Curve Representation for Outlines of Planar Images using Multilevel Coordinate Search

MHAMMAD SARFRAZ and NAELAH AL-DABBOUS
Department of Information Science
Kuwait University
Adailiya Campus, P.O. Box 5969, Safat 13060
KUWAIT
prof.m.sarfraz@gmail.com

Abstract: - This paper proposes an optimization technique for the outline capture of planar images. This is inspired by a global optimization algorithm based on multilevel coordinate search (MCS). By starting a search from certain good points (initially detected corner points), an improved convergence result is obtained. The overall technique has various phases including extracting outlines of images, detecting corner points from the detected outline, curve fitting, and addition of extra knot points if needed. The idea of multilevel coordinate search has been used to optimize the shape parameters in the description of the generalized cubic spline introduced. The spline method ultimately produces optimal results for the approximate vectorization of the digital contour obtained from the generic shapes. It provides an optimal fit as far as curve fitting is concerned. The proposed algorithm is fully automatic and requires no human intervention. Implementation details are sufficiently discussed. Some numerical and pictorial results are also demonstrated to support the proposed technique.

Key-Words: - Optimization, multilevel coordinate search, Generic shapes, curve fitting, cubic spline

1 Introduction

Capturing and vectorizing outlines of images is one of the important problems of computer graphics, vision, and imaging. Various mathematical and computational phases are involved in the whole process. This is usually done by computing a curve close to the data point set [3-5, 23-25]. Computationally economical and optimally good solution is an ultimate objective to achieve the vectorized outlines of images for planar objects.

The representation of planar objects in terms of curves has many advantages. For example, scaling, shearing, translation, rotation and clipping operations can be performed without any difficulty. Although a good amount of work has been done in the area [10-16, 31], it is still desired to proceed further to explore more advanced and interactive strategies. Most of the up-to-date research has tackled this kind of problem by curve subdivision or curve segmentation. Curve segmentation is advantageous in a way that it gives a rough geometry of the shape. Approaches used to achieve this task, in the literature, are polygonal approximations [8, 13], circular arc approximations [10,15,17,18, 22] and approximations using cubics or higher order spline functions [2,14, 24-25].

A non-parametric dominant point detection algorithm was proposed in [8], it used the dominant points for polygonization of digital curves. The problem with polygonal approximation is that these approaches are rarely used for shape analysis. A combination of line segments and circular arcs for object approximation is used in [17, 18]. A scheme to construct a curvature continuous conic spline is proposed in [15]. This approach presented the conic spline curve fitting and fairing algorithm using conic arc scaling. The smoothing is done by removing unwanted curvature extrema. Similar algorithms for data fitting by arc spline curves are presented in [22]. A method for segmentation of curves into line segments and circular arcs by using types of breakpoints is proposed in [10]. Advantage of this technique is that it is threshold free and transformation invariant. Five categories of breakpoints have been defined. The line and conic segmentation and merging is based on these breakpoints.

Least square fitting is mostly adopted in approximations, which uses splines and higher order polynomials. Some approaches are based on active contour models known as snakes. These techniques are also based on parameterization. Enhancement to the scheme by adjusting both number and positions

of control points of the active spline curve is shown in [14]. This scheme is based on curve approximation using iterative optimization with B-spline curve by squared distance minimization.

Another way, other than parametric form, is to use implicit form of the polynomial. Curve reconstruction problem is solved by approximating the point clouds using implicit B-spline curve [12]. The authors have used trust region algorithm in optimization theory as minimization heuristics. Techniques described for fitting implicitly defined algebraic spline curves and surfaces to scattered data by simultaneously approximating points and associated normal vectors are proposed in [19, 20, 21].

The proposed work, in this paper, is inspired by the fast growing area of soft computing. A good amount of literature has been produced on various heuristics [30-32] for optimization problems. The proposed work is motivated by an optimization algorithm based on multilevel coordinate search (MCS) by Huyer and Neumaier [30]. It motivates the author to an optimization technique proposed for the outline capture of planar images. It is an extension of the work in [25]. In this paper, the data point set represents any generic shape whose outline is required to be captured. We present an iterative process to achieve our objectives. The algorithm comprises of various phases to achieve the target. First of all, it finds the contour of the gray scaled bitmap image. Secondly it detects corners. These phases are considered as preprocessing steps. The next phase detects the corner points on the digital contour of the generic shape under consideration. The idea of multilevel coordinate search (MCS) is then used to fit a generalized cubic spline which passes through the corner points. It globally optimizes the shape parameters in the description of the generalized cubic spline to provide a good approximation to the digital curve.

In most of the cases, corner points are not enough to approximate the digital object and hence some more points are also needed. These points are known as break points or knots as they are used to break a segment for better approximation. For onwards discussion, the set of corner points together with the break points will be called as the set of significant points. In the fourth phase of the proposed algorithm, for each iteration, we will insert a point as knot in every piece (if needed) in a manner that the distance, d, of the computed point on the spline curve and its corresponding contour point is greater than a threshold ε. This process increases the set of significant points and hence needs multilevel coordinate search to be employed again for the updated set of significant points to fit an optimal spline curve. This process continues until it rectifies the solution and helps towards the objective optimization in a global fashion. We stop the iterative process when all d's are less than ε. The proposed spline method, using multilevel coordinate search, ultimately produces optimal results for vectorizing the digital contour of the generic shapes. It provides an optimal fit as far as curve fitting is concerned.

The organization of the paper is as follows, Section 2 discusses about pre-processing step which includes finding the boundary of planar object and corner detection algorithm for finding the significant points. Section 3 is about the interpolant form of cubic spline curves and computation of its associated tangents. The process of multilevel coordinate search is explained in Section 4. Overall methodology of curve fitting is explained in Section 5, it includes the idea of knot insertion as well as the algorithm design for the proposed vectroization scheme. Demonstration of the proposed scheme is presented in Section 6. Finally, the paper is concluded in Section 7.

2 Preprocessing

The proposed scheme starts with first finding the boundary of the generic shape and then using the output to find the corner points or the significant points. Forthcoming Sections 2.1 and 2.2 will explain these phases.

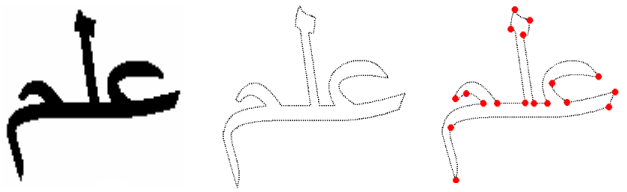
2.1 Finding Boundary of Generic Shapes

The image of the generic shape can be acquired either by scanning or by some other mean. The quality of scanned images is dependent upon factors such as paper quality and scanning resolution. The better the resolution and paper quality, the better will be the image. The aim of boundary detection is to produce an object's shape in graphical or non-scalar representation. Chain codes [6, 7, 27-28] are the most widely used representations. Other well-known representations are syntactic techniques, boundary approximations and scale-space techniques. The benefit of using chain code is that it gives the direction of edges. The boundary points are selected as contour points based on their corner strength and fluctuations. To arrange the extracted boundary points in a sequence (clockwise direction), a boundary tracing is performed, using the algorithm in [26] for boundary tracing. Demonstration of the

method can be seen in Figure 1(b) which is the contour of the bitmap image shown in Figure 1(a).

2.2 Detecting Corner Points

Corners in digital images give important clues for the shape representation and analysis. Generally objects information can be represented in terms of its corners, which play a very vital role in object recognition, shape representation and image interpretation [1,8]. These are the points that partition the boundary into various segments. The strategy of getting these points is based on the method proposed in [1]. The details of this procedure is left for the reader to see in [1]. The demonstration of the algorithm is made on Figure 1(b). The corner points of the image are shown in Figure 1(c).



(a) (b) (c)

Figure 1. Pre-processing Steps: (a) Original Image, (b) Outline of the image, (c) Corner points achieved.

3 Curve Fitting with Cubic Spline

The motive of finding the corner points, in Section 2.2, was to divide the contour into pieces. Each piece contains the data points in between two subsequent corners inclusive. This means that if there are m corner points $cp_1, cp_2, \ldots, cp_m$ then there will be m pieces $pi_1, pi_2, \ldots, pi_m$. We treat each piece separately and fit the spline [9, 24-25] to it. First piece includes all the contour points in between cp_1 and cp_2 inclusive. Second piece contains all contour points in between cp_2 and cp_3 inclusive. Consequently, the m^{th} piece contains all contour points between cp_m and cp_1 inclusive. In general, the i^{th} piece contains all the data points between cp_i and cp_{i+1} inclusive.

After breaking the contour of the image into different pieces, we fit the spline curve to each piece. For this purpose we have used piecewise parametric cubic spline interpolant. The spline formulation globally is C^1 continuous.

3.1 Cubic Spline Interpolant

The cubic spline is defined as follows:

$$P(t) = (1-\theta)^3 F_i + 3\theta(1-\theta)^2 V_i + 3\theta^2(1-\theta)W_i + \theta^3 F_{i+1} \quad (1)$$

where

$$\theta|_{[t_i,t_{i+1})}(t) = \frac{(t-t_i)}{h_i},\ h_i = t_{i+1} - t_i$$

and

$$V_i = F_i + \frac{h_i D_i}{3},\ W_i = F_{i+1} - \frac{h_i D_{i+1}}{3}$$

Equation (4) can be rewritten as

$$P|_{(t_i,t_{i+1})}(t) = R_{0,i}(t)F_i + R_{1,i}(t)V_i + R_{2,i}(t) + R_{3,i}(t)F_{i+1} \quad (2)$$

where

$$\left.\begin{aligned} R_{0,i}(t) &= (1-t)^3, \\ R_{1,i}(t) &= 3t(1-t)^2, \\ R_{2,i}(t) &= 3t^2(1-t), \\ R_{3,i}(t) &= t^3, \end{aligned}\right\} \quad (3)$$

The functions $R_{j,i}$, j = 0,1,2,3 are Bernstein Bézier like basis functions, such that

$$\sum_{j=0}^{3} R_{j,i}(t) = 1 \quad (4)$$

From the Bernstein-Bézier theory, it follows that the curve segment $P|_{[t_i,t_{i+1}]}$ lies in the convex hull of the control points $\{F_i, V_i, W_i, F_{i+1}\}$ and is variation diminishing with respect to the control polygon joining these points.

To get the control points $\{F_i, V_i, W_i, F_{i+1}\}$, we make use of a Bernstein-Bézier representation where we can impose the Hermite interpolation conditions:

$$P(t_i) = F_i \text{ and } P'(t_i) = D_i,\ i \in Z \quad (5)$$

where F_i and F_{i+1} are corner points of i^{th} piece. D_i and D_{i+1} are the corresponding tangents at corner points.

To construct the parametric C^1 cubic spline interpolant on the interval $[t_0, t_n]$ we have $F_i \in R^m$, $i = 0,1,\ldots\ldots,n$, as interpolation data, at knots t_i, $i = 0,1,\ldots\ldots,n$. The derivatives $D_i \in R^m$ can be found out by the imposition of C^1 constraints on the piecewise cubic form. The C^1 constraints can be written as:

$$P'(t_i^+) = P'(t_i^-). \quad (6)$$

The tangent vectors are calculated as follows:

$$\left.\begin{aligned} D_0 &= 2(P_1 - P_0) - \frac{(P_2 - P_0)}{2} \\ D_i &= a_i(P_i - P_{i-1}) + (1-a_i)(P_{i+1} - P_i) \\ D_n &= 2(P_n - P_{n-1}) - \frac{(P_n - P_{n-2})}{2} \end{aligned}\right\}, \quad (7)$$

where

$$a_i = \frac{\|P_{i+1} - P_i\|}{\|P_{i+1} - P_i\| + \|P_i - P_{i-1}\|}.$$

Since, the objective of the paper is to come up with an optimal technique which can provide a decent curve fit to the digital data. Therefore, the interest would be to compute the curve in such a way that sum square error of the computed curve with the actual curve (digitized contour) is minimized. Mathematically, the sum squared distance is given by:

$$S_i = \sum_{j=1}^{m_i} \left[P_i(u_{i,j}) - P_{i,j}\right]^2,\ t_{i,j} \in [t_i, t_{i+1}],\ i = 0,1,...,n-1 \quad (8)$$

where

$$P_{i,j} = (x_{i,j}, y_{i,j}),\quad j = 1,2,\ldots,m_i, \quad (9)$$

are the data points of the *i*th segment on the digitized contour. The parameterization over *t*'s is in accordance with the chord length parameterization. Thus the curve fitted in this way will be a candidate of best fit.

3.2 Generalized Cubic Spline Interpolant

The curve fitted in Section 3.1 is a candidate of best fit, but it may not be a desired fit. This leads to the need of introducing some shape parameters in the description of the cubic spline. This section deals with the generalized form of cubic spline. It introduces two parameters *v* and *w* in the description of cubic spline defined as follows:

$$P(t) = (1-\theta)^3 F_i + 3\theta(1-\theta)^2 V_i + 3\theta^2(1-\theta)W_i + \theta^3 F_{i+1} \quad (10)$$

where

$$V_i = F_i + h_i v_i D_i,\ W_i = F_{i+1} - h_i w_i D_{i+1} \quad (11)$$

It follows from Bézier theory that the curve segment $P(t)$ lies in the convex hull of the control points $\{F_i, V_i, W_i, F_{i+1}\}$ and is variation diminishing with respect to the *control polygon* joining these points. It should also be observed that the weights are independent and that one could take $v_i = w_i$ a constant for all $i \in Z$ without loss of generality. However, we find it useful to consider the two scalar weights v_i and w_i on $[t_i, t_{i+1})$ in the development of the theory.

Remark 1. If $P(t)$ is the interpolant for scalar data $F_i \in R$, with derivatives $D_i \in R, i \in Z$, then $(t, p(t))$ can be considered as the interpolation scheme applied in R^2 to data (t_i, F_i), with derivatives $(1, D_i)$, $i \in Z$. This is a consequence of the property that the interpolant is able to reproduce linear functions. In particular, for $v_i = w_i = 1$, the scalar data $F_i := t_i$, and derivatives $D_i := 1, i \in Z$, the interpolant reproduces the function t.

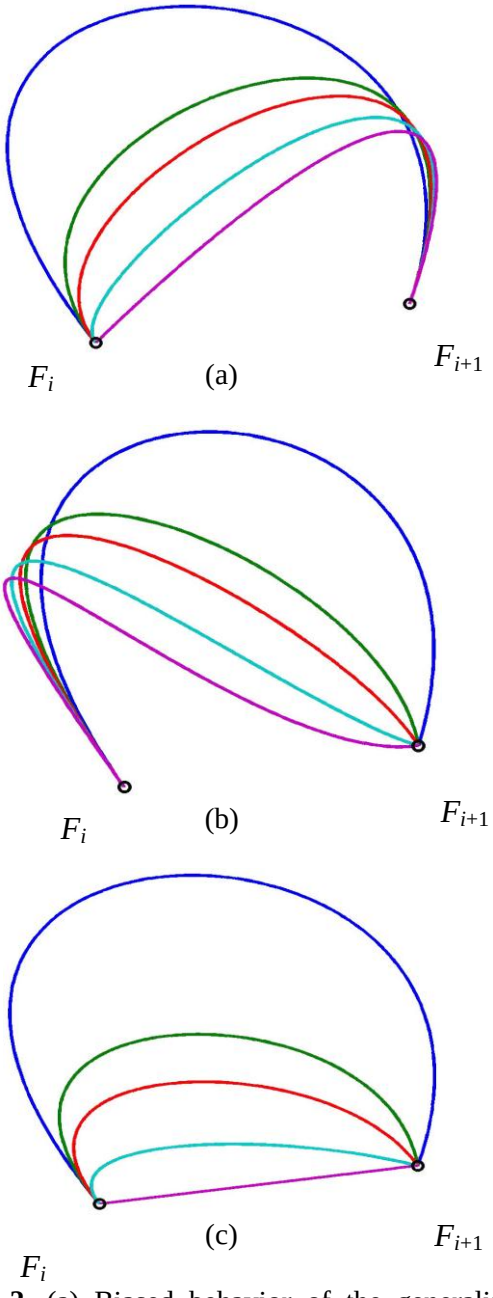
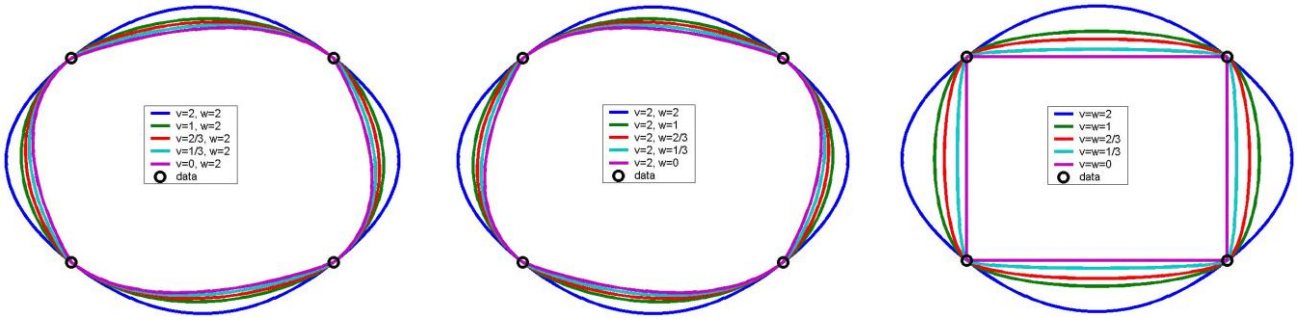


Figure 2. (a) Biased behavior of the generalized cubic for different values of *v* with *w* = 0, (b) Biased behavior of the generalized cubic for *v* = 0 and different values of *w*, (c) interval tension behavior of the generalized cubic for the same values of *v* and *w*.

Remark 2. If $P(t)$ is the interpolant for planar data $F_i \in R^m$, $m \geq 2$, we need to have some specific parametrization over t. Although, there are number of parametrization schemes in the literature, we will prefer to use the chord lenght parametrization as follows:

$$\left.\begin{aligned} t(0) &= 0; \\ t(i) &= t(i-1) + |F_i F_{i+1}|,\ i = 1,2,3,....n, \end{aligned}\right\} \quad (12)$$

where $|F_i F_{i+1}|$ denotes the distance of the i th chord segment. Some more details about the parametrization is given in the next section.



(a) (b) (c)

Figure 3. (a) Global biased behavior of the generalized cubic for different values of v with w = 0, (b) Global biased behavior of the generalized cubic for v = 0 and different values of w, (c) Global interval tension behavior of the generalized cubic for the same values of v and w.

The following *tension* properties of the Hermite like form are now immediately apparent from (10) and (11).

Biased Tension: The biased tension behavior is possible for the curve designing. For the *biased tension behavior to the left*, one can observe that for any $i \in Z$, one can have the following:

$$\lim_{v_i \to 0} V_i = F_i \quad (13)$$

This behavior follows from equation (11). For the demonstration, see Figure 2(a) for local behavior in one interval of the design curve, and see Figure 3(a) for global behavior in the whole curve. Similarly, for the *biased tension behavior to the right*, one can observe that for any $i \in Z$, one can have the following:

$$\lim_{w_i \to 0} W_i = F_{i+1}. \quad (14)$$

This behavior follows from equation (11), see Figures 2(b) and 3(b) for local and global behavior respectively.

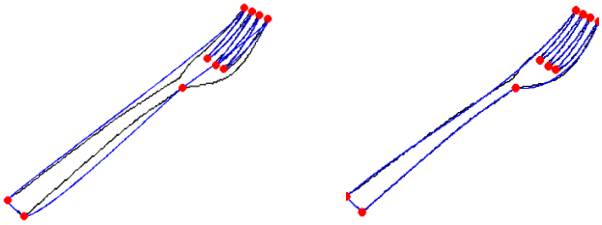
Interval Tension: The interval tension behavior is possible for the curve designing too. For any $i \in Z$, one can have the following:

$$\lim_{v_i \to 0} V_i = F_i \text{ and } \lim_{w_i \to 0} W_i = F_{i+1} \quad (15)$$

and the interpolant (10) reduces to:

$$\lim_{v_i, w_i \to 0} P(t) = (1-\theta)F_i + \theta F_{i+1}. \quad (16)$$

For the demonstration, see Figure 2(c) for local behavior in one interval of the design curve, and see Figure 3(c) for global behavior in the whole curve.



(a) (b)

Figure 4. Curve fitting on a data of a fork (a) Default cubic spline, (b) Variety of shape control used.

A variety of shape control has been demonstrated in the Figure 4, where different shape parameter values have been used to get a desired shape in Figure 4(e). The Figure 4(a) the default cubic spline fitted on the data points of a fork image.

Remark 3. The case $v_i = w_i = r_i$ is that of the interval tension method. Obviously, the parameter values $v_i = w_i = 1/3$ provide the special case of cubic spline of Section 3.1. Otherwise, these parameters can be used to loose or tight the curve. This paper proposes an evolutionary technique, namely multilevel coordinate search (MCS), to

optimize these parameters so that the curve fitted is optimal.

4 Multilevel Coordinate Search

Multi-level coordinate search (MCS) is a global optimization technique [30]. It guarantees the converge of the optimal solution if the function is continuous in the neighborhood of a global minimizer. It works by combining two types of searching: global searching and local searching. The advantage is that if the optimal value is somewhere near the current position, local search makes sure that the algorithm does not divert to distant locations in the solution space. It also reduces the time to reach the exact optimal value after reaching near it. MCS makes use of other complex implemented techniques such as global line search and bound constrained quadratic program solver [30]. Derivation of the MCS algorithm and underlying theory can be found in [30]. A detailed description of the mapping of the MCS technique on our problem is given in the Section 5.

5 Proposed Approach

The proposed approach to the curve problem is described here in detail. It includes the phases of problem matching with MCS using cubic spline, description of parameters used for MCS, curve fitting, and the overall algorithm design.

5.1 Problem Mapping

This section describes about the MCS formulation of the solution to the problem in detail. Our interest is to optimize the values of shape parameters *v* and *w*, in the description of the spline in Section 3.2, such that the defined curve fits as close to the original contour segment as possible. We use MCS for the optimization of these two variables for the fitted curve. Hence the dimensionality of the solution space is 2, and each point in MCS represents a pair of values for *v* and *w*. We start with an initial set of points that are taken to be the corner points of the 2-dimensional solution space and the midpoints along the two directions. Since the solution space is bounded, with boundary values as -1 and 1 for both the dimensions, the initial points are chosen at these corners. Then we make boxes of the solution spaces using these points. For each point, we also compute and store the objective function value and associate each with one of the boxes. Now each box corresponds to a range of values of *v* and *w*. From all these boxes (ranges of *v* and *w* values), we first select the one having an associated point with the lowest function value. In this box, we apply local search and try to find the optimum in the determined direction of minimization within the box. If the *v* and *w* pair found in this box is not the optimal solution, then this box is split. That is, the range of *v* and *w* values within this box is further split into smaller mutually exclusive ranges. Each new range is associated with a new representative point in the solution space and its fitness value. The shopping basket is hence kept updated with these ranges and fitness values.

Note that we apply MCS independently for each segment of a contour between two consecutive corner points that we have identified using corner point algorithm. MCS is applied sequentially on each of the segments, generating an optimized fitted curve for each segment. The algorithm is run until the maximum level of allowed splitting is reached, or an optimal value is reached. Once, all the contour segments are exhausted and still the desired global optimum solution is not achieved, MCS is applied again. MCS is applied sequentially on each of the segments.

5.1.1 Initialization

Once we have the bitmap image of a generic shape, the boundary of the image can be extracted using the method described in Section 2.1. After the boundary points of the image are found, the next step is to detect corner points as explained in Section 2.2. This corner detection technique assigns a measure of 'corner strength' to each of the points on the boundary of the image. This step helps to divide the boundary of the image into *n* segments. Each of these segments is then approximated by interpolating spline described in Section 3.2. The initial solution of spline parameters (*v* and *w*) are randomly selected within the range [-1, 1].

5.1.2 Initialization

After an initial approximation for the segment is obtained, better approximations are obtained through MCS to reach the optimal solution. We experiment with our system by approximating each segment of the boundary using the generalized cubic spline of Section 3.2. This spline method is a variation of the well-known Hermite cubic spline. This modified Hermite cubic spline provides greater control on the shape of the curve and also efficient to compute. The tangents, in the description of the spline, are computed using least square method. Each boundary segment is approximated by the spline. The shape parameters *v* and *w*, in the cubic

spline, provide greater flexibility over the shape of the curve. These parameters are adjusted using MCS to get the optimal fit.

Once an initial fit for a particular segment is obtained, the parameters of the fitted curve (*v*'s and *w*'s) are adjusted to get better fit. Here, we try to minimize the sum squared error. Using MCS, we try to obtain the optimal values of the curve parameters. We choose this technique because it is powerful, yet simple to implement and as shown in Section 6, performs well for our purpose.

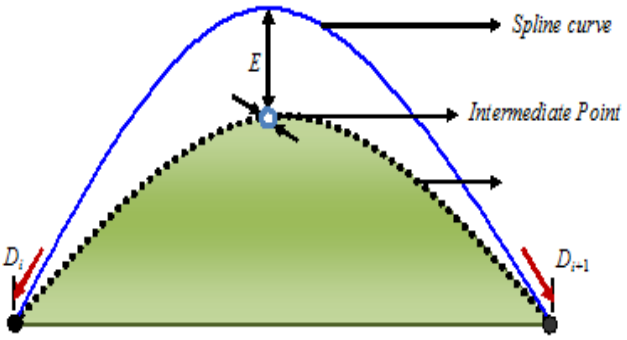


Figure 5. Calculation of Intermediate Point (a hollow bullet).

5.1.3 Segmentation using Intermediate Points

For some segments, the best fit obtained through iterative improvement may not be satisfactory. In that case, we subdivide the segment into smaller segments at points where the distance between the boundary and parametric curve exceeds some predefined threshold, see Figure 5. Such points are termed as *intermediate points*. A new parametric curve is fitted for each new segment.

5.2 The Algorithm

We can summarize all the phases from digitization to optimization discussed in the previous sections. The algorithm of the proposed scheme is contained on various steps as shown in the Pseudo code in Figure 6. A detailed description, describing the whole system with step by step flow, is shown in the flowchart demonstrated in Figure 7.

5.3 MCS Parameters

Although MCS sets default values (see Table 1) of the algorithm variables, it gives the option of manipulating some parameters that define various factors affecting its performance. One of the factors is that how much weight MCS should give to global searching as opposed to local searching. The higher this value, the more global level search will be done. Similarly, another parameter that defines how much local search to do is also specified.

```
Extract Contours from image
 For each extracted contour
  Detect Corner Points of contour
   For each segment between Corner Points
    Compute Parameterization for segment
    Compute Tangent Vectors for segment using Least Square Method
     MCS starts: Initialize with starting point, generate initial
     set of points, generate initial boxes
     Do
      Find lowest level with un-split box
      Find box with lowest function value
      Build and minimize local quadratic model
      Do line search in determined direction
        Calculate fitness value i.e. total distance between
        contour points and corresponding curve points defined by
        the given values of v and w from current point in
        solution space
        If fitness value is equal to or close to optimal value,
        then stop with this point as optimal solution
        Else
          Split box
          Update shopping basket
        End If
      While maximum splitting level is not reached
    End contour segments loop
End contours loop
```

Figure 6. Pseudo-code Algorithm with MCS.

An initial set of starting solution points have to be specified for the system to start with. MCS requires an initial guess for the solution. It is this starting state parameters that affect the performance of the algorithm. If the starting solution is very near the optimal solution, it is more likely to find the optimal solution readily than if the starting solution is distant from the optimal solution. An acceptable error value has to be defined, so that if the system comes within this error range from the optimal value, it terminates with the found solution.

An overall constraining factor is the maximum number of epochs that the algorithm may run, so that it does not run indefinitely if it is not reaching a stable solution after that number of epochs. The direction of optimization of the fitness function has to be specified i.e. specific value that has to be attained. The default value is negative infinity and it can be used for our problem since the lowest value for our objective function is zero. The dimension of the problem has to be defined as the number of inputs that will be passed to MCS, and the allowable range of these variables. Table 1 shows the MCS parameter settings that have been used for the proposed curve fitting optimization problem in a global way.

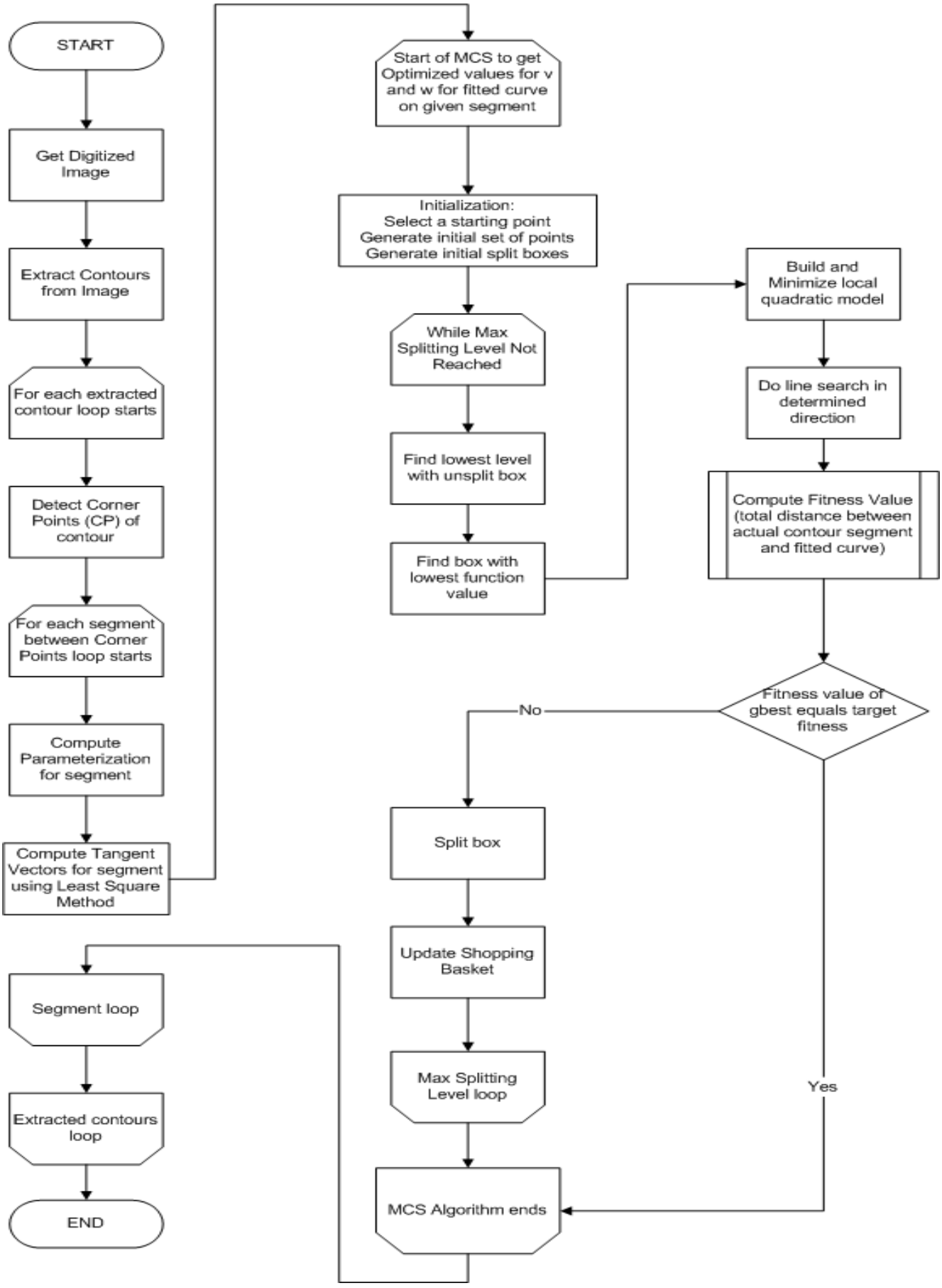


Figure 7. Flowchart of the system with MCS.

Table 1. Parameter Settings for MCS.

MCS parameters	Values
Range of input parameters *v* and *w*	[-1,1]
Fitness Function Optimization Target	0 (function minimization)
Dimension of problem (number of inputs to MCS)	2
Weight given to global search versus local search	20
Maximum number of iterations (epochs)	200
Stopping relative error (if distance from optima is less than this, the algorithm terminates)	1e-4
Initial Set of Solution Points	Corner-points and Mid-points of solution space
Number of Steps in Local Search	50

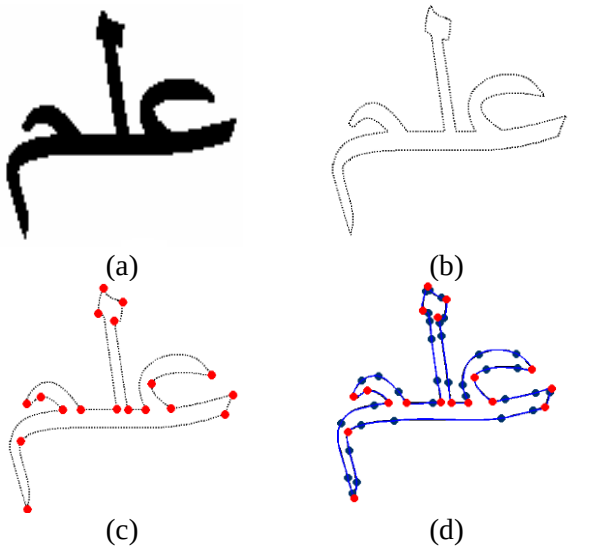


Figure 8. Pre-processing Steps: (a) Original Image, (b) Outline of the image, (c) Corner points achieved, (d) Fitted Outline of the image.

6 Demonstration

The algorithm in Section 5 has been implemented practically and the proposed curve scheme has been implemented with and without intermediate point incorporation. Use of MCS has provided pleasing and efficient results. We evaluate the performance of our system by fitting parametric curves to different binary images.

Figure 8 shows the implementation results of the algorithm with MCS. Figures 8(a), 8(b), 8(c) and 8(d) are respectively the original image of an Arabic language word "Ilm", its outline, outline together with corner points detected, and the fitted outline together with corner points and intermediate points.

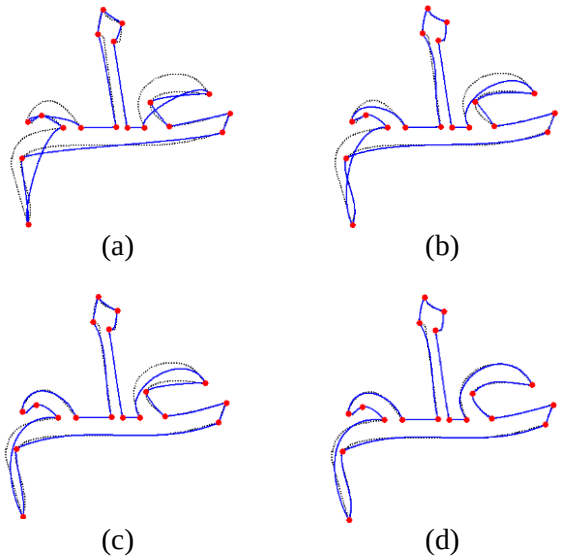
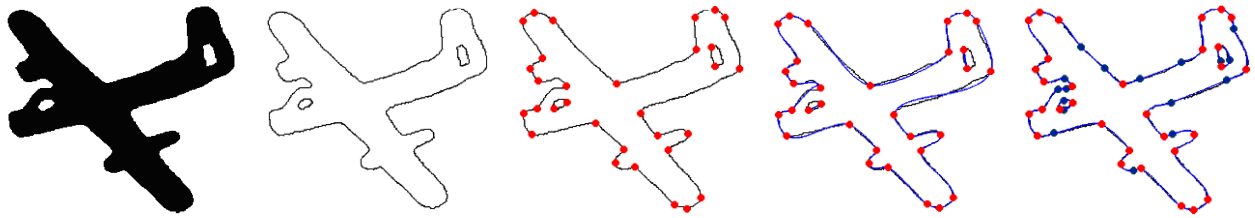


Figure 9. Demonstration of spline fitting at different iterations (average per segment) using MCS without Intermediate Point: (a) 1st iteration, (b) 5th iteration, (c) 10th iteration, (b) 62th (final) iteration.

Figures 9(a), 9(b), 9(c) and 9(d) demonstrate, respectively, 1st, 5th, 10th and last (62th) iteration of the algorithm using MCS without inserting any intermediate point. Since the number of iterations may not be same for each segment, therefore the number of iterations for the whole curve mentioned here actually represents the average number of iterations per segment. This experiment is done without inserting any intermediate point in any of the curve segment. One can notice that after some iterations, although an approximation curve has been achieved, still it is required to have some further improvements. Figures 8(d) demonstrates the improvement in the output. This is done by inserting some appropriate intermediate points in the desired curve segments. The process of such an insertion has been explained in Section 5.1.3. One can notice that after some insertions, a pleasing approximate curve has been achieved. However, some cost of having some intermediate points has been paid. This cost, the author believes, is bearable as the computation time consumed is not very significant as compared to the time paid to achieve not a good approximation in Figure 9. Moreover, accuracy

achieved in Figure 8 is much higher and visually acceptable.



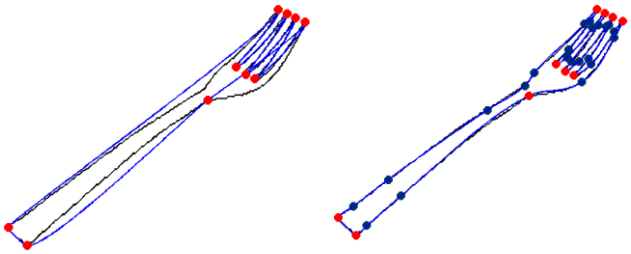
(a) (b) (c) (d) (e)

Figure 10. Pre-processing steps for curve fitting (a) Image of a plane, (b) Extracted outline (c) Initial corner points.

Some more experiments are done on different images. An image of a plane is shown in Figure 10(a), its outline is detected in Figure 10(b), and the corner points are shown in 10(c). Figures 10(d) and 10(e) demonstrate the fitted curves to the outline of Figure 10(b) corresponding to the proposed scheme without and with insertion points respectively. It can be noticed that the fitted curve in Figure 10(d) has a good approximation without inserting extra points. However, inserting extra points, has highly refined the approximation in Figure 10(e).

(a) (b) (c)

Figure 11. Pre-processing steps for curve fitting (a) Image of a fork, (b) Extracted outline (c) Initial corner points.



(a) (b)

Figure 12. Cubic curve fitting (a) without intermediate points (b) with intermediate points.

Another experiment is made on an image of Fork in Figure 11(a). Its outline is detected in Figure 11(b), and the corner points are shown in 11(c). Figures 12(a) and 12(b) demonstrate the fitted curves to the outline of Figure 11(b) corresponding to the scheme without and with insertion points respectively. It can be noticed that the fitted curve in Figure 12(a) has a good approximation, without inserting extra points, except at two segments. However, inserting extra points, has highly refined the approximation everywhere in Figure 12(b).

Table 2. Names and contour details of images.

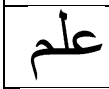
Image	Name	# of Contours	# of Contour Points
	Ilm.bmp	1	[1641]
	Plane.bmp	3	[1106+61+83]
	Fork.bmp	1	[693]

Table 3. Comparison of number of initial corner points, intermediate points and total time taken (in seconds) for cubic interpolation approaches.

Image	# of Initial Corner Points	# of Intermediate Points in Cubic Interpolation	Total Time Taken For Cubic Interpolation	
			Without Intermediate Points	With Intermediate Points
Ilm.bmp	18	34	46.312	164.17
Plane.bmp	31	13	56.766	100.58
Fork.bmp	10	22	18.438	70.297

Tables 2 to 4 summarize the experimental results for different bitmap images. These results highlight various information including contour details of

images (Table 2), intermediate points (Table 3), and number of iterations (Table 4).

Table 4. Comparison of number of epochs taken by MCS cubic interpolation approach with and without intermediate points.

Image	# of Epochs taken by MCS	
	Cubic Interpolation Without Intermediate Points	Cubic Interpolation With Intermediate Points
Ilm.bmp	2459	8915
Plane.bmp	4726	7613
Fork.bmp	1035	4690

4 Conclusion

A global optimization technique, based on multilevel coordinate search, is proposed for the outline capture of planar images. The proposed technique uses the multilevel coordinate search to optimize a cubic spline to the digital outline of planar images. By starting a search from certain good points (initially detected corner points), an improved convergence result is obtained. The overall technique has various phases including extracting outlines of images, detecting corner points from the detected outline, curve fitting, and addition of extra knot points if needed. The idea of multilevel coordinate search has been used to optimize the shape parameters in the description of the generalized cubic spline introduced. The spline method ultimately produces optimal results for the approximate vectorization of the digital contour obtained from the generic shapes. It provides an optimal fit with an efficient computation cost as far as curve fitting is concerned. The proposed algorithm is fully automatic and requires no human intervention. Implementation details are sufficiently discussed using both with and without insertion of intermediate points. The proposed technique has been supported with numerical and pictorial results demonstrated.

Acknowledgement

This work was supported by Kuwait University, Research Grant No. [WI 01/09].

References:

[1] D. Chetrikov, S. Zsabo, A Simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves, *The Proceedings of the 23rd Workshop of the Australian Pattern Recognition Group*, 1999, pp. 1751-2184.

[2] M. Sarfraz, Representing Shapes by Fitting Data using an Evolutionary Approach, *International Journal of Computer-Aided Design & Applications*, Vol. 1(1-4), 2004, pp 179-186.

[3] A. Goshtasby, Grouping and Parameterizing Irregularly Spaced Points for Curve Fitting, *ACM Transactions on Graphics*, 2000, pp. 185-203.

[4] M. Sarfraz, M. A. Khan, An Automatic Algorithm for Approximating Boundary of Bitmap Characters, *Future Generation Computer Systems*, 2004, pp. 1327-1336.

[5] M. Sarfraz , Some Algorithms for Curve Design and Automatic Outline Capturing of Images, *International Journal of Image and Graphics*, 2004, pp. 301-324.

[6] Z. J. Hou, G.W.Wei, A New Approach to Edge Detection, *Pattern Recognition*, 2002, pp. 1559-1570.

[7] P. Reche, C. Urdiales, A. Bandera, C. Trazegnies, F. Sandoval, Corner Detection by Means of Contour Local Vectors, *Electronic Letters*, Vol. 38, No. 14, 2002.

[8] M. Marji, P. Siv, A New Algorithm for Dominant Points Detection and Polygonization of Digital Curves, *Pattern Recognition*, 2003, pp. 2239-2251.

[9] M. Sarfraz, Designing Objects with a Spline, International *Journal of Computer Mathematics*, Taylor & Francis, Vol. 85, No. 7, 2008.

[10] Wu-Chih Hu, Multiprimitive Segmentation Based on Meaningful Breakpoints for Fitting Digital Planar Curves with Line Segments and Conic Arcs, *Image and Vision Computing*, 2005, pp. 783-789.

[11] H. Kano, H. Nakata, C. F. Martin, Optimal Curve Fitting and Smoothing using Normalized Uniform B-Splines: A Tool for Studying Complex Systems, *Applied Mathematics and Computation*, 2005, pp. 96-128.

[12] Z. Yang, J. Deng, F. Chen, Fitting Unorganized Point Clouds with Active Implicit B-Spline Curves, *Visual Computer*, 2005, pp. 831-839.

[13] G. Lavoue, F. Dupont, A. Baskurt, A New Subdivision Based Approach for Piecewise Smooth Approximation of 3D Polygonal Curves, *Pattern Recognition*, 2005, pp. 1139-1151.

[14] H. Yang, W. Wang, J. Sun, Control Point Adjustment for B-Spline Curve

Approximation, *Computer Aided Design*, 2004, pp. 639-652.

[15] X. Yang, Curve Fitting and Fairing using Conic Spines, *Computer Aided Design*, 2004, pp. 461-472.

[16] M. Sarfraz, Computer-Aided Reverse Engineering using Simulated Evolution on NURBS, *International Journal of Virtual & Physical Prototyping*, Vol. 1, No. 4, 2006, pp. 243-257.

[17] J. H. Horng, An Adaptive Smooting Approach for Fitting Digital Planar Curves with Line Segments and Circular Arcs, *Pattern Recognition Letters*, 2003, pp. 565-577.

[18] B. Sarkar, L. K. Singh,, D. Sarkar, Approximation of Digital Curves with Line Segments and Circular Arcs using Genetic Algorithms, *Pattern Recognition Letters*, 2003, pp. 2585-2595.

[19] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and Representation of 3D Objects with Radial Basis Functions, *Proceedings of SIGGRAPH 01, 6776*, 2001.

[20] B. Juttler, A. Felis, A Least Square Fitting of Algebraic Spline Surfaces, *Advance Computer Mathematics*, 2002, pp. 135-152.

[21] B. S. Morse, T. S. Yoo, D. T. Chen, P. Rheingans, K. R. Subramanian, Interpolating Implicit Surfaces from Scattered Surface Data using Compactly Supported Radial Basis Functions, *SMI 01 Proceedings of the International Conference on Shape Modeling and Applications, 8998*, IEEE Computer Society, Washington DC, 2001.

[22] X. N. Yang, G. Z. Wang, Planar Point Set Fairing and Fitting by Arc Splines, *Computer Aided Design*, 2001, pp. 35-43.

[23] M. Sarfraz, M. Riyazuddin, M. H. Baig, Capturing Planar Shapes by Approximating their Outlines, *International Journal of Computational and Applied Mathematics*, Vol. 189, No. 1-2, 2006, pp. 494 – 512.

[24] M. Sarfraz, A. Rasheed, A Randomized Knot Insertion Algorithm for Outline Capture of Planar Images using Cubic Spline, *The Proceedings of The 22th ACM Symposium on Applied Computing* (ACM SAC-07), Seoul, Korea, 2007, pp. 71 – 75, ACM Press.

[25] M. Sarfraz, Outline Capture of Images by Multilevel Coordinate Search on Cubic Splines, *Lecture Notes in Artificial Intelligence: Advances in Artificial Intelligence*, A. Nicholson, X. Li (Eds.): Vol. 5866, Springer-Verlag Berlin Heidelberg, 2009, pp. 636–645.

[26] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, Optimization by Simulated Annealing, *Science*, Vol. 220(4598), 1983, pp. 671-680.

[27] H. Freeman, L.S. Davis, A corner finding algorithm for chain-coded curves, *IEEE Trans. Computers*, Vol. 26, 1977, pp. 297-303.

[28] M. Sonka, V. Hlavac, R. Boyle, *Image processing, analysis, and machine vision*. Brooks/Cole publication, 2001, pp. 142-143.

[29] N. Richard, T. Gilbert, Extraction of Dominant Points by estimation of the contour fluctuations, Pattern Recognition, Vol. (35), 2002, pp. 1447-1462.

[30] W. Huyer, A. Neumaier, Global Optimization by Multilevel Coordinate Search, *Journal of Global Optimization*, Vol. 14, 1999, pp. 331-355.

[31] Y. Kumar, S. K. Srivastava, A. K. Bajpai, N. Kumar, Development of CAD Algorithms for Bezier Curves/Surfaces Independent of Operating System, *WSEAS Transactions on Computers*, Vol. 11, No. 6, 2012, pp. 159-169.

[32] K. Thanushkodi, K. Deeba, On Performance Analysis of Hybrid Intelligent Algorithms (Improved PSO with SA and Improved PSO with AIS) with GA, PSO for Multiprocessor Job Scheduling, *WSEAS Transactions on Computers*, Vol. 11, No. 5, 2012, pp. 159-169.