''^T B_j'') X_j = B_l^T x, \quad l = 0, 1, \dots, m, \quad (37)$$

or

$$(B^T B + \mu_X B''^T B'') X = B^T x. \quad (38)$$

The coefficient matrix in (38) is symmetric positive definite, because the parameter values  $U_i$  are distinct, the  $B_{j,k}$  are basis functions, and  $m < n$ . One can make an effective use of the symmetry in (38) and employ the Cholesky factorization (see [4]) to solve the normal equations.

## 5 Computational Considerations and Remarks

We begin with an outline of an algorithm to determine the control vertices  $V_j$  of a nonlinear segment represented by the points  $P_i$ .

**Algorithm:** Control Vertex Computations.

**INPUT:**  $P_i = (x_i, y_i), i = 0, 1, \dots, n$ .

**Set** the weights  $\mu_X$  and  $\mu_Y$ .

**Set**  $m$  and  $k$ .

**Compute** the knots  $\{u_i\}, i = 0, 1, \dots, m + k$  according to (16).

**Compute** the parameters  $\{U_i\}, i = 0, 1, \dots, n$  according to (17).

**Compute** the matrices  $B$  and  $B''$  using (34).

**Solve** the following least squares problems for  $X$  and  $Y$  respectively

$$\begin{aligned} & \text{minimize } (\| (BX - x) \|_2^2 + \mu_X \| B'' X \|_2^2), \\ & \text{minimize } (\| (BY - y) \|_2^2 + \mu_Y \| B'' Y \|_2^2). \end{aligned}$$

**OUTPUT:**  $V_j = (X_j, Y_j), j = 0, 1, \dots, m$ .

## 5.1 Computational Issues

We discuss our choice of settings for the parameters of the algorithm and also point out some alternatives for the least squares model and its solution.

- **Setting  $m$  and  $k$**

We set  $k$ , the order of the spline, to 4. This gives us a cubic model and serves well for most practical purposes. The value of  $m$  is set in relation with the value of  $n$ . Other than imposing the lower bound value of  $k$  and the upper bound of 500, the value of  $m$  is set as follows:

$$m = (n \text{ DIV } 12) + 5. \quad (39)$$

- **Solving the Least Squares Problems**

We solve the least squares problems by the normal equations and the use of Cholesky factorization. It is possible to solve these problems by orthogonalization methods such as the **QR** or the **SVD** (see [4]). For these methods, the coefficient matrix of the objective function  $E_X$  (for use in orthogonalization) is displayed in the following alternative formulation:

$$E_X = \left\| \begin{bmatrix} B \\ \sqrt{\mu_X} B'' \end{bmatrix} X - \begin{bmatrix} x \\ 0 \end{bmatrix} \right\|_2^2. \quad (40)$$

- **Spline Parameters Setting**

Our choice of the parameters  $U_i$  results in a linear least squares model. With the complication of introducing a nonlinear least squares model, one can leave the parameters to be decided as the solution of the model along with the control vertices  $V_j$ . Alternatively, for known values of  $m$  and  $k$ , it may be interesting to investigate the choice of the parameters  $U_i$  so that certain structures may arise in the least squares model.

- **Segment Joints Continuity**

Care is taken to make sure that the segments of the model are joined together at the joints. For this, the last control vertex of a preceding segment and the first control vertex of the succeeding one may both be set to the average of the two vertices. In most of our examples, these two vertices were found to be practically the same without a need for this setting.

## 5.2 Remarks

We used our segmentation scheme along with the linear least squares models (for the nonlinear segments) to represent various digital contours. Figures 5-7 show some representatives. Figure 7 shows the effect of the weights  $\mu_x$  and  $\mu_y$  on the removal of contour noise as well as on the smoothness of the model. Smoother contours may be obtained by increasing these weights. Of course, this is achieved at the expense of losing a slight

Figure 5: A Digital Contour of  $\Psi$  (Right) and its Geometric Model (Left).

Figure 6: A Digital Image (Right) and its Geometric Model (Left).

accuracy in the model predicting the digital image, since the increase of the weights for the curvature results in the relative decrease of the weight for compatibility of the model with the nonlinear segment.

Finally, Figure 8 shows a contour representation constructed from a somewhat complicated image.

### **Acknowledgements**

This work has been supported by Sharif University of Technology. The authors are thankful to Richard Bartels of University of Waterloo for his support and collaborations during the first author's visit at Waterloo. The second author is specially grateful to Joseph Liu of York University for the provision of support and hospitality throughout his sabbatical period at York. Thanks are also due to Eshrat Arjomandi and Andy Mirzaian for their cooperations to facilitate the second author's stay at York.

Figure 7: A Noisy Digital Contour (Right) and its Geometric Models with Increasing Curvature Weights (Left).

## References

- [1] B. A. G. Barsky. *Computer Graphics and Geometric Modeling Using Beta-Splines*. Springer Verlag, 1987.
- [2] R. H. Bartels, J. C. Beaty, and B. A. G. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Kauffman and Morgan, 1987.
- [3] M. Gangent. Approximation of digitized contours. In *Theoretical Foundations of Computer Graphics and CAD, NATO ASI Series, Vol. F40*. Clarendon Press, 1988.
- [4] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- [5] R. C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley, 1987.
- [6] A. Guezic and N. Ayache. Smoothing and matching of 3d curves. In *Lecture Notes in Computer Science 588*. Computer Vision, 1992.
- [7] C. Potier and C. Vercken. Geometric modeling of digitized curves. In *First International Conference on Document Analysis and Recognition*, 1991.
- [8] P. Siant-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision, Antibes*, 1990.
- [9] R. T. Stevens. *The C Graphics Handbook*. Academic Press, 1992.
- [10] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer Verlag, 1980.
- [11] Kai Xin, Kah Bin Lim, and G. Soon Hong. A scale-space filtering approach for visual feature extraction. *Pattern Recognition*, 82(8):1145–1158, 1995.

Figure 8: A Scanned Image (Right) and its Reconstructed Contour (Left).