INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

VISUALIZATION WITH SPLINE USING A GENETIC ALGORITHM

BY

SYED ARSHAD RAZA

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

MAY 2001

UMI Number: 1404203

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by SYED ARSHAD RAZA under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Deanship of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE.

THESIS COMMITTEE

Dr. Muhammad Sarfraz (Chairman)

Dr. Muhammad Al-Suwaiyel (Member)

Dr. Moataz Ahmad (Member)

Dr. Kalim Uddin Qureshi (Member)

Dr. AbulFazal Mohammad Arif (Member)

Department Chairman

Prof. Osama A. Jannadi
Dean of Graduate Studies

13/6/2001
Date

Dedicated to

My Parents

and

My

Grand Mother

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful

All praise and glory to Almighty Allah (SWT) who gave me courage and patience to carry out this work. Peace and blessing of Allah be upon His last Prophet Muhammad (Sallulaho-Alaihe-Wassalam) and all his Sahaba (Razi-Allaho-Anhum) who devoted their lives for the spread of Islam.

Acknowledgement is due to King Fahd University of Petroleum and Minerals for providing support for this work.

My deep appreciation goes to my thesis advisor Dr. Muhammad Sarfraz for his constant help, guidance and the countless hours of attention he devoted throughout the course of this research work. His priceless suggestions made this work interesting and learning for me.

Thanks are due to my thesis committee members Dr. Mohammad Al-Suwaiyel, Dr. Moataz Ahmad, Dr. Kalim Uddin Qureshi and Dr. AbulFazal Mohammad Arif

for their interest, invaluable cooperation and support.

Acknowledgement is due to Shafayat Abrar, Murtaza Ali Khan, Mirza Naveed Baig and Atiq Waliullah who helped me at various stages of this work.

Special thanks are due to my house mates and colleagues and my friends for their help and encouragement. A few of them are Noman Khan, Zeeshan Jilani, Amin-uddin, Hamid, Asif Iqbal, Sarfraz, Khurram Mohib, Mir Asif, Mofazzal, Rehan, Irfan-ul-Haque, Owais Malik, Zafar Shazli, Khurshid, Tariq, Fareed, Saqib Adeeb, Ahmer Shafi and Masroor. They have made my work and stay at KFUPM very pleasant.

And finally, my heartfelt thanks to my parents, my grand mother , my uncle and aunty, my sister, brother in law, my niece and other family members. Their prayers and support are always with me.

Contents

List of Tables

List of Figures

THESIS ABSTRACT

Name: Syed Arshad Raza

Title: Visualization with Spline using a Genetic Algorithm

Degree: MASTER OF SCIENCE

Major Field: Computer Science

Date of Degree: May 2001

If we have to obtain a good spline model from large measurement data, we frequently have to deal with knots as variables, which becomes a continuous, non-linear and multivariate optimization problem with many local optima. Hence, it is very difficult to obtain a global optima. In this thesis, we present a method to convert the original problem into a discrete combinatorial optimization problem and solve the converted ones by a genetic algorithm.

In our work, we have used NURBS for our spline model. The optimization of both the knots and the weights corresponding to the control points has been done. The work has been applied to both curves and surfaces. In case of curves, we have also incorporated a corner detection algorithm to detect significant points which are necessary to capture a pleasant looking spline fitting for shapes such as fonts. A parametric B-Spline has been approximated to various characters and symbols. The chromosomes have been constructed by considering the candidates of the locations of knots as genes. The best model among the candidates is searched by using Akaike's Information Criterion(AIC). The method determines the appropriate number and location of knots automatically and simultaneously. Some examples are given to show the results obtained from the algorithm.

King Fahd University of Petroleum and Minerals, Dhahran.
May 2001

خلاصة الرسالة

الاســـــــــم : سيد أرشد رضا
عنوان الرسالة : الرؤية بواسطة الشرائح الخطية واستعمال الخوارزميات الجينية
الدرجة العلمية : ماجستير العلوم
التخصـــــص : علوم الحاسب الآلي
التاريـــــــخ : مايو ٢٠٠١م

إذا أردنا الحصول على نموذج شرائح خطية جيد من بيانات قياسات كبيرة فإننا كثيراً ما نحتاج للتعامل مع العقد كمتغيرات مما ينتج عنه مسألة إيجاد الحل الأمثل لدالة غير خطية مستمرة عديدة المتغيرات ذات نقاط مثلى محلية عديدة. لذلك فإنه يصعب الحصول على نقطة مثلى عامة. تعرض هذه الرسالة طريقة لتحويل المسألة الأصلية إلى مسألة إيجاد الحل الأمثل لتوافقية متقطعة وحل المسألة باستعمال خوارزمية جينية.

في هذا البحث تم استخدام NURB لنموذج الشرائح الخطية وتم إيجاد الحل الأمثل لكل من العقد وأوزان نقاط التحكم المقابلة.

تم تطبيق هذا العمل على كل المنحنيات والسطوح. في حالة المنحنيات تم إدخال خوارزمية كشف الأركان لتكشف النقاط الهامة في تحديد شرائح خطية تجعل بعض الأشكال تبدو بمظهر حسن. معامل الشرائح الخطية B تم تعريفه للعديد من الأحرف والرموز. تم بناء الكروموسومات باعتبار مواقع العقد على أنها جينات. تم البحث عن أفضل نموذج بطريقة (AIC). هذه الطريقة تحدد عدد ومواقع العقد الأنسب بطريقة أتوماتيكية آنية. أعطيت بعض الأمثلة لعرض النتائج التي تم الحصول عليها من الخوارزمية.

جامعة الملك فهد للبترول والمعادن
الظهران
مايو ٢٠٠١م

Chapter 1

Introduction

1.1 Problem

There are various well-known applications that require curve fitting which is the term used to refer collectively to interpolation and approximation. The problem of interpolation is finding the curve that passes through a set of points, whereas the problem of approximation is finding a curve that passes near a set of points. Curve fitting problems occur in signal processing, graphics, statistical analysis and in geometric modeling.

There are number of applications where finding a mathematical curve description of the desired shape is beneficial e.g. design of automobiles, ships, aircrafts, equipment and of course fonts. Data Visualization is also a very important and emerging area in the field of computer graphics. Typically, there exists an array

of data that is to represent mathematically by some analytical function. The data received may be noisy or noise less, uniform or non-uniform. In this technique, it is tried to approximate the underlying function as closely as possible for a better data interpretation.

Polynomials are usually a simple but rather poor choice. A better choice is to use one of many good piecewise polynomial functions or splines. A piecewise approximation consists of a number of single cubic pieces connected end to end. The position and properties of connection points known as knots determine the type of spline. One advantage of such a representation is local control, which is the ability to modify a data point and only have a small region of the curve affected. Another important consideration is that such representations are more numerically stable and computationally efficient since they require a lower degree polynomial for the same fit performance.

B-splines have many desirable properties such as completeness, compactness, minimal support, variation diminishing, transformation independence and the existence of many computationally attractive algorithms.

The proposed work suggested the use of genetic algorithm for data fitting for visualization, using NURBS (Non-Uniform Rational B-Splines). A detailed discussion on NURBS has been given in chapter 2. The work was carried out in TWO phases:

Phase-1: consisted of implementing the spline's parametric form for open and

closed curves. The implemented work was then tested with the following special cases.

- Fonts
- 2D Data Visualization

Phase-2: consisted of implementing NURBS for surfaces. The implemented work was then tested for :

- 3D Data Visualization

1.2 Why GA ?

Data fitting with spline is one of the important technologies for geometric modeling in shape design. For example, in the reverse geometric modeling of a car design, we have to make a good model from many measurement data. Since the shape of the underlying function data is frequently complicated, it is difficult to approximate it by a single polynomial. In this case, a spline is one of the most appropriate approximating functions.

It is well known, however, that the key to using a spline is the determination of good knots, and in order to obtain good approximation, we have to place the knots as precisely as possible [5, 25]. In order to obtain a good spline model from many measurement data, frequently we have to deal with knots as variables. Then the

problem to be solved becomes a continuous nonlinear and multivariate optimization problem with many local optima. Therefore, it is difficult to obtain a global optimum.

1.2.1 Earlier Methods' Limitations

The earlier methods mentioned in the literature survey have the following limitations:

They:

- were not automated.
- gave redundant knots [22].

needed:

- Error tolerance or smoothing factor to be determined [5, 18].
- Good initial location of knots, which is not easy to determine [11].

1.2.2 Application of GA and its Advantages

Recently *Genetic Algorithms* (GAs) (A brief introduction to GAs has also been given in the chapter 4) have been widely reckoned as a useful vehicle for obtaining high quality or even optimal solutions in this area. However, the knots of a spline don't

need to be optimal: usually sub optimal is sufficient. So, GAs can be conveniently applied for the determination of good knots.

The paper titled *"Automatic Knot Placement by a Genetic Algorithm for Data Fitting with a Spline"*[34] presents a GA-based algorithm with an objective of simultaneous determination of the number of knots and their locations. In the algorithm, the individuals are constructed by considering the candidates of the locations of knots as genes. The fitness of the solutions in each generation is examined by *Akaike's Information Criterion* (AIC) [1]. The number of knots in a solution is controlled by a control parameter knot ratio R.

Advantages:

- can automatically determine appropriate number and location of knots without requiring their good initial locations.
- does not give redundant knots.
- does not require any error tolerance or smoothing factor to be determined.

1.3 Objectives

This study aims to:

1. Approximate any function whose data may be noisy.
2. Find the optimal number and location of knots.

3. Approximate the fonts outline.

4. Approximate a surface which may possibly contain noise.

5. Automation of all above mentioned tasks.

1.4 Literature Survey

The literature survey of our study was tri-directional in a sense that we had to search for:

- Earlier work on data fitting with splines in general and NURBS in particular.
- Prevailing work incorporating GAs in computer graphics.
- Material regarding GA basics for its proper understanding and application.

1.4.1 Data Fitting with Splines

The problem of curve and surface fitting using B-splines has been addressed by [13]. B-splines are particularly attractive interpolants due to such properties as optimal smoothness, variation diminishing, local control, convex hull and existence of good evaluation algorithms. This work is new and explores a geometric approach from a signal processing perspective. The geometric approach studied here is to start with a straight-line approximation to the given data (which corresponds to multiple *knots*

at each point in the B-spline representation). The *knots* are the positions where the piecewise polynomial pieces meet and in interpolation are the given data points.

The key to using a spline is the determination of good knots [5, 25] and in order to obtain good approximation, we have to place the knots as precisely as possible. In order to obtain a good spline model from many measurement data, frequently we have to deal with knots as variables. However, the knots of a spline don't need to be optimal: usually sub-optimal is sufficient [25]. Many methods of knot determination have been proposed.

A method for multi curve approximation with B-spline has been presented by [23]. The approximation is formulated as a constrained optimization problem with the least squares error as the objective function and the knot vector as variables.

Some methods of determining good knots have been proposed by [22, 5, 17, 11].

As stated earlier, the method by [22], in general, gives redundant knots.

Methods proposed by [5, 17] need error tolerance to be determined subjectively in many cases and the method by [11] needs good initial location of knots, which is not easy to determine.

Therefore, it can be concluded that the methods presented by [22, 5, 17, 11] are not sufficient enough for obtaining a good model of data automatically. Here, *'good model'* means that the number of parameters in the model is as small as possible and the difference between the model and the underlying function of data is as small as possible. The next section addresses this objective to be achieved from the GAs

point of view.

1.4.2 Data fitting, focused on NURBS

The major ingredients of NURBS are Non Uniform knots, Rational and B Splines. Rational curve and surfaces was first introduced into computer graphics literature by Coons around 1967. He also suggested their usage to represent conic sections. Forrest gives a rigorous treatment of rational conics and cubics.

Following the work done by De Boor, Cox and Riesenfeld, Versprille , in 1975 proposed the Non uniform Rational B splines or NURBS. After that , Piegl and Tiller formed the basis of the current implementation. The NURBS shape representation for geometric design generalized Riesenfeld's B Splines. NURBS quickly gained popularity and were incorporated into several commercial modeling systems like IEGS, PHIGS+ , Product Data Exchange Specification , International Standard Office Standard for the Exchange of Product Model Data [26].

It offers a unified mathematical formulation for representing not only free-form curves and surfaces, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution. By adjusting the positions of control points and manipulating associated weights, one can design a large variety of shapes using NURBS. A matrix representation for NURBS curves and surfaces has been described by [9]. They represent the matrix form for NURB by straightforward algebraic manipulation by using Bohem's knot insertion algorithm instead of Deboor. A NURB curve of

degree *'d'*, the basic handles are control points, weights and knots. The method first perform a linear transformation between t (knots) and $u[0,1]$ by using a normalized parameter

After having a good flavor of design paradigm of control points oriented algorithms, a new alternative Object Oriented paradigm is presented by [2]. The Object oriented paradigm requires a central constructor *evaluator*, for generating the control points and derivatives for a given mapping. Computing control points is an classical approximation. Following the object oriented design principles of data hiding, the defining curves (private) control points are only accessed by their homogeneous evaluators and the approximation procedures does not know about the ruling curves. A theoretically optimal solution for this is produced by metaalgorithm.

As NURBS have a number of control handlers like control points, knot vectors, weights etc, it is some time confusing for a designer to select the parameter correctly by varying which, he got the desired shape. It is one of the most important issues in CAGD. Piegl[21] discussed several handles to modify the shape of NURBS directly from their mathematical definition i.e. recomputing the weights or control points.

In [27], a simple tool addresses the above mentioned problem , based on the well know displacement of control points along a user defined direction. It consists of a perspective functional transformation of arbitrary origin O. The extra freedom provided by the weights in rational form is controlled in a geometric way without any numerical input. The displacement of several control points , keeping a common

center O, can manipulate NURBS in ways that are simply impossible to achieve in integral form. This tool effectively employs the added flexibility provided by weights. By varying weights, we got a push / pull in the curve towards/ away from the control polygon. They also consider the case of involving several control points in perspective functional transformation.

Piegel and Tiller[15] discuss NURBS very much in detail in their comprehensive book about NURBS. They discuss all types of B-spline curves with special consideration to NURBS.

1.4.3 Prevailing Work incorporating GAs

In [18], first a general framework is setup for the application of genetic algorithms in curve design. Then, within this scheme, the problem of spline interpolation—a frequently used method for representing complex geometrical shapes in CAD/CAM systems–is dealt with. While the method is simple and robust, it suffers from the drawback that some parameters must be given that are needed for the mathematical description but are not closely related to the geometrical input data of the object.

Any subjective parameter such as error tolerance or a smoothing factor and good initial location of knots for iterative search is not required by [34]. In this paper Akaike's Information Criterion [1] has been used as a function measure, which take care of the sum of square error between the actual data and the fitted ones and the number of knots at the same time. The paper presents a non-parametric represen-

tation as a model function. The parametric representation has been discussed in [33].

1.4.4 Material regarding GA Basics

A good look inside GA basics and their details is provided by [8, 20, 7, 19]. However, [18, 19] also discuss the GA basics in brief.

1.5 Thesis Organization

Chapter 2 is about Non-uniform Rational B-Splines (NURBS). It covers the theoretical details and the properties of NURBS which make them superior to the other parametric curves. Chapter 3 addresses corner detection in brief. Chapter 4 presents a brief introduction to GAs. It discusses some basic concepts involved in this optimization technique. Chapter 5 is the most important chapter, which puts light on the implementation details of the suggested system. It covers the data fitting scheme used both for curves and surfaces. Chapter 6 presents some critical analysis of the implemented system. Chapter 7 concludes our work and gives suggestions for possible future work.

Chapter 2

NURBS

2.1 Introduction

The major ingredients of NURBS are Non Uniform knots, Rational and B Splines. Rational curve and surfaces was first introduced into computer graphics literature by Coons around 1967. He also suggested their usage to represent conic sections. Forrest gives a rigorous treatment of rational conics and cubics. Following the work done by De Boor, Cox and Riesenfeld, Versprille , in 1975 proposed the Non uniform Rational B splines or NURBS. After that, Piegl and Tiller form the basis of the current implementation. The NURBS shape representation for geometric design generalized Riesenfeld's B-Splines. NURBS quickly gained popularity and were incorporated into several commercial modeling systems like IEGS, PHIGS+, Product Data Exchange Specification , International Standard Office Standard for the

Exchange of Product Model Data[26]. It offers a unified mathematical formulation for representing not only free-form curves and surfaces, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution. By adjusting the positions of control points and manipulating associated weights, one can design a large variety of shapes using NURBS.

NURBS are defined on a knot vector where the interior knot spans are not equal. As an example, we may have interior knots with spans of zero. Some common curves require this type of non-uniform knot spacing. The use of this option allows better shape control and the ability to model a larger class of shapes.

2.2 Theoretical Details

Given a set of $n+1$ control points $P_1, P_2, ..., P_n$, each of which is associated with a non-negative weight w_i (i.e., P_i has weight $w_i \geq 0$), and a knot vector $t = \{t_0, t_1, ..., t_m\}$ of $m+1$ knots, the degree p NURBS curve is defined as follows:

$$C(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(t) w_i} \qquad a \leq t \leq b \tag{2.1}$$

where P_i's are the *control points* (forming a *control polygon*), the w_i are the weights, and $N_{i,p}(t)$are the pth-degree NURBS basis functions defined on the non-

periodic (and nonuniform) knot vector

$$U = \{\underbrace{a, \ldots\ldots, a}_{p+1}, u_{p+1}, \ldots\ldots, u_{m-p-1}, \underbrace{b, \ldots\ldots, b}_{p+1}\}. \tag{2.2}$$

Unless otherwise stated, we assume that $a = 0, b = 1$, and $w_i > 0$ for all i.

Setting

$$R_{i,p}(t) = \frac{N_{i,p}(t)w_i}{\sum_{j=0}^{n} N_{j,p}(t)w_j} \tag{2.3}$$

allows us to rewrite Eq.(3) in the form

$$C(t) = \sum_{i=0}^{n} R_{i,p}(t)P_i \tag{2.4}$$

The $R_{i,p}(t)P_i$ are the *rational basis functions;* they are piecewise rational functions on $t \in [0, 1]$.

For the i^{th} normalized B-spline basis function or order k (degree $k-1$) is defined

by the Cox-deBoor recursion formulas.

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1}; \\ 0 & \textit{otherwise.} \end{cases}$$

and

$$N_{i,p}(t) = \frac{(t - t_i)N_{i,p-1}(t)}{t_{i+p-1} - t_i} + \frac{(t_{i+p} - t)N_{i+1,p-1}(t)}{t_{i+p} - t_{i+1}} \tag{2.5}$$

2.3 Basis Functions

The function $N_{i,p}(t)$, which determines how strongly control point P_i influences the curve at time t, is called the basis function for that control point. There are two interesting properties of Basis Functions, namely:

- The domain is subdivided by the so called knots.
- Basis functions are not non-zero on the entire interval.

In fact, each B-spline basis function is non-zero on a few adjacent subintervals and, as a result, B-spline basis functions are quite *"local"*.

Figure 2.1 shows the B-spline basis functions.

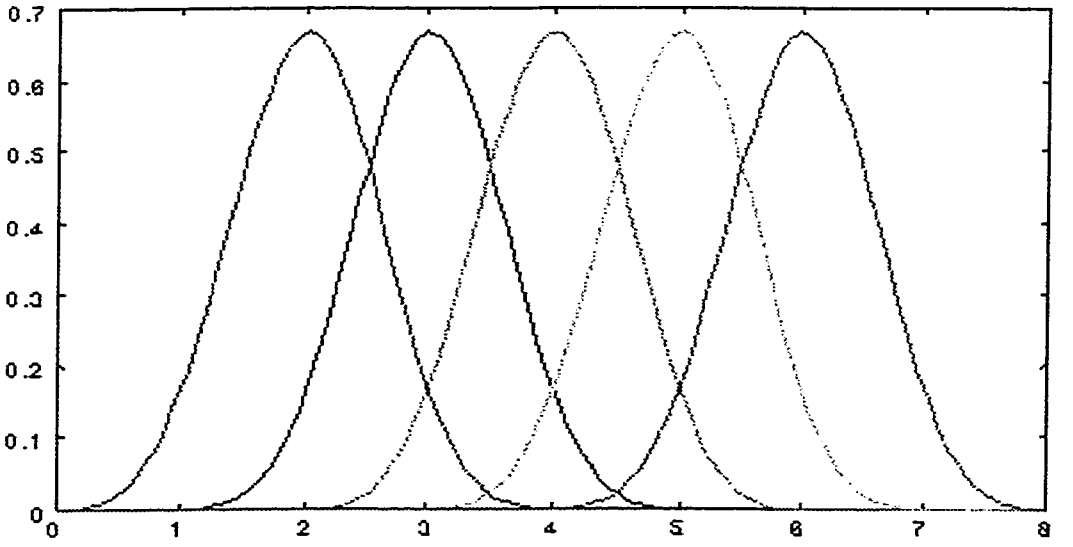

Figure 2.1: B-spline Basis Functions

2.3.1 Important Properties of NURBS Basis Functions

Since NURBS is the generalization of B-spline, it enjoys all the properties of B-spline. Following are some of the main properties of NURBS Basis Functions.

- $R_{i,p}(t)$ is a degree p rational function in t.
- *Non-negativity*- for all i, P and t, $R_{i,p}(t)$ is non-negative.
- it local Support- $N_{i,p}(t)$ is a non-zero polynomial on $[t_i, t_{i+p+1}]$.
- Since $N_{i,p}(t)$ is non-zero on $[t_i, t_{i+p+1}]$, so does $R_{i,p}(t)$. Note that we assume $w_i s$ are non-negative.
- On any span $[t_i, t_{i+1}]$, at most $p+1$ degree p basis functions are non-zero, namely: $N_{i-1,p}(t), N_{i-p+1,p}(t), N_{i-p+2,p}(t), \ldots$. and $N_{i,p}(t)$.
- *Partition of Unity*- The sum of all non-zero degree p basis functions on span $[t_i, t_{i+1}]$ is 1.
- If the number of knots is $m+1$, the degree of the basis functions is p, and the number of degree p basis functions is $n+1$, then $m = n+p+1$.
- At a knot multiplicity of k, basis function $N_{i,p}(t)$ is C^{p-k} continuous. Therefore, increasing multiplicity decreases the level of continuity, and increasing degree increases continuity.

- Basis function $R_{i,p}(t)$ is a composite curve of degree p rational functions with joining points at knots in $[t_i, t_{i+p+1}]$.

- If $w_i = c$ for all i, where c is a non-zero constant, $R_{i,p}(t) = N_{i,p}(t)$.

- Therefore, B-spline basis functions are special cases of NURBS basis functions when all weights become a non-zero constant. We have mentioned the special case of $c = 1$.

2.4 Important Properties of NURBS Curves

The following are the important properties of NURBS curves.

- NURBS curve $C(t)$ is a piecewise curve with each component a degree p rational curve.

- Equality $m = n + p + 1$ must be satisfied.

- A clamped NURBS curve $C(t)$ passes through two end control points p_0 and p_n.

- *Strong Convex Hull Property*: The NURBS curve is contained in the convex hull of its control polyline. More over, if t is in knot span $[t_i, t_{i+1}]$, then $p(t)$ is in the convex hull of control points $p_{i-p}, p_{i-p+1}, ..., p_i$.

- *Local Modification Scheme:* Changing the position of control point p_i only affects the curve $p(t)$ on interval $[t_i, t_{i+1}]$. This local modification scheme is very important to curve design, because we can modify a curve locally without changing the shape in a global way.

- $p(t)$ is C^{p-k} continuous at a knot of multiplicity k.

- *Variation Diminishing Property:* The variation diminishing property also holds for NURBS curves. If the curve is contained in a plane (resp., space), this mean no straight line (resp., plane) intersects a NURBS curve more than it intersects the curve's control polyline (see Figure 2.2).

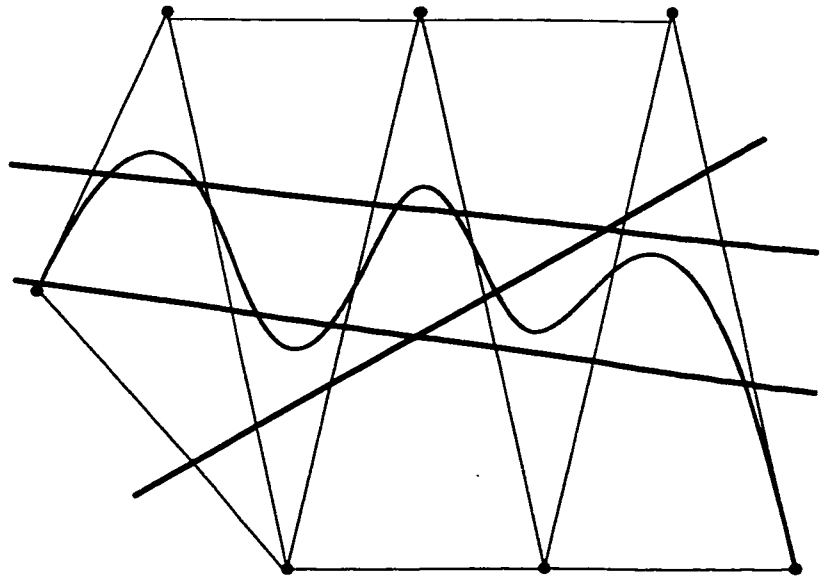

Figure 2.2: Variation Diminishing Property in NURBS

- B-spline Curves and Bezier are special cases of NURBS Curves.

- If all weights are equal, a NURBS curve becomes a B-spline curve. If further more $n = p$ (i.e., the degree of a B-spline curve is equal to n, the number of control points minus 1) and there are $2(p+1) = 2(n+1)$ knots with $p+1$ of them clamped at each end, this NURBS curve reduces to a Bezier curve.

- *Projective Invariance* If a projectivev transformation is applied to a NURBS curve, the result can be constructed from the projective images of its control points. This is a nice property. When we want to apply a geometric or even projective transformation to a NURBS curve, this property guarantees that we can apply transformation to control points, which is quite easy, and once the transformed control points are available the transformed NURBS curve is the one defined by these transformed points. Therefore, we do not have to transform the curve. On the other hand, Bezier curves and B-spline curves only satisfy the affine invariance property. This is because only NURBS curves involve projective transformations.

2.5 Effect of Weights' Modification on NURBS

Since NURBS curves contain B-spline curves as special cases, methods for modifying the shape of a B-spline curve such as moving control points and modifying knot vector also work for NURBS. NURBS curves are defined with a set of control points, a knot vector, a degree and a set of weights, we have on e more parameter for

modifying the shape of a NURBS curve, the *weights*.

The basis functions of a NURBS curve has been shown in Fig.2.1. Therefore, increasing and decreasing the value of w_i will increase and decrease the value of $R_{i,p}(t)$, respectively. By changing the weight w_i of a control point P_i affects only the range $[t_i, t_{i+p+1}]$. More precisely, increasing the value of w_i will pull the curve toward the control point P_i . In fact all affected points on the curve will also be pulled in the direction of P_i. When w_i approaches infinity, the curve will pass through control point P_i. On the other hand, decreasing the value of w_i will push the curve away form control point P_i.

A Simple NURBS curve is shown in Figure 2.3 and Figure 2.4 shows NURBS curve with varying weights at a control point.

In summary, we can say that *"Increasing (resp., decreasing) the value of weight w_i pulls (resp., pushes) the curve toward (resp., away from) control point P_i. When the value of of w_i becomes infinity, the curve passes through control point P_i and when w_i is zero, control point P_i does not have impact on the curve.*

Non-uniform Rational B-splines(NURBS) are useful for two reasons. The first and the most important reason is that they are invariant under perspective transformations of the control points. The second advantage of rational splines is that they define precisely any of the conic sections. A conic section can only be approximated with rational, by using many control points close to the conic.

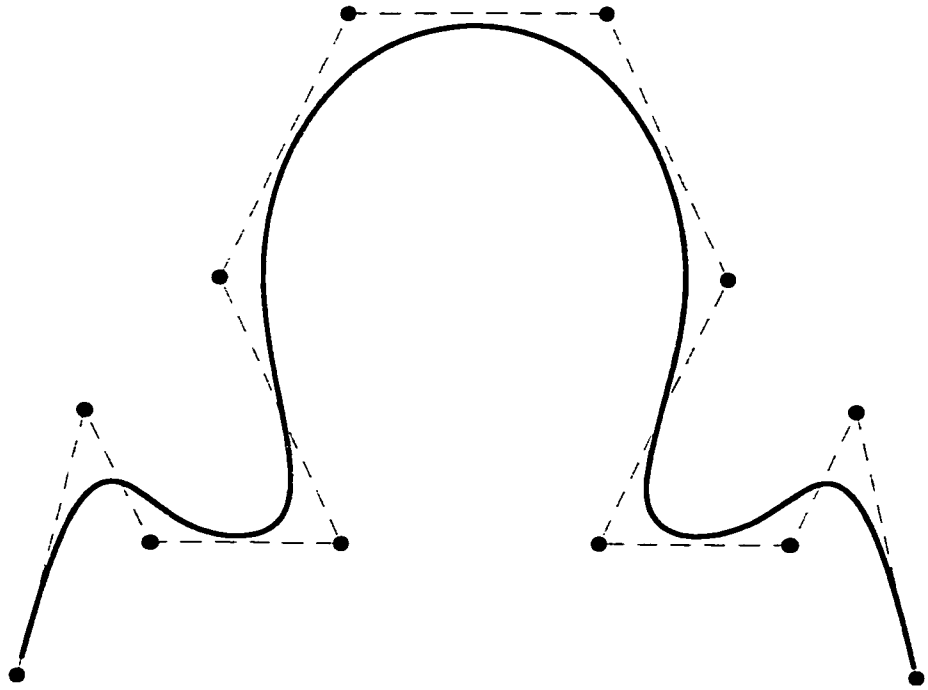

Figure 2.3: NURBS Curve with Default Shape Parameters.

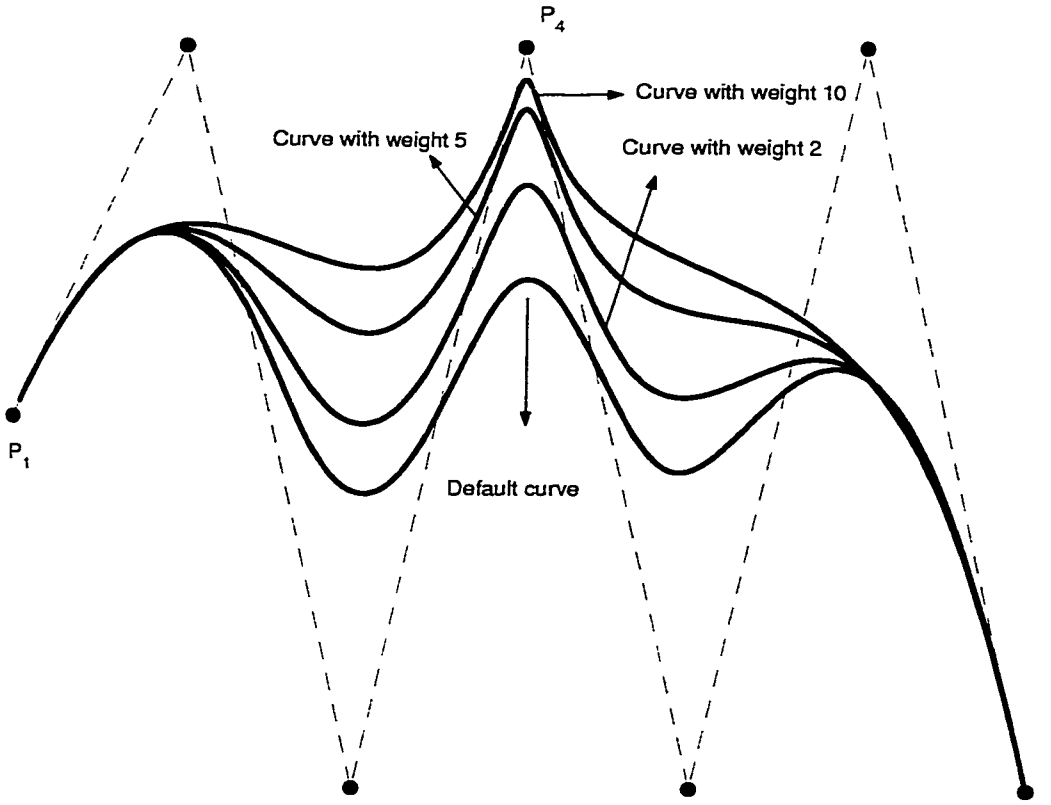

Figure 2.4: Application of Different Weight Values on P_4 and its Effect on the Shape of the Curve.

Chapter 3

Corner Detection

Corners in digital images provide valuable information for shape representation and analysis. They provide important features for object recognition, shape representation, image interpretation and motion analysis [10, 4, 6]. Therefore, if these characteristic contour points are identified properly then a shape can be represented accurately and efficiently.

3.1 Corner Detection in Planar Curves

Corner detection is related to detection of high curvature points in planar curves. Various corner detection algorithms have been developed. Frequently cited approaches of corner detection are discussed and compared by [16] and [3]. For images a corner detection algorithm, based on the property of corners that the change of

image intensity should be high in all directions is described by [32].

What is corner? There is no generally accepted mathematical definition of corner. Rather it is intuitively understandable. Since we used curvature measure to detect corner, therefore in our case we set some threshold value of angle to declare a point as corner point.

In our font design method, corner points are the end control points (P_0 and P_3) of cubic Bezier curve. If corner points are detected accurately in early stage then computation will be minimize in the later stages.

A curve consists of sequence of points $p_i = (x_i, y_i), i = 1, 2, \ldots, n$. For each point its cornerness (measure of corner strength) is determined. Points whose cornerness is above predefined threshold are declared as corner points. When processing a point p_i, the algorithm considers a number of subsequent points (p_k^+) and previous points (p_k^-) in the sequence, as candidates for potential corner points.

3.2 Corner Detection Algorithm

The algorithm we used is a modified form of [3]. Corner point is defined as a point where triangle of specified angle can be inscribed within specified distance from its neighbor points. The number of neighbor points to be checked are also predefined.

First Pass: For each point p_i it is checked if triangle of specified size and angle

is inscribed or not. Following three conditions are used.

$$d_{min}^2 \leq |p - p_k^+|^2 \leq d_{max}^2 \tag{3.1}$$

$$d_{min}^2 \leq |p - p_k^-|^2 \leq d_{max}^2 \tag{3.2}$$

$$\alpha \leq \alpha_{max} \tag{3.3}$$

where

p is point under consideration for corner point.

p_k^+ is the k^{th} clockwise neighbor of p.

p_k^- is the k^{th} anti-clockwise neighbor of p.

Taking

$a = |p - p_k^+|$ $\quad$ $b = |p - p_k^-|$ $\quad$ $c = |p_k^+ - p_k^-|$

The angle α can be computed by using cosine law

$$a^2 + b^2 - c^2 - 2ab\,cos\alpha = 0 \tag{3.4}$$

$$\alpha = cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \tag{3.5}$$

All the three conditions described in equations (3.1), (3.2) and (3.3) are necessary for first pass. Now each point p may have zero, one or more than one alpha values. Among all alpha values, minimum value is taken as the alpha value of that point p.

Second Pass: The corner detection algorithm can detect adjacent points as corner. Because we specify threshold value of alpha and corner points as well as points neighbor to corners points can have value less than alpha. Second pass removes such points. If the difference of contour indices of corner points cp_i and cp_{i+1} is less than IndexLimit then the corner point whose alpha is high is removed from the candidate corner list. It means that if corner point cp_i has contour index i and corner point cp_{i+1} has contour index j then $(j - i \geq$IndexLimit).

The procedure of detecting corner points is given in the pseudocodes shown in Figures 3.1 and 3.2.

```
Input: Points(p1, p2, ..., pn) where pi=(px, py)
Output: Corner Points (cp1, cp2, ..., cpm) where cpi=(cpx, cpy)
alphaCount ← 0
dmin ←3
dmax ←dmin+3
neighbourMax← 8
αmax ← 150
IndexLimit ← 10

/* First Pass */
for p1Index ←1 to n
    for neighbourIndex←1 to neighbourMax
        if ((p1Index+ neighbourIndex)> n)
            p+Index← p1Index+ neighbourIndex-n
        else
            p+Index← p1Index + neighbourIndex
        end
        if ((p1Index- neighbourIndex)<=0)
            p-Index←n- neighbourIndex+ p1Index
        else
            p-Index← p1Index - neighbourIndex
        end
        ds_pp+ ← ( px[p1Index]- px[p+Index] )2 + ( py[p1Index]- py[p+Index] )2
        ds_pp- ← ( px[p1Index]- px[p-Index] )2 + ( py[p1Index]- py[p-Index] )2
        ds_p+p- ← ( px[p+Index]- px[p-Index] )2 + ( py[p+Index]- py[p-Index] )2
        a ←sqrt(ds_pp+)
        b ←sqrt(ds_pp-)
        c ←sqrt(ds_p+p-)
        α ← cos-1 [(a2+b2-c2)/2ab]
        if ( ((ds_pp+ ≥ dmin) & (ds_pp+≤ dmax)) &
             ((ds_pp- ≥ dmin) & (ds_pp-≤ dmax)) &
             (α ≤ αmax ) )
             alphaCount← alphaCount+1
             α[p1Index, alphaCount]← α
        endif
    endfor // End of neighbourIndex loop
        if (alphaCount > 0 )
           pass1Count← pass1Count+1 // count of first pass candidate corners
           // FindMinIndex function finds index of minimum α
           minIndex←FindMinIndex (α[p1Index, 1]... α[p1Index, alphaCount] )
           αmin [pass1Count]← α[p1Index, minIndex]
           cpIndex[pass1Count ]← p1Index //  index of original contour points
           cpx[pass1Count]← px[p1Index] // corner points after first pass
           cpy[pass1Count]← py[p1Index] // corner points after first pass
        end
end // End of p1Index loop
```

Figure 3.1: First Pass of Corneer Detection Algorithm

```
/* Second Pass */

/* To remove very close corner points. Closeness is measured w.r.t contour index */
pass2Count← pass1Count
p2Index←2
while (p2Index<= pass2Count)
      if (cpIndex[p2Index] - cpIndex[p2Index-1] <IndexLimit)
        if (α_min [p2Index-1] > α_min [p2Index] ) // if true then replace
            α_min [p2Index-1]← α_min [p2Index]  // (p2Index-1)th record
            cpIndex[p2Index-1]← cpIndex[p2Index] // with
            cp_x[p2Index-1]← cp_x[p2Index]  // p2Index th record
            cp_y[p2Index-1]← cp_y[p2Index]
         end
         for shiftIndex←p2Index to pass2Count-1 // shift
            α_min [shiftIndex]← α_min [shiftIndex+1] // (p2Index+1) to pass2Count
            cpIndex[shiftIndex]← cpIndex[shiftIndex+1] //records
            cp_x[shiftIndex]← cp_x[shiftIndex+1] // one position up
            cp_y[shiftIndex]← cp_y[shiftIndex+1]
         end
         pass2Count←pass2Count-1  // last record in no longer corner record
         p2Index←p2Index-1
      end
      p2Index←p2Index+1 // proceed to next record
end
if (n- cpIndex[pass2Count]+ cpIndex[1] < IndexLimit) // check first and last records
    if  (α_min [1] > α_min [pass2Count]) // if true replace 1st record with 2nd record
        for shiftIndex←1 to pass2Count-1 // shift
        α_min [shiftIndex]← α_min [shiftIndex+1] // (1+1)th to pass2Count
        cpIndex[shiftIndex]← cpIndex[shiftIndex+1] //records
        cp_x[shiftIndex]← cp_x[shiftIndex+1] // one position up
        cp_y[shiftIndex]← cp_y[shiftIndex+1]
    end
  pass2Count←pass2Count-1  // last record in no longer corner record
end

Corners( (cp_x[1], cp_y[1]), (cp_x[2], cp_y[2]),...,( cp_x[pass2Count], cp_y[pass2Count]))
```

Figure 3.2: Second Pass of Corner Detection Algorithm

Chapter 4

A Brief Introduction to GA Concepts

4.1 History

The idea of evolutionary computing was introduced in the 1960s by I. Rechenberg in his work *"Evolution strategies"* (Evolutionsstrategie in original). His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. This lead to Holland's book *"Adaption in Natural and Artificial Systems"* published in 1975. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method *"Genetic Programming"* (GP). LISP programs were used, because programs in this language can expressed in the form of a *"parse tree"*,

which is the object the GA works on.

4.2 Biological Background

Chromosome: All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serves as a model for the whole organism. A chromosome consist of genes, blocks of DNA (see Figure 4.1). Each gene encodes a particular protein. Basically can be said, that each gene encodes a trait, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called alleles. Each gene has its own position in the chromosome. This position is called locus. Complete set of genetic material (all chromosomes) is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype, its physical and mental characteristics, such as eye color, intelligence etc.

Reproduction: During reproduction, first occurs recombination (or crossover). Genes from parents form in some way the whole new chromosome. The new created offspring can then be mutated. Mutation means, that the elements of DNA are a bit changed. This changes are mainly caused by errors in copying genes from parents. The fitness of an organism is measured by success of the organism in its life.

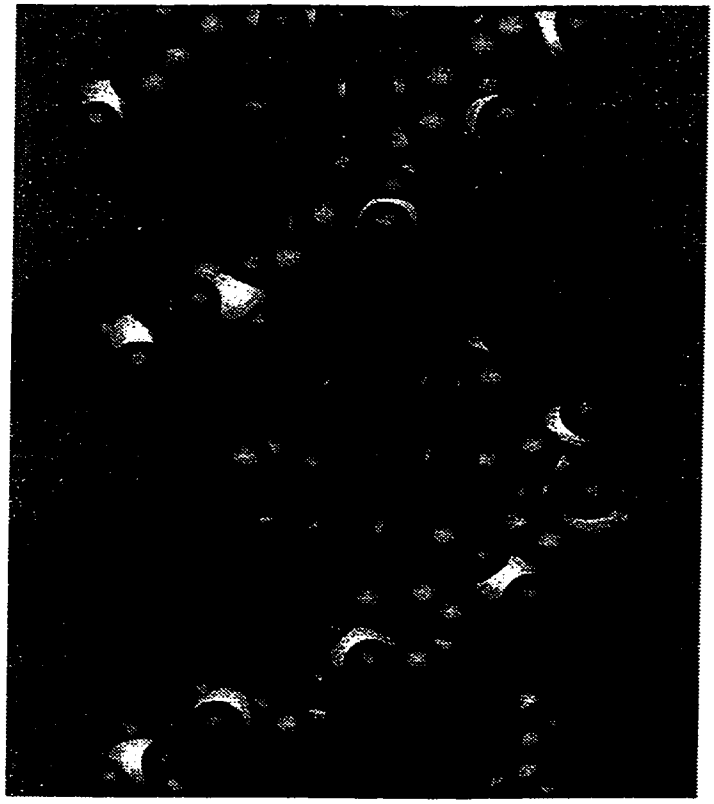

Figure 4.1: The DNA

4.3 Search Space

If we are solving some problem, we are usually looking for some solution, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution is) is called search space (also state space). Each point in the search space represent one feasible solution. Each feasible solution can be "marked" by its value or fitness for the problem. We are looking for our solution, which is one point (or more) among feasible solutions - that is one point in the search space. The looking for a solution is then equal to a looking for some extreme

(minimum or maximum) in the search space. The search space can be whole known by the time of solving a problem, but usually we know only a few points from it and we are generating other points as the process of finding solution continues.

The problem is that the search can be very complicated. One does not know where to look for the solution and where to start. There are many methods, how to find some suitable solution (ie. not necessarily the best solution), for example hill climbing, tabu search, simulated annealing and genetic algorithm. The solution found by this methods is often considered as a good solution, because it is not often possible to prove what is the real optimum.

4.4 The Genetic Algorithms

In this section, a brief overview of standard genetic algorithms (SGA) has been presented. In the last decade, genetic algorithms have emerged as practical robust optimization search methods. Genetic algorithms (GAs), introduced by Holland in the 1970s, are search techniques that are designed based on the concept of evolution. In simple terms, given a well-defined search space in which each solution is represented by a bit string, called a chromosome, a GA is applied with its three genetic search operators— selection, crossover and mutation—to transform a population of chromosomes with the objective of improving the quality of the chromosomes. The individual bits of a chromosome are called genes. A detailed mechanism of the three

operators will be discussed in the coming sections.

A GA is usually employed to determine the optimal solution of a specific objective function. The search space, therefore, is defined as the solution space so that each feasible solution is represented by a distinct chromosome. Before the search starts, a set of chromosomes is randomly chosen from the search space to form the initial population. The three genetic search operations are then applied one after the other to obtain a new generation of chromosomes in which the expected quality over all the chromosomes is better than that of the previous generation. Thus the process is repeated until stopping criterion is met (for example, a predefined number of generations are processed). In the end the best chromosome of the last generation is reported as a final solution.

In the literature, Holland's genetic algorithm is commonly called the Simple /Standard Genetic Algorithm or SGA. An outline of a GA is as follows.

```
STANDARD GENETIC ALGORITHM (SGA):
{
Initialize population;
Evaluate population;
  while termination criterion not reached
  {
  select solutions for next population;
  perform crossover and mutation;
  evaluate population;
  }
}
```

The flow diagram of a genetic algorithm is shown in Figure 4.2.

From the above discussion it is clear that for using GAs to solve a given combinatorial optimization problem one has to come up with:

* A suitable encoding of solutions to the problem as chromosomes, and

* Translate cost function into a fitness measure.

4.4.1 Some Comments

As we can see, the outline of Basic GA is very general. There are many things that can be implemented differently in various problems. First question is how to create

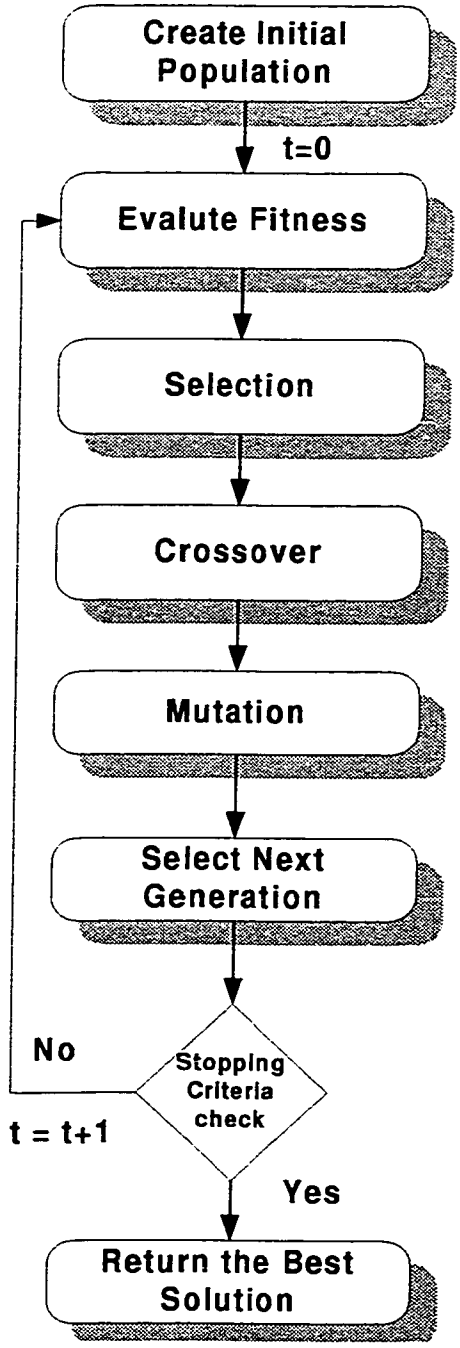

Figure 4.2: Outline of a Genetic Algorithm

chromosomes, what type of encoding choose. With this is connected crossover and mutation, the two basic operators of GA. Encoding, crossover and mutation are introduced in next chapter. Next questions is how to select parents for crossover. This can be done in many ways, but the main idea is to select the better parents (in hope that the better parents will produce better offspring). Also we may think that making new population only by new offspring can cause lost of the best chromosome from the last population. This is true, so so called elitism is often used. This means, that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

4.5 Encoding

Encoding of chromosomes is one of the problems, when you are starting to solve problem with GA. Encoding very depends on the problem. In this chapter will be introduced some encodings, which have been already used with some success.

4.5.1 Binary Encoding

Binary encoding is the most common, mainly because first works about GA used this type of encoding. In binary encoding every chromosome is a string of bits, 0 or 1. For example,

Chromosome A : 1011001011001 Chromosome B : 1111111000001

Binary encoding gives many possible chromosomes even with a small number of alleles. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

4.5.2 Permutation Encoding

Permutation Encoding Permutation encoding can be used in ordering problems, such as travelling salesman problem or task ordering problem. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence. For Example,

Chromosome A : 1 5 3 2 6 4 7 9 8 Chromosome B : 8 5 6 7 2 3 1 4 9

Permutation encoding is only useful for ordering problems. Even for this problems for some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it).

4.5.3 Value Encoding

Direct value encoding can be used in problems, where some complicated value, such as real numbers, are used. Use of binary encoding for this type of problems would be very difficult. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.

Value encoding is very good for some special problems. On the other hand,

for this encoding it is often necessary to develop some new crossover and mutation specific for the problem.

4.6 Genetic Search Operators

4.6.1 Selection

In nature we observe that stronger individuals survive, reproduce and hence transmit their good characteristics to subsequent generations. This natural selection process inspires the design of the selection operator in the GAs. Given a generation of chromosomes, each of them is evaluated by measuring its fitness, which is, in fact, the quality of the solution the chromosome represents. The fitness value is usually normalized to a real number between 0 and 1, and the higher the value the fitter the chromosome. Usually a proportionate selection scheme is used. With this scheme, a chromosome with a fitness value f is allocated $f/favg$ offspring in the subsequent generation, where $f/favg$ is the average fitness value of the population. Thus a chromosome with a larger fitness value is allocated more offsprings while a chromosome with a smaller fitness value, for example, less than the average may be discarded in the next generation.

4.6.2 Crossover

Crossover is a crucial operator of GAs and is applied after selection. While selection is used to improve the overall quality of the population, crossover is used to explore the search space to find better individuals. Pairs of chromosomes are selected randomly from the population for application of the crossover operator. In the simplest approach, a point is chosen randomly as the crossover point. The two chromosomes then exchange the portions beyond the crossover point to generate two new chromosomes.

A simple example of the standard crossover is given in Figure 4.3. The rationale is that after the exchange the newly generated chromosomes may contain the good characteristics from both the parent chromosomes and hence, possesses a higher fitness value. Nevertheless the newly generated chromosomes may be worse their parents. With respect to this, the crossover operator is not always applied to the selected pair of chromosomes. It is applied with certain pre-specified probability called the crossover rate, denoted by μ_c.

4.6.3 Mutation

After crossover, strings are subjected to mutation. Mutation of a bit involves flipping it: changing 0 to 1 or vice versa. Like the crossover, mutation is applied with a certain probability called the mutation rate, which is denoted by μ_m. The bits of

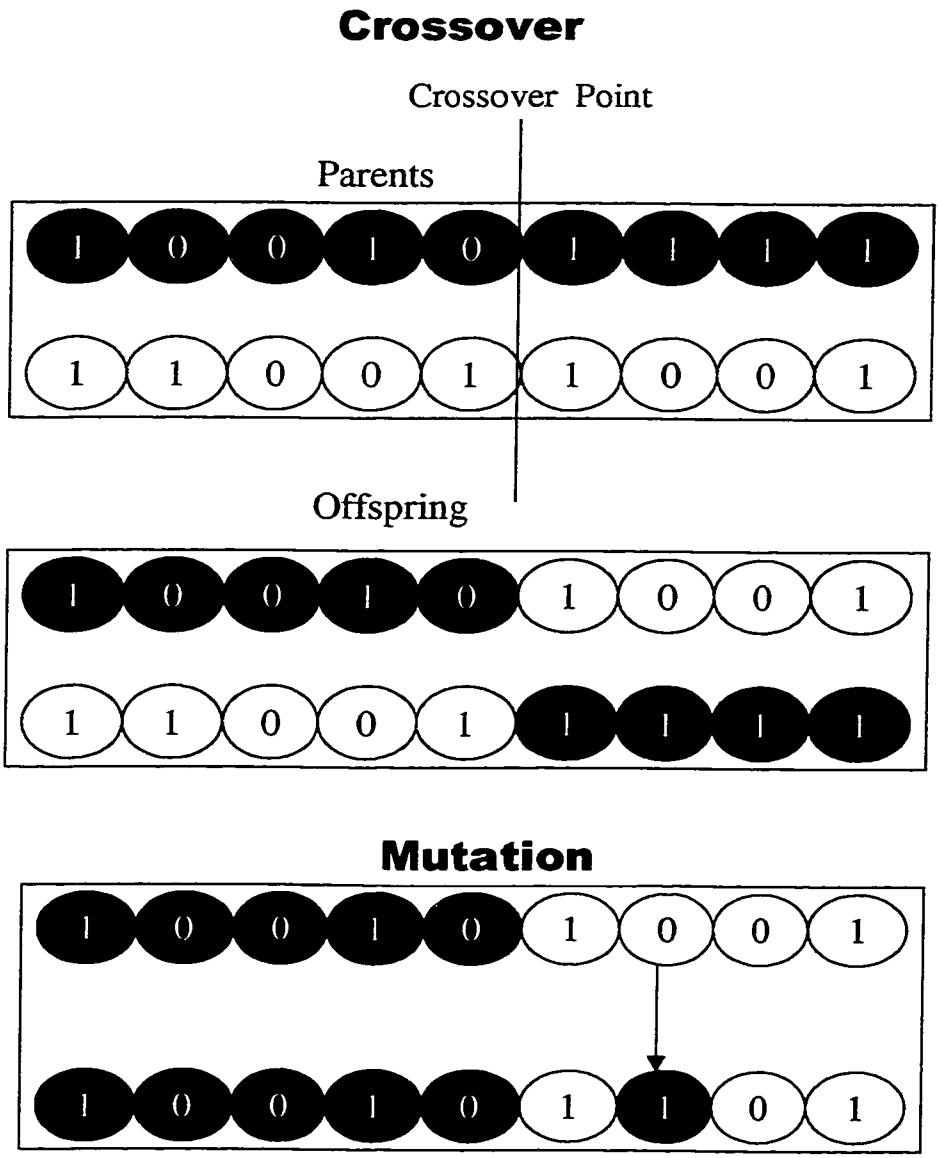

Figure 4.3: A Simple Crossover and Mutation

a string are independently mutated—i.e. the mutation of a bit does not affect the probability of mutation of the other bits. Although mutation is a secondary search operator, it is useful for escaping from the local minima. The SGA treats mutation only as a secondary operator with the role of restoring lost genetic material. For example, suppose all the strings in a population have converged to a 0 at a given position and the optimal solution has a 1 at that position. Then crossover cannot regenerate a 1 at that position, while a mutation could.

Consider a population of chromosomes below:

$$S_1 = [011001]; S_2 = [101100]; S_3 = [110101]; S_4 = [111000].$$

The gene in the fifth position in the entire population is '0'. Therefore, independent of the choice of the cut point and the choice of parents for crossover, this value will never become '1' in any offspring. In other words, if the parent chromosomes do not have a certain characteristic, that characteristic cannot appear in the offsprings. The only way this gene can change its value is by mutation.

4.7 Control Parameters

We can visualize the functioning of GAs as a balanced combination of exploration of new regions in the search space and exploitation of already sampled regions. This

balance, which critically controls the performance of GAs, is determined by the right choice of control parameters. A GA is governed by a number of parameters: Population size N_p, number of generations N_g, crossover rate μ_c and mutation rate μ_m. Finding appropriate values for these parameters requires extensive experimentations. Even with appropriate parameters, optimal solutions cannot be guaranteed due to the probabilistic nature of GAs.

4.8 Recommendations

This section gives some basic recommendations if you have decided to implement your genetic algorithm. These recommendations are very general. Probably we will have to experiment with your own GA for specific problem, because today there is no general theory which would describe parameters of GA for any problem. Recommendations are often results of some empirical studies of GAs, which were often performed only on binary encoding.

4.8.1 Crossover Rate

Crossover rate generally should be high, about 80% -95% (However some results show that for some problems crossover rate about 70% is the best).

4.8.2 Mutation Rate

On the other side, mutation rate should be very low. They are usually kept from 0.001 to 0.05.

4.8.3 Population Size

It may be surprising, that very big population size usually does not improve performance of GA (in meaning of speed of finding solution). Good population size is about 20-30, however sometimes sizes 50-100 are reported as best. Some research also shows, that best population size depends on encoding, on size of encoded string. It means, if you have chromosome with 32 bits, the population should be say 32, but surely two times more than the best population size for chromosome with 16 bits.

4.8.4 Selection

Many methods of selection are given in the literature like, Roulette-wheel Selection, Rank based selection, Tournament Selection etc. Basic roulette wheel selection can be used, but sometimes rank selection can be better.

4.8.5 Encoding

Encoding depends upon the problem and also on the size of instance of the problem. Check chapter about encoding for some suggestions or look to other resources.

4.8.6 Crossover and Mutation Type

The Crossover and mutation type depend on encoding and on the problem. Normally, single, double or cyclic crossovers are used and mutation is implemented in term of flipping of randomly selected genes.

4.9 Applications of GA

Genetic algorithms has been used for difficult problems (such as NP-hard problems), for machine learning and also for evolving simple programs. They have been also used for some art, for evolving pictures and music. Advantage of GAs is in their parallelism. GA is travelling in a search space with more individuals. They are also easy to implement. Once you have some GA, you just have to write new chromosome (just one object) to solve another problem. With the same encoding you just change the fitness function and it is all. On the other hand, choosing encoding and fitness function can be difficult. Disadvantage of GAs is in their computational time. They can be slower than some other methods. But with todays computers it is not so big problem. To get an idea about problems solved by GA, here is a short list of some

applications:

- Data Fitting
- Shape Recognition
- Image Processing
- Nonlinear Dynamical Systems - predicting, data analysis
- Designing Neural networks, both architecture and weights
- Robot Trajectory
- Evolving LISP programs (genetic programming)
- Strategy Planning
- Finding shape of Protein molecules
- TSP and Sequence Scheduling
- Functions for creating images

Chapter 5

Automatic Knot Placement with NURBS using GA

5.1 Introduction

This chapter gives the details of the implemented system. Our method automates the whole process of knot selection and consists of many steps. So it is necessary to understand each step in order to grasp the overall picture of the implemented work.

Our work basically consists of Parametric NURBS fitting for:

- Curves
- Surfaces

For the curves, we have enhanced the work of [34] for the parametric curves and

surfaces. In case of curves, as stated earlier, we have implemented NURBS fitting taking fonts as a case study. Before we go into the details of each, let us first discuss two basic techniques applied for data fitting.

1. Interpolation

2. Approximation

5.1.1 Capture by Interpolation

In this form of capture, parametric curve is constrained to pass through all the given set of data points. Lets for example t is our parameter, ranging from 0 to 1, and for some values of t we have corresponding (x, y) values. These (x, y) values are our data points. But for those values of t we do not have corresponding (x, y) values, we devise an interpolating function to find them. The technique of capturing outline of data parametrically by interpolation is suitable when the data points describing the digitized contour are sufficiently accurate and smooth.

5.1.2 Capture by Approximation

In this form of capture, parametric curve passes close to the given set of data points, but not necessarily through them. By some distance criteria, it is ensured that the overall shape of the font, described by given set of data points, is preserved. The distance (between captured curve and given data point) is commonly measured along

a coordinate or along a normal to the captured curve. The distance value can be used in a variety of ways to achieve best-fitting approximation to given set of data points. It may be employed in a way that requires all the data points to be within a predefined tolerance limit. Alternatively, an acceptable approximation may be where the average of all the distance values is below some predefined tolerance level.

The technique of outline capture of data parametrically by approximation is suitable when the data points describing the digitized contour contain some erroneous values (e.g. noise). The function/curve that approximates the known values, on the average, minimize the total error. This idea is based on the statistical notion that errors in data are randomly distributed.

In our method of data fitting we have chosen capture by approximation technique. There are several reasons for this decision.

The fundamental objective behind our system is to obtain the parametric representation of character/data/surface that matches with its digitized image/original data.

1. When the original image on paper is scanned to get its bitmap representation, there is always a possibility that all the features of original image are not preserved. Because limitation of scanner scanning capability and inter pixel gap problem. To understand the inter pixel gap problem lets consider a situation that a delicate curve passes through gap between pixels. Then its bitmap representation will select a pixel and that selection may not truly reflect the

font designer choice. Therefore to design fonts of smooth outline, capture by approximation is better choice.

2. Extracted contour or boundary has noise (jagged edges). In this situation capture by interpolation will require a large number of data points to be saved and outline will not be smooth. Therefore, capture by approximation not only require very few data points but it gives smooth outline fonts.

3. Another advantage of capture by approximation technique is that we approximates the original contour up to certain tolerance level and if this tolerance level is zero then it is interpolation. So we have the maximum flexibility in our method.

5.2 Data Fitting using NURBS

Let us assume that F is the given data to be fitted with NURBS. The given data may or may not contain any noise. For generalization let us assume that ϵ is the measurement error. So, we can write.

$$F(t) = f(t) + \epsilon(t) \tag{5.1}$$

where t represents the parameter.In the above equation, $f(t)$ is the underlying func-

tion which is to be approximated using NURBS and $\epsilon(t)$ represents the measurement error at the particular value of t at that data point.

Let $\xi_i(i = 1-m, 2-m, ..., n+m)$ be knots for data fitting, where n is the number of knots $\xi_i(i = 1, 2, ..., n)$ located in the interval (0,1) of the parameter t and m is the order (degree+1) of NURBS. In our case we have used *chord-length parameterization* for the parameter t (see §5.2.1). At the end of the interval [0,1], we set,

$$\left\{ \begin{array}{l} a = \xi_{1-m} = ...\xi_0, \\ b = \xi_{n+1} = ... = \xi_{n+m}. \end{array} \right\} \tag{5.2}$$

Now, the model function for $f(t)$ is given by

$$S(x) = \sum_{i=1}^{n+m} c_i N_{m,i}(t) \tag{5.3}$$

where c_i are the coefficients of the NURBS.

5.2.1 Chord-length Parameterization

We used *chord-length parameterization* to estimate the parametric value t associated with each point p_i. There are other possible ways to estimate the parametric value t as well. For example *unit parameterization* could be used. But we found *chord-length parameterization* more suitable and accurate. After having the t value associated with each point, we fit parametric cubic Bezier curve to set of data points of each segment.

$$t_i = \begin{cases} 0 & \text{if } i = 1; \\ \frac{|p_1p_2|+|p_2p_3|+,\ldots,+|p_{i-1}p_i|}{|p_1p_2|+|p_2p_3|+,\ldots,+|p_{n-1}p_n|} & \text{if } 2 \leq i \leq n-1; \\ 1 & \text{if } i = n \end{cases}$$

5.2.2 Calculation of Control Points

The approach used in our algorithm for both curves and surfaces is based on non-uniform knot-vector. In the genetic formulation of the problem (See §5.3), these knot vectors actually represent the chromosomes. The *knots* are the data points $[V_1, V_2, ...V_n]$, which are to be mapped to the control points $P = [P_0, P_1, ..., P_n, P_{n+1}]$. Note that n data points are mapped to $n + 2$ control points. The mapping can be

formulated as the solution to a system of linear equation

$$MP = V \tag{5.4}$$

The above equation relates the data points V to the control vertices P through a matrix M. The matrix M is easy to invert and is given by

$$\mathbf{M} = 1/6 \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \end{bmatrix} \tag{5.5}$$

Eq. 5.2.2 can now be written as

$$\mathbf{M} = 1/6 \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ \vdots \\ P_{n-1} \\ P_n \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ V_1 \\ V_2 \\ \vdots \\ V_{n-1} \\ V_n \\ 0 \end{bmatrix} \tag{5.6}$$

We can see that in Eq. 5.6 the vector V is padded with zeros at the beginning and at the end.

5.3 Genetic Formulation of the Problem

5.3.1 Curves

In this section the genetic formulation of the problem has been presented. The genetic control parameters have also been discussed[8]. The scheme used to convert the original continuous problem into a discrete optimization problem is same as [34]. Each data point corresponds to a single gene in the bit string of a chromosome. In

this formulation if a gene is equal to 1, we put a knot at the corresponding data point and if the gene is equal to 0 we do not (see Figure 5.1).

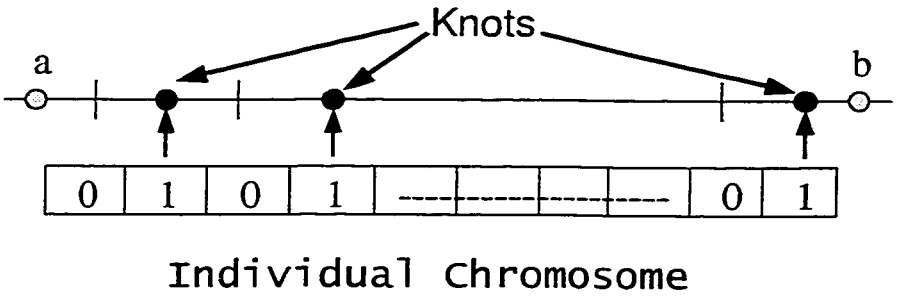

Figure 5.1: Genetic Formulation

If the given points lie in the interval [a,b], then the appropriate number of knots are determined in the interval (a,b) called the *interior knots n.*

These knots determine the control points(see §5.2.2) needed to approximate the underlying curve outline. Since we are applying NURBS, therefore, the weights associated with these control points are also to be optimized, which makes the problem consisting of two search spaces at the same time. One search space is related to the optimization of the *interior knots*, while the other one to the positive weights associated with the calculated control points. The weights consist of genes consisting of numbers, which can be positive real or integers.e.g.

10 45 67 98 34 99

represents a valid gene in our application.

This dual space search in makes our algorithm a *"Nested Genetic Algorithm"*.

5.3.2 Surfaces

For surfaces a chromosomes are nothing but a mesh, in which 1's and 0's are randomly distributed. As in the case for curves, 1's show the position of knots on the surface (see Figure 5.2).

In case of surfaces,these mesh structure chromosomes are produced two times. Once for calculating NURBS along parameter u and the other along parameter$t$ and NURBS are calculated separately along both the parameters (see §5.15).

As far as the construction of chromosomes for weights is concerned, they are also meshes, not consisting of 0's and 1's but numbers. e.g.

23	45	67	89	90
34	56	78	23	12
87	65	32	54	21
22	63	92	81	49

is a valid chromosome.

1	1	1	0	1	1	0	0
1	1	1	1	0	1	0	1
0	1	1	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0
0	1	0	1	0	1	0	0
0	1	0	0	1	0	1	1

Figure 5.2: A Chromosome Representing a Surface

5.4 Initialization of Population

The initial population, consists of K individuals of genelength L.The genes are randomly set to 0 and 1. However,in case of curve fitting on a font outline, the significant points are determined before the creation of initial population and the genes corresponding to those points are intentionally set to 1 in the initial population and in the population of the subsequent generations.The idea behind this scheme is not to lose those points as they are important in determining the outlines of the shapes.

5.5 Selection

The method used for selection used in our algorithm is the Roulette-wheel selection. This selection method has been used both for curves and surfaces.

5.6 Crossover

The crossover, used in algorithm is the simple double point crossover both for the bit string chromosomes and the chromosomes representing the weights of the control points. The working of a double point crossover is same as a simple single point crossover4.3, but in this crossover two cut points are chosen randomly instead. The middle parts are exchanged between the two chromosomes selected for crossover. The probability of crossover C is taken to be 0.7 for both sort of chromosomes and

for curves and surfaces.

5.7 Mutation

The mutation used for the population consisting of bit strings is illustrated in Figure 4.3.For the weights population, it is evident that we can not use the same scheme, therefore, we adopted the following strategy:

1. For each gene in the chromosome repeat
2. Generate a random number uniform in the interval [0,1]. If the value is less than or equal to the probability of mutation, then go to step 3. otherwise, go to step 5.
3. Generate a random integer number uniform in the interval zero and the gene length L. Go to the gene having the index same as the generated number.
4. Generate another random integer number and replace the previously selected number with it.
5. Move to the next gene.

The probability of mutation M is taken to be 0.001 for bit string chromosomes and 0.01 for number chromosomes both for curves and surfaces.

5.8 The Fitness Measure

Akaike's Information Criterion(AIC) [1] is used as a fitness measure. By using AIC we can choose the best model among the candidate models automatically. AIC is given by

$$AIC = Nlog_e Q_1 + 2(2n + m). \tag{5.7}$$

where N is the number of data,Q_1 is given by equation 4, n is the number of interior knots,m is the order of the spline to be fitted on the given data. The subscript of AIC means the dimension of the data. It should be noted that the smaller value of Eqn 5.7 gives better fitness.

$$Q_1 = \sum_{j=1}^{N} w_j\{\{Sx_j(t) - x_j(t)\}^2 + \{Sy_j(t) - y_j(t)\}^2\}, \tag{5.8}$$

where $S(t)$ is the spline approximated over the data F and w_j is the weight of data, taken to be 1 for all data points in our case. The subscript of Q means the dimension of the data.

5.9 The Knot Ratio

This is the ratio of 0's and 1's in a bit string chromosome. So it has only been used for the bit string chromosomes. It is denoted by R. In our algorithm $0 < R < 0.5$ has been used.

5.10 Decimation

In case of fonts, we have suggested a parameter which we have named as *"decimation"*. This parameter enables the data to be selected intervally, leaving some points, without loosing the contour of the font. This has been used in order to decrease the genelength of the chromosomes.

5.11 Population for Weights

It is an important parameter added in our GA, which actually corresponds to the weights corresponding to the control points of the individual under consideration.

Weights play an important role in the modification of NURBS shape. There are several ways to alter the shape of NURBS [21]:

- Reposition control points
- Change the weights

- Modify the knot vector
- Move datapoints

The population of weights provides the second alternative among the choices mentioned above. The search space for the weights is explored by the conventional genetic operators like crossover and mutation (see §5.7).

In our algorithm a 20-30 weights'population size was used.

5.12 Generations for Weights

The generations of weights define how much we concentrate on the optimization of an individual under consideration. The initially generated population is crossed, mutated and new weights' population is formed for exploring more in the search space of the weights in order to struggle for the achievement of the desired output (see Figures 5.13 and 5.32)

5.13 Saving the Best Results

This is a very important step in the execution of a GA that we must store the best result obtained so far in order to keep it in the newly formed population. It is hoped that this best chromosome obtained so far probably will produce better solutions after being crossed with the fitter individuals after selection.

5.14 The Curve Fitting to Fonts' Outline

Figure 5.3 shows the basic building blocks of our implemented System. The expanded version of the system is shown in Figure 5.13.

Before peeping into the details of the fitting of NURBS to font outline, let us have a flavor of the steps shown in Fig.5.3.

5.14.1 Getting Digitized Image

Digitized image of a character can be obtained directly from some electronic device or by scanning an image. We used both methods. The quality of digitized scanned image depends on various factors such as image on paper, scanner and attributes set during scanning. The quality of digitized image obtained directly from electronic device depends on the resolution of device, source of image, type of image etc. Some of the digitized images are shown in Figures 5.4, 5.7 and 5.10.

5.14.2 Extraction of Contour

Contour of digitized image is extracted by using some boundary detection algorithm. There are numerous algorithms for detecting boundary. We used the algorithm proposed by [24]. The input to this algorithm is a bitmap file. The algorithm returns number of pieces and for each piece number of boundary points and values of these boundary points $p_i = (x_i, y_i), i = 1, \ldots, N$. Since we are dealing with closed

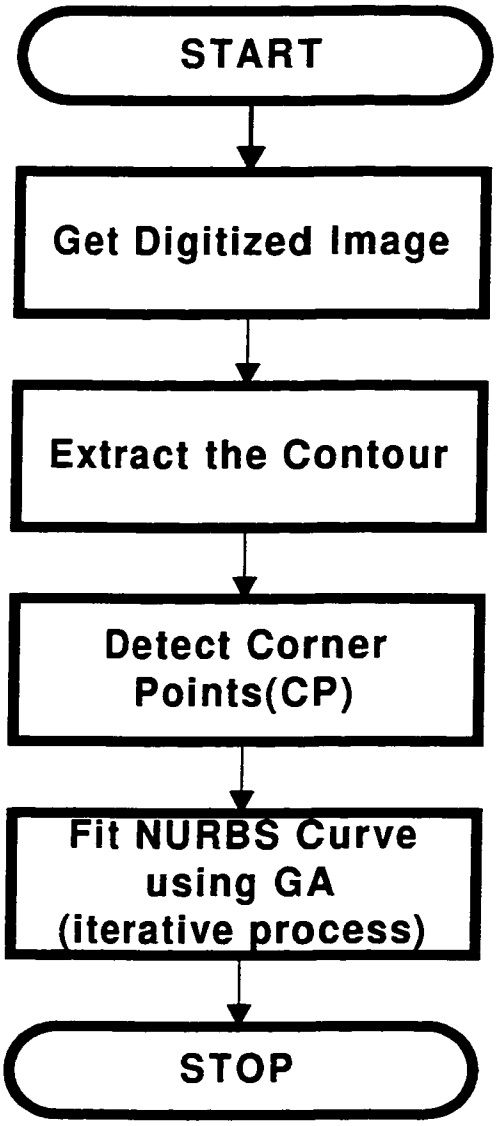

Figure 5.3: Building Blocks of Data Fitting System

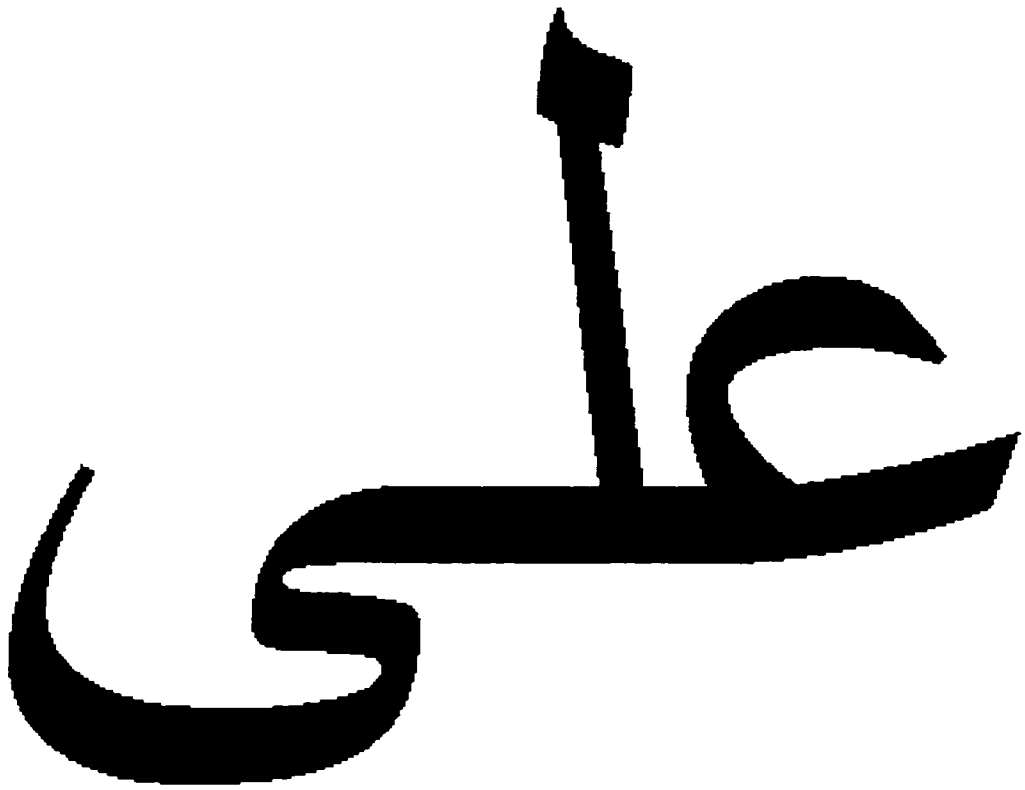

Figure 5.4: Bitmap Image of the Word 'Ali'

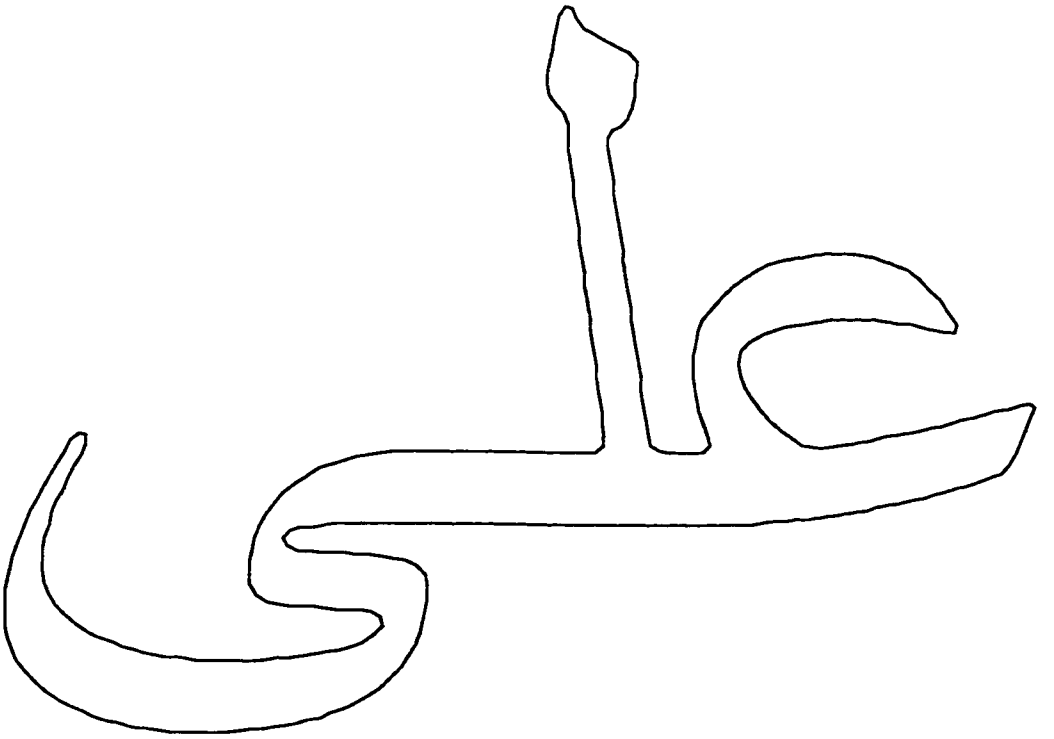

Figure 5.5: Outline Obtained after Boundary Detection

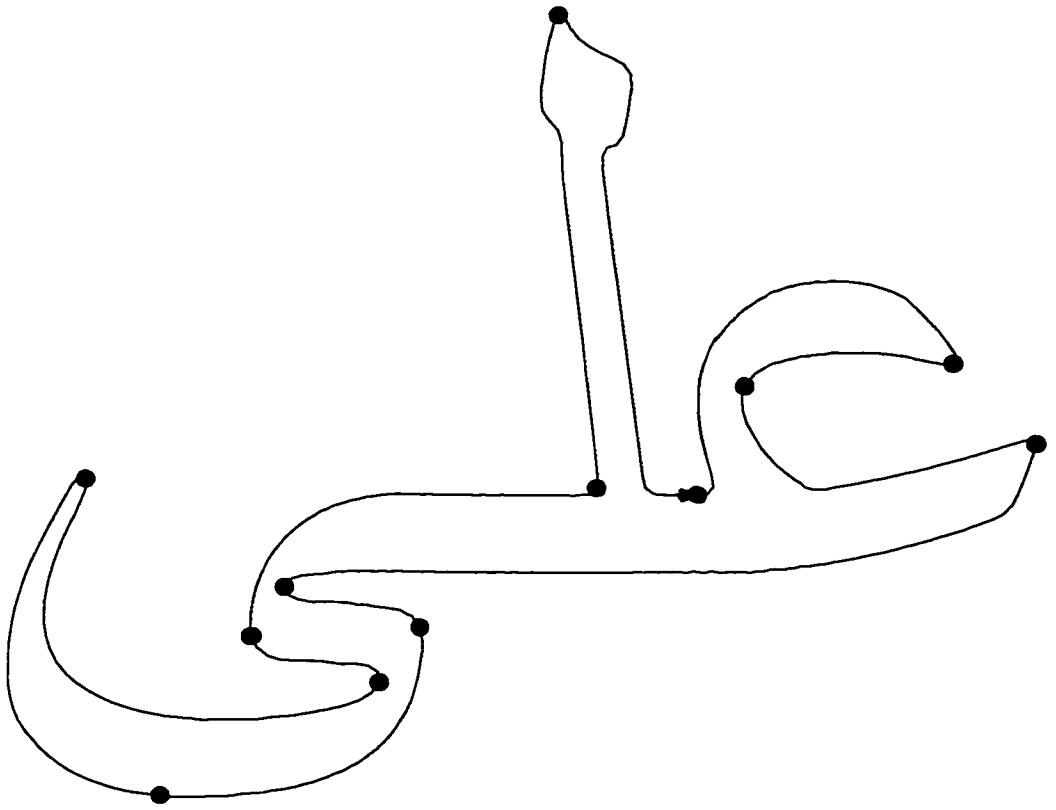

Figure 5.6: Significant Points Obtained after Corner Detection

Figure 5.7: Bitmap Image of Pound Symbol

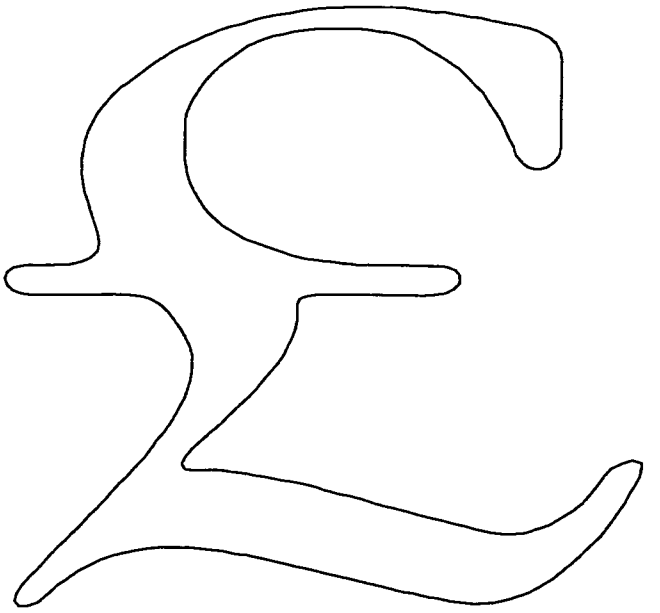

Figure 5.8: Outline Obtained after Boundary Detection

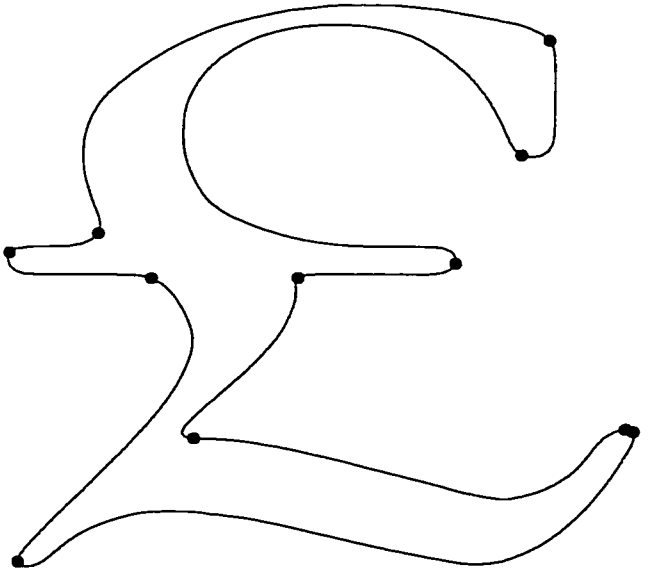

Figure 5.9: Significant Points Obtained after Corner Detection

boundary curves therefore first and last boundary points of each piece have the same value. Figures 5.5, 5.8 and 5.11 show detected boundary of the images of Figures 5.4, 5.7 and 5.10 respectively. Table (5.1) summarizes the results.

The following table gives the number boundary points detected by the boundary detection algorithm for the word 'Ali', the symbol 'Pound' and the letter 'Aich'.

Table 5.1: Contour Data

Figure #	# of Boundary Points	Decimation Applied	Points after Decimation
5.5	1640	4	410
5.8	689	2	344
5.11	320	2	160

Figure 5.10: Bitmap Image of the Letter Aich

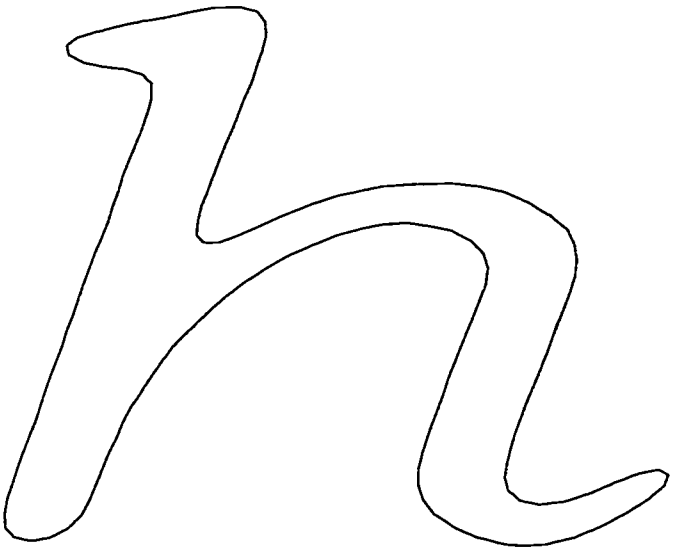

Figure 5.11: Outline Obtained after Boundary Detection

5.14.3 Detecting Significant Points

Significant points are detected from the contour points, (see §5.14.2) using curvature analysis technique. We used the algorithm implemented by [12] for the detection of these points. Significant points have also been stated as corner points in this thesis document.

5.14.4 Conservation of Significant Points

The significant or corner points are conserved in the initial population and the subsequent ones in order to prevent our genetic algorithm form losing these points

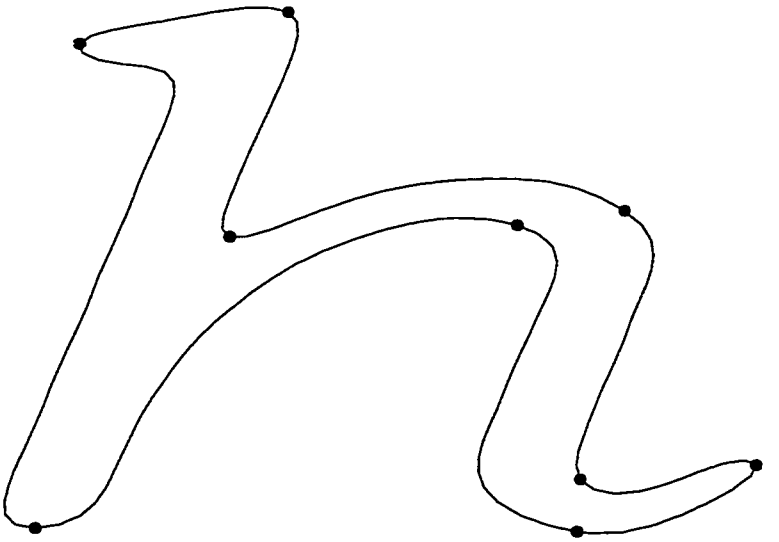

Figure 5.12: Significant Points Obtained after Corner Detection

during crossover and mutation operations. These points are important in capturing the outline of a font and their conservation is achieved by applying an OR operation of the whole population with the chromosome having ones only at the corner point locations only. The operation is performed just after a new population is formed after crossover and mutation operations (see Figure 5.13).

5.14.5 Results

This section presents the results obtained by our data fitting system for fonts. The input parameters have also been shown for each case.

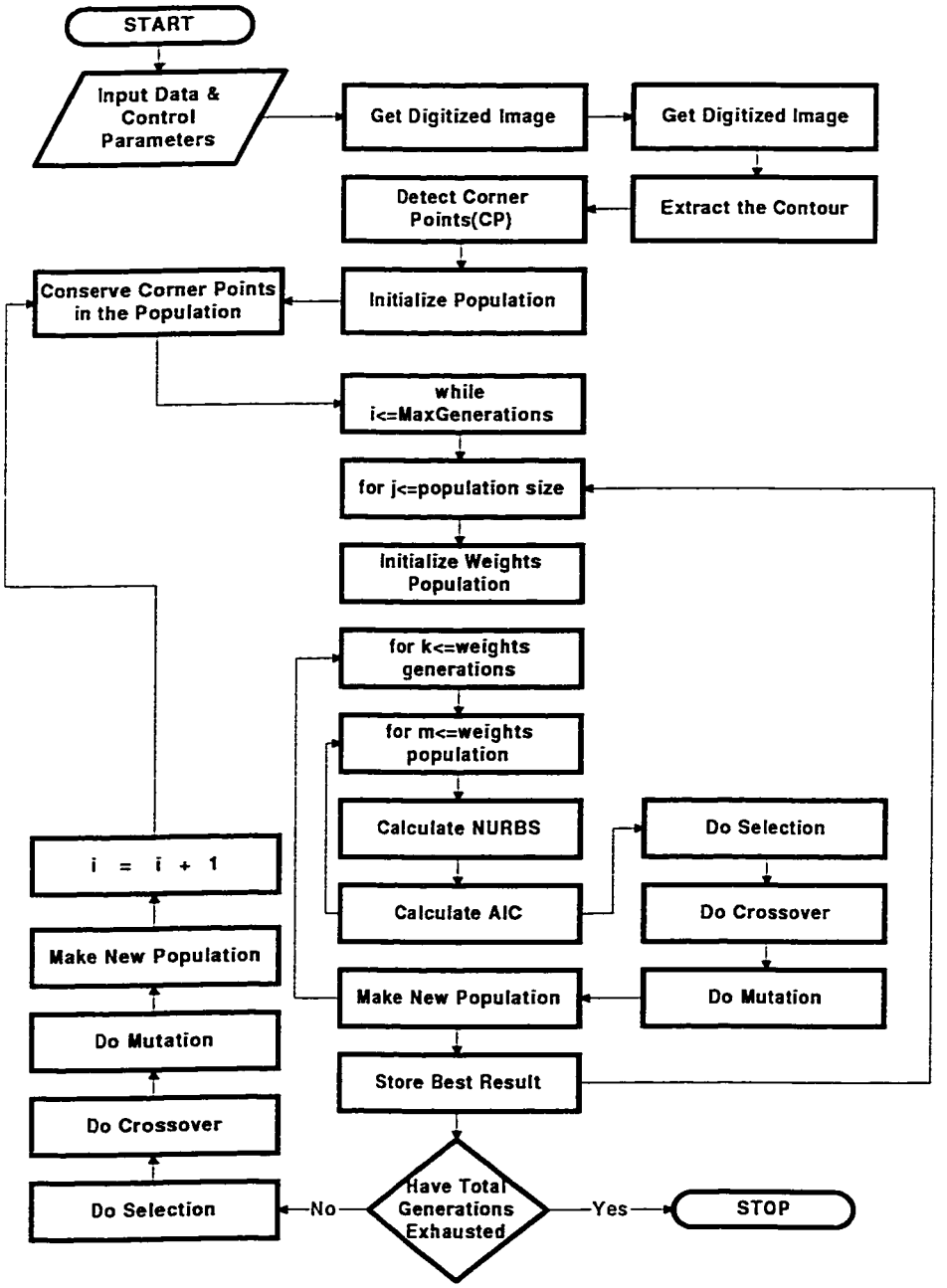

Figure 5.13: Flow Diagram of Data Fitting System for Fonts

Table 5.2: Input Parameters for the Word 'Ali'

Decimation	4
No. of Generations	120
Knot Ratio	0.3
Population size	30
Generations for weights	25
Population size for weights	20
Crossover rate	0.7
Mutation rate	0.001
Selection	Roulette-wheel

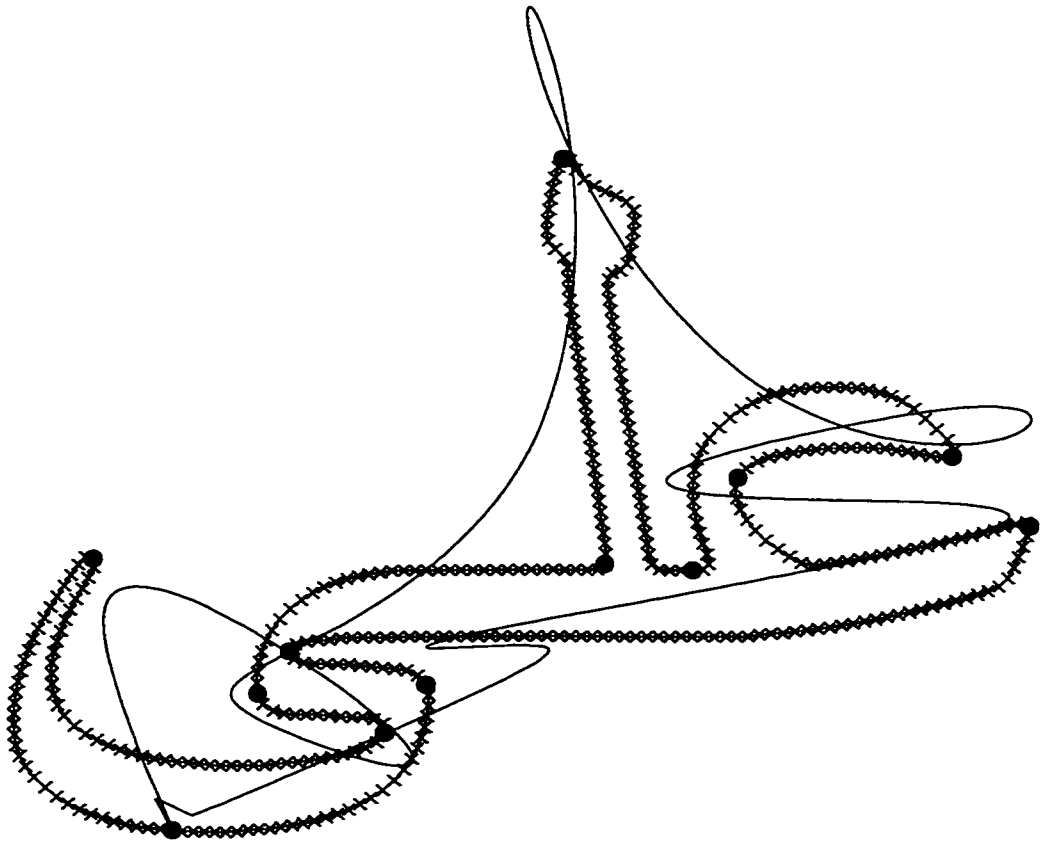

Figure 5.14: Spline Approximated to the Significant Points before Running GA

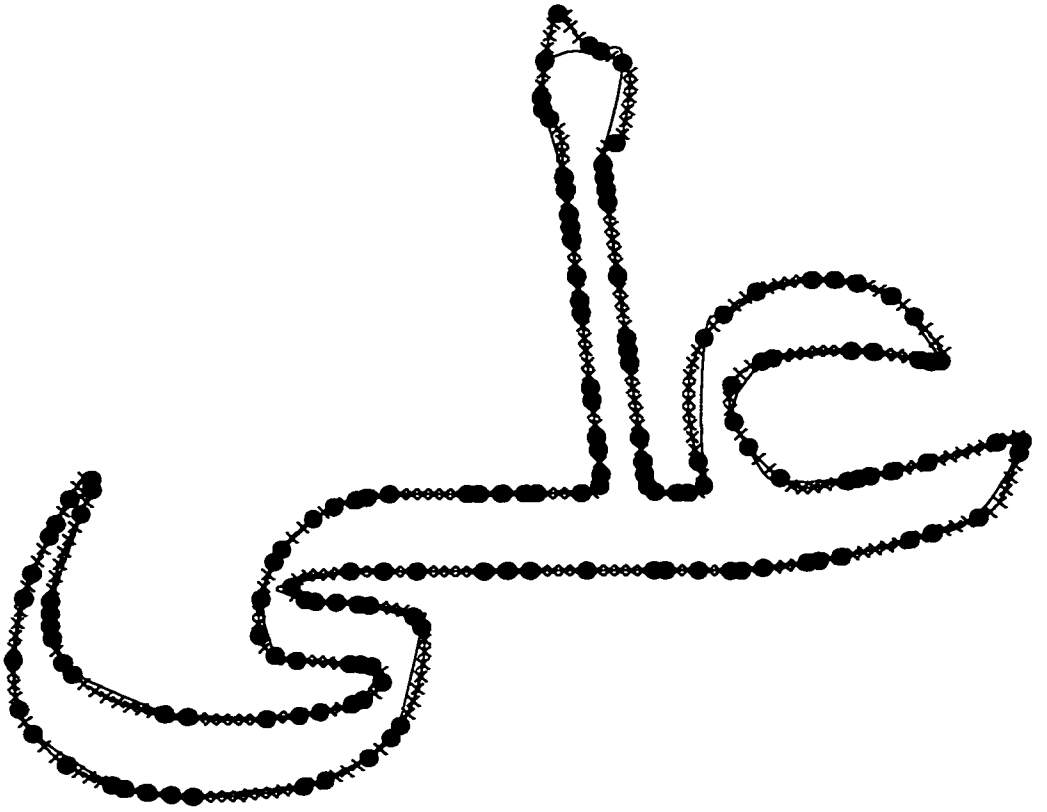

Figure 5.15: Spline Approximation at 25th Generation

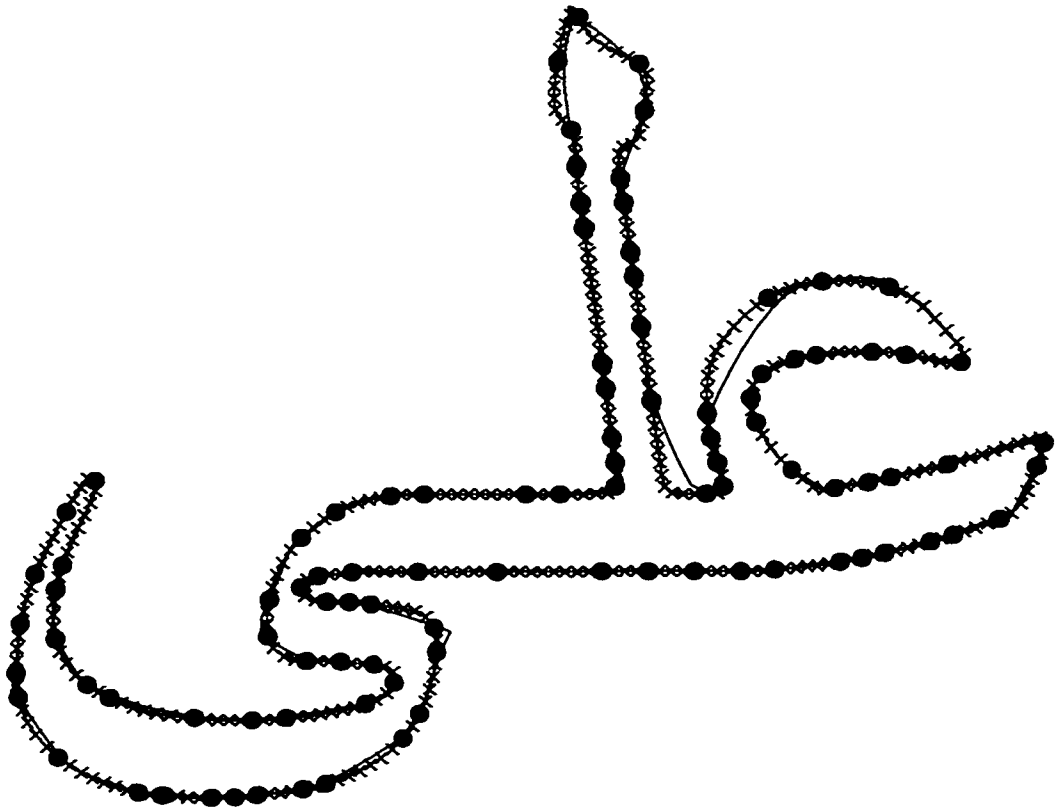

Figure 5.16: Spline Approximation at 50th Generation

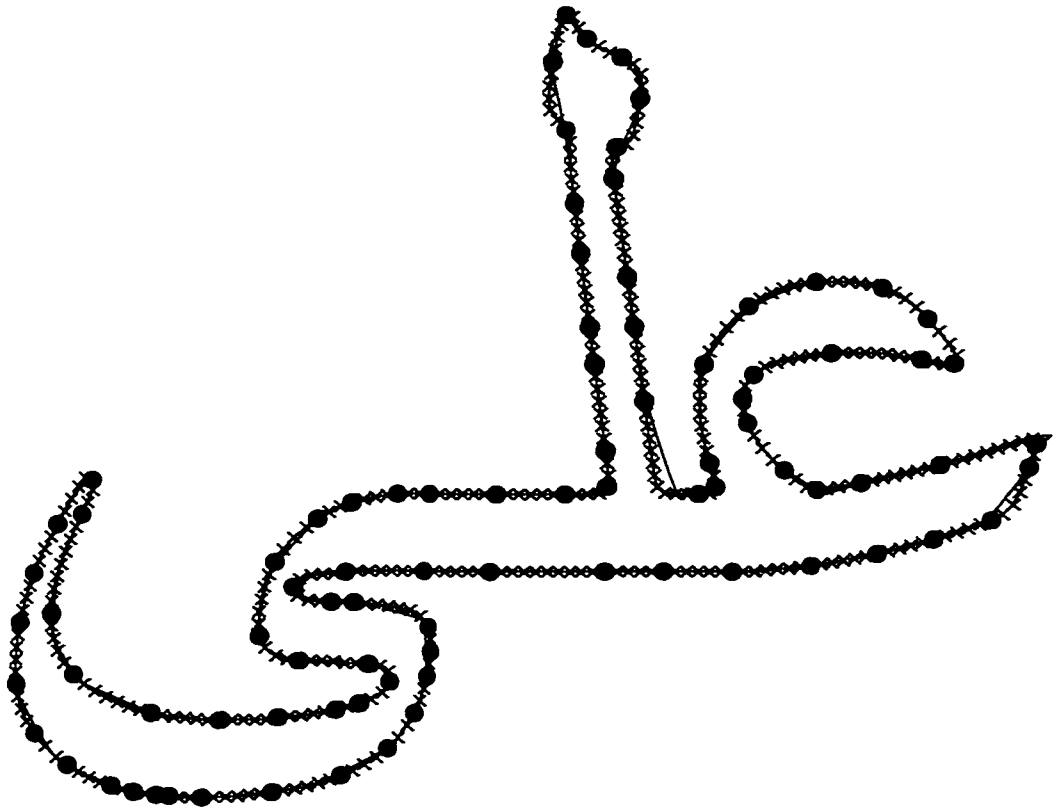

Figure 5.17: Spline Approximation at 75th Generation

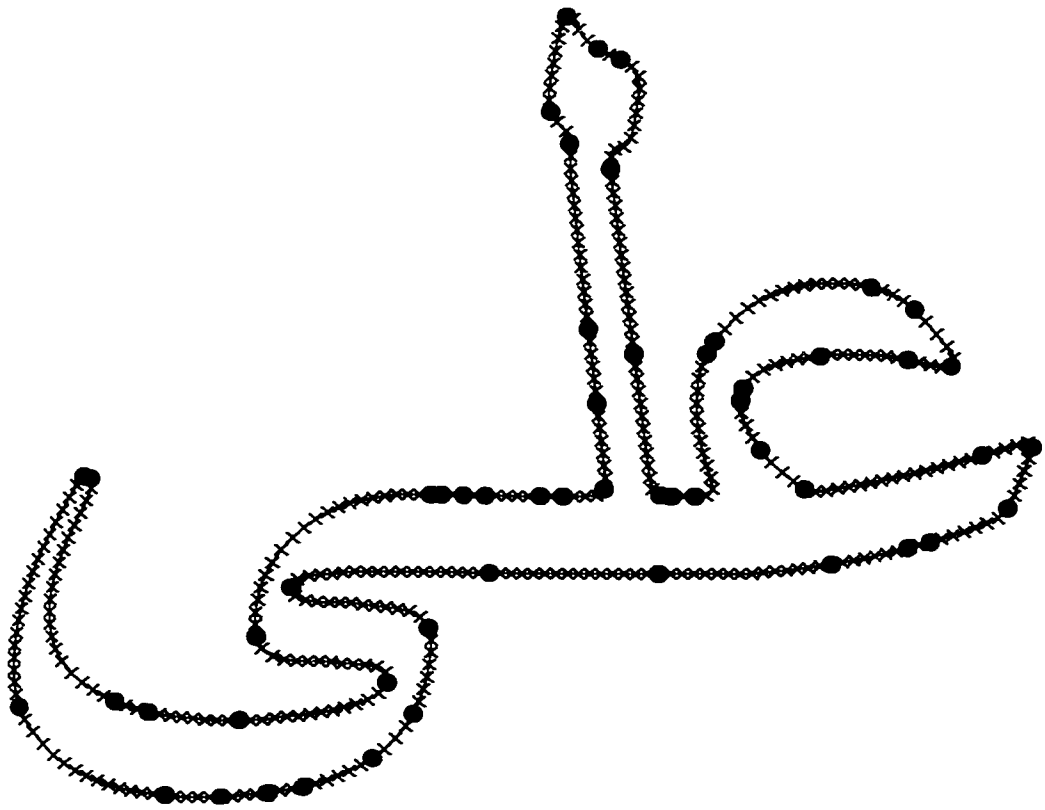

Figure 5.18: Spline Approximation after Convergence

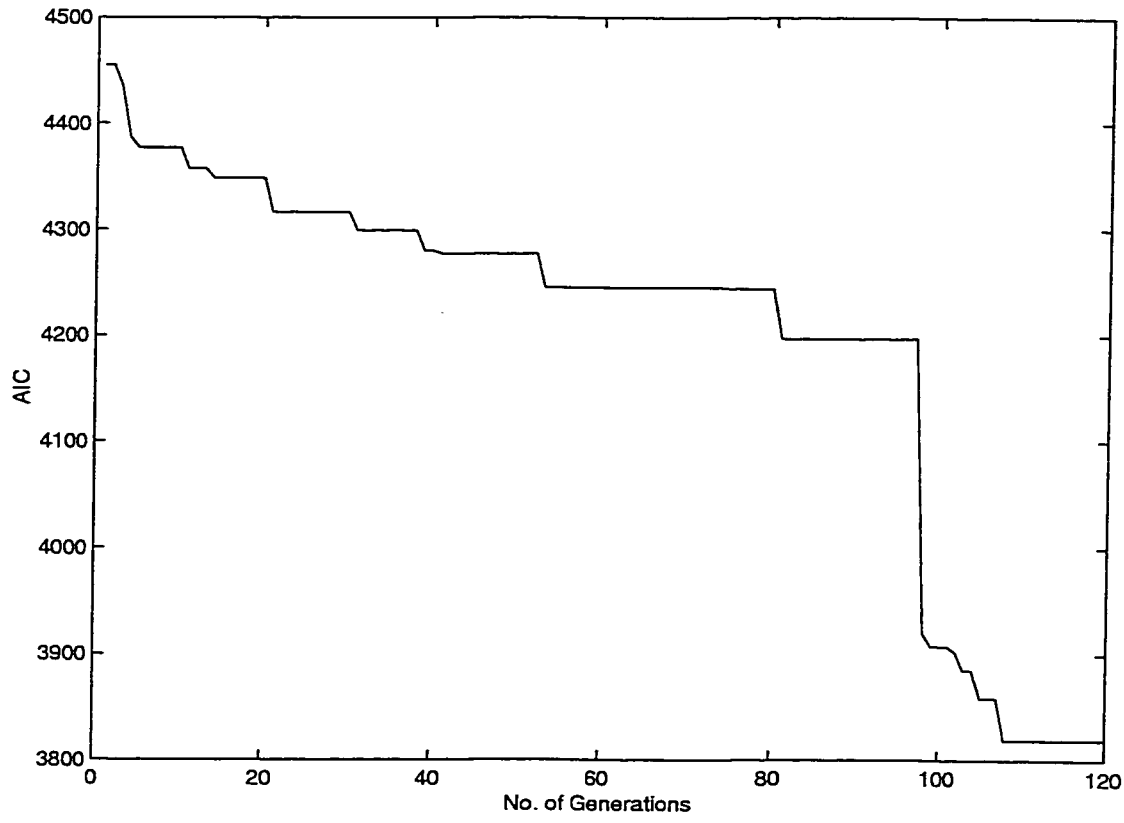

Figure 5.19: AIC Plotted against the Number of Generations

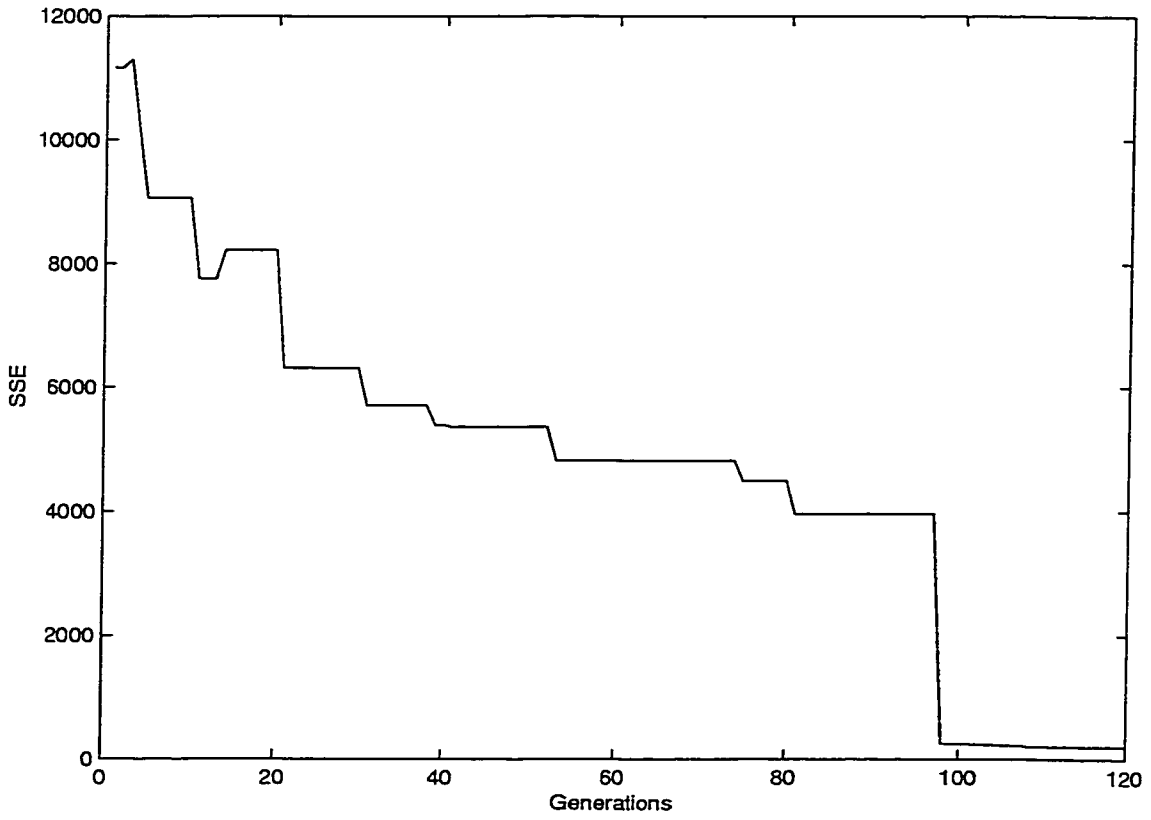

Figure 5.20: Sum Square Error Plotted against the Number of Generations

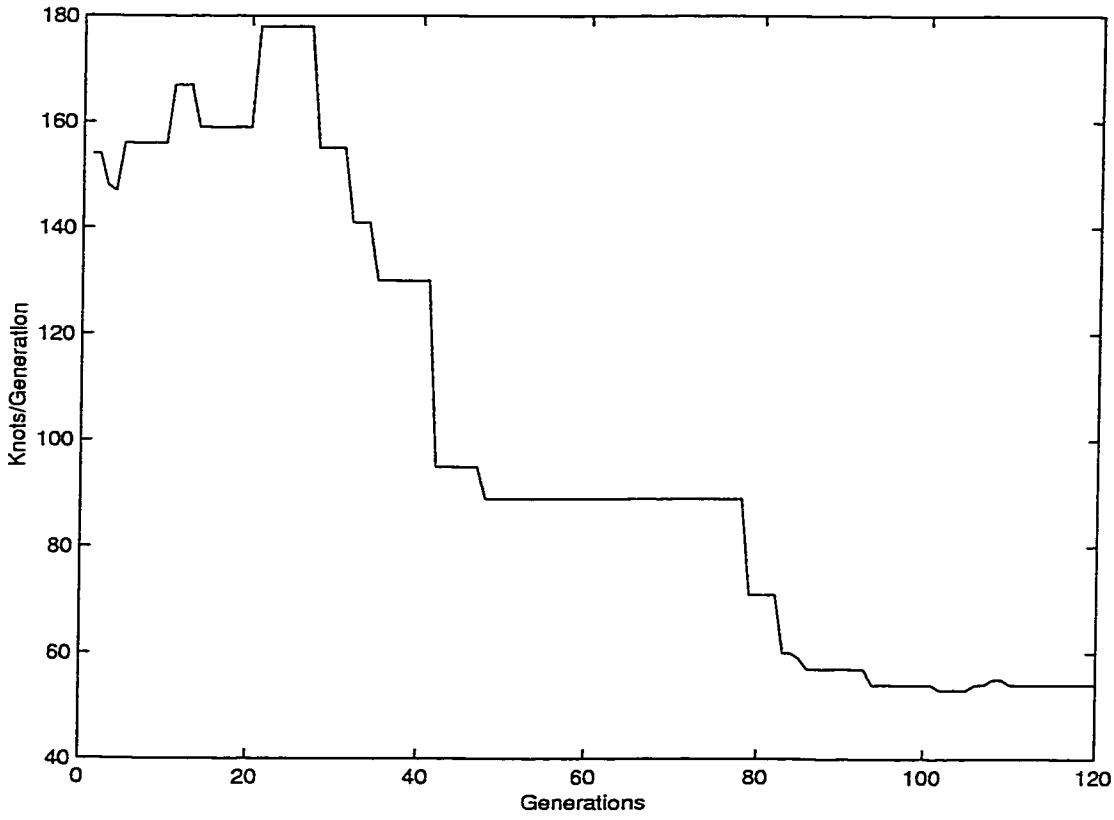

Figure 5.21: Knots Plotted against the Number of Generations

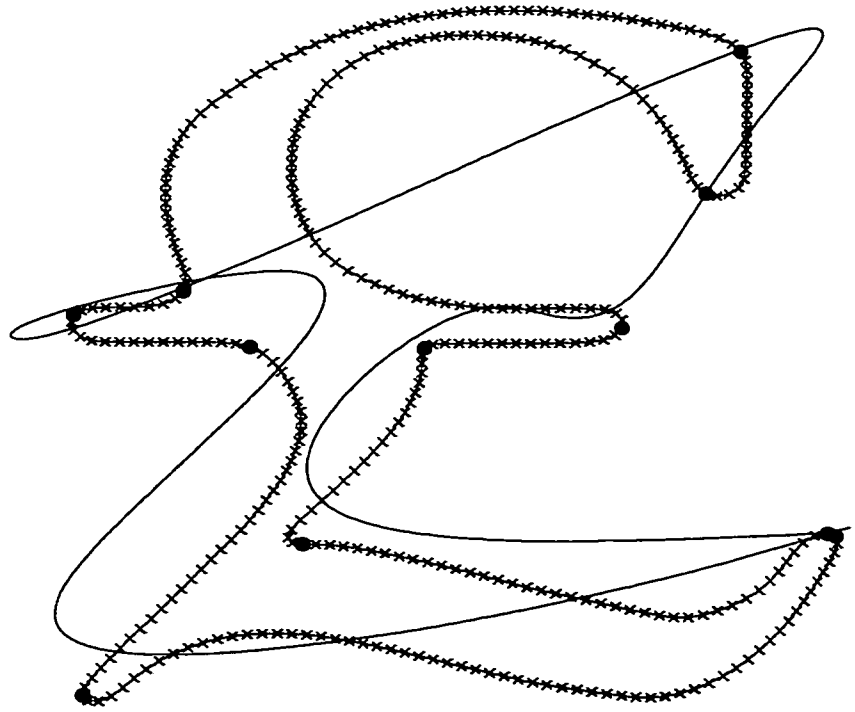

Figure 5.22: Spline Approximated to the Significant Points before Running GA

Table 5.3: Input Parameters for the Symbol 'Pound'

Decimation	2
No. of Generations	130
Knot Ratio	0.3
Population size	30
Generations for weights	25
Population size for weights	20
Crossover rate	0.7
Mutation rate	0.001
Selection	Roulette-wheel

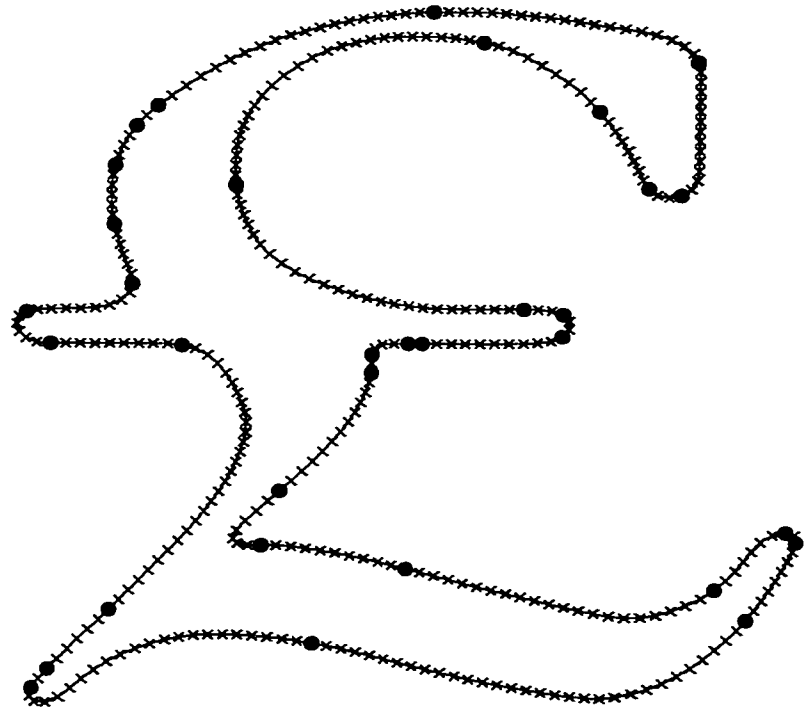

Figure 5.23: Spline Approximation after Convergence

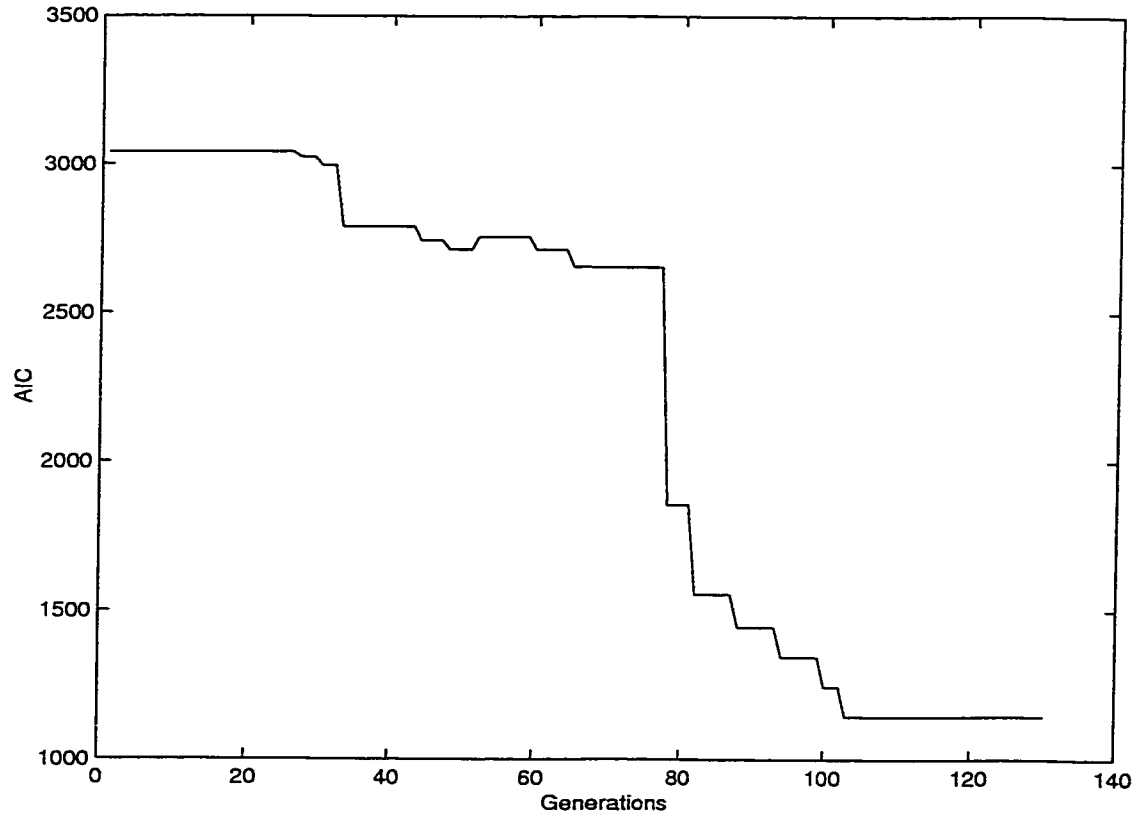

Figure 5.24: AIC Plotted against the Number of Generations

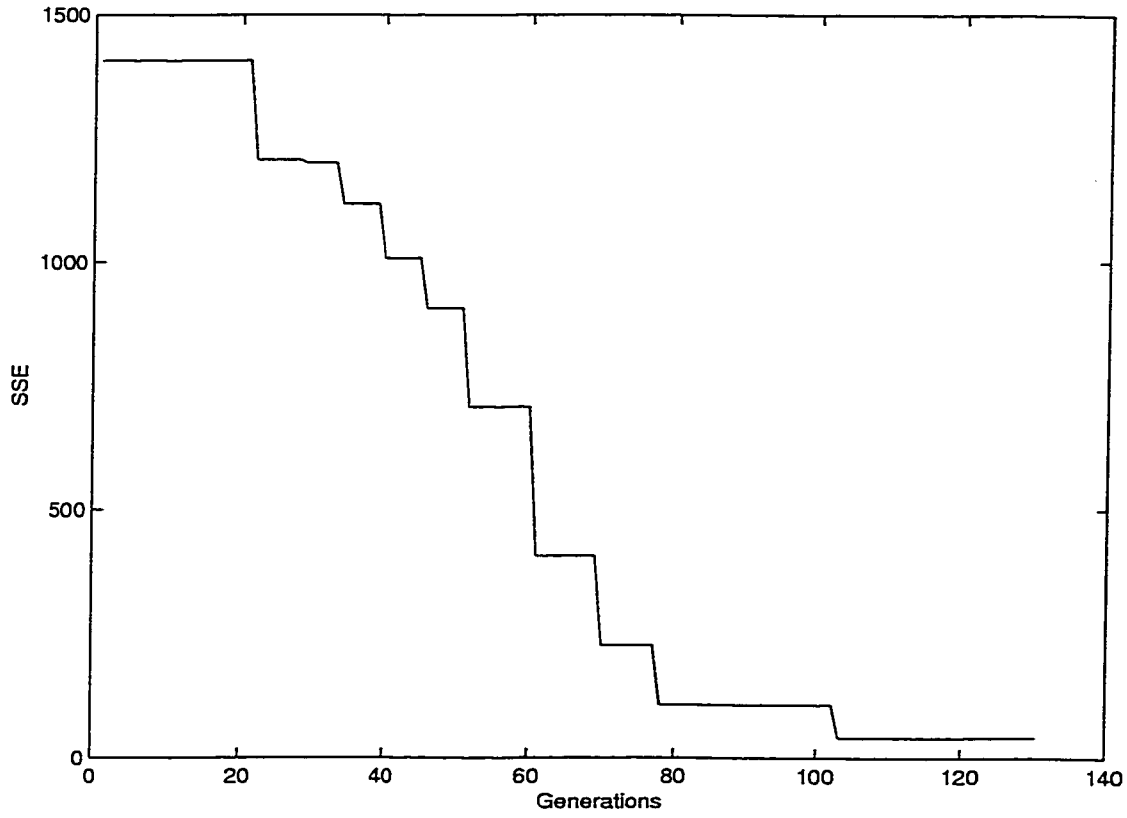

Figure 5.25: Sum Square Error Plotted against the Number of Generations

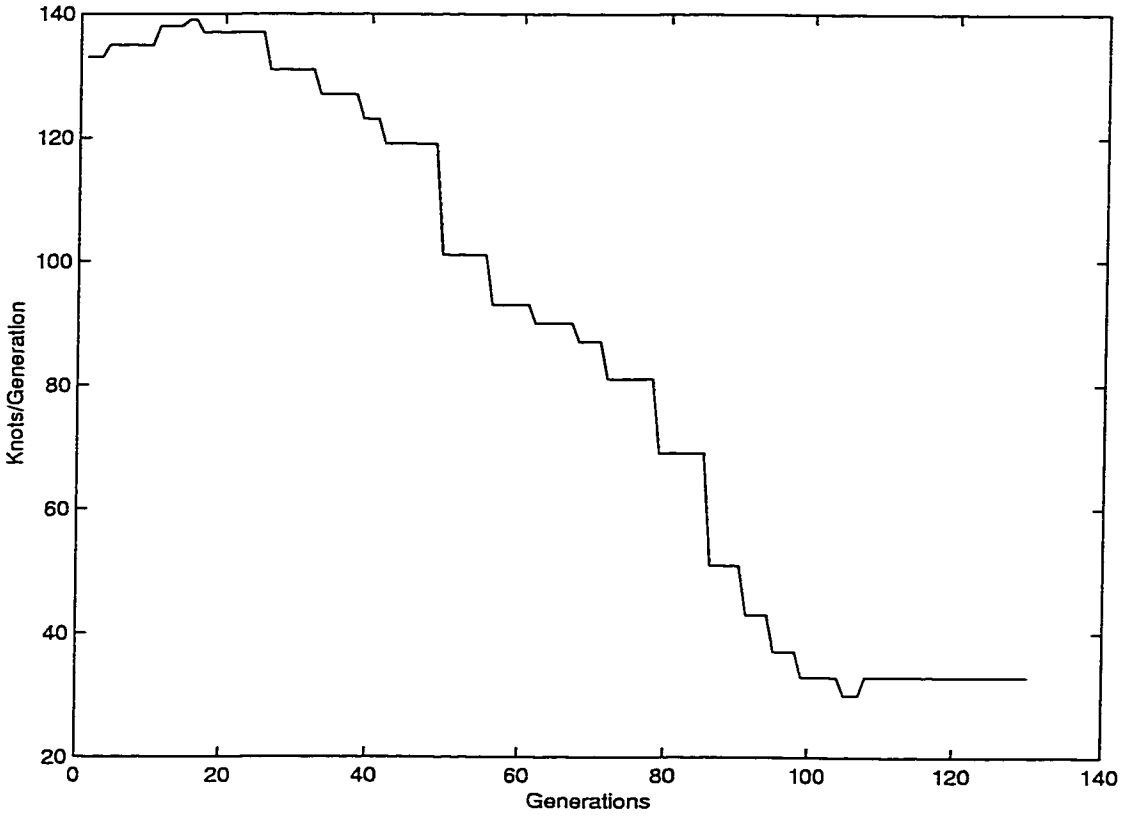

Figure 5.26: Knots Plotted against the Number of Generations

Table 5.4: Input Parameters for the Letter 'Aich'

Decimation	2
No. of Generations	120
Knot Ratio	0.3
Population size	30
Generations for weights	25
Population size for weights	20
Crossover rate	0.7
Mutation rate	0.001
Selection	Roulette-wheel

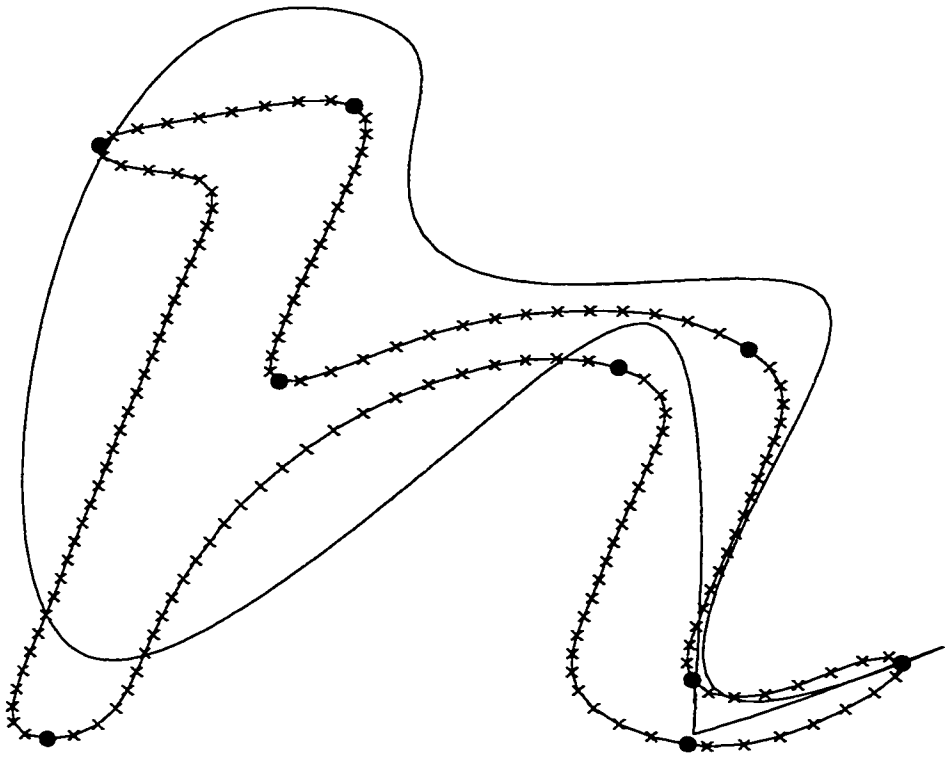

Figure 5.27: Spline Approximated to the Significant Points before Running GA

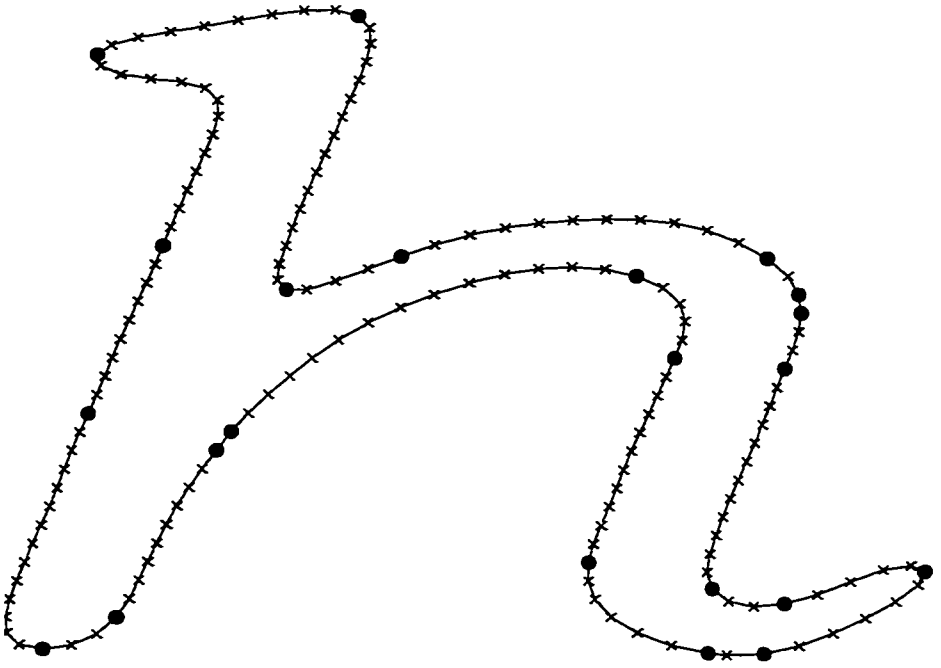

Figure 5.28: Spline Approximation after Convergence

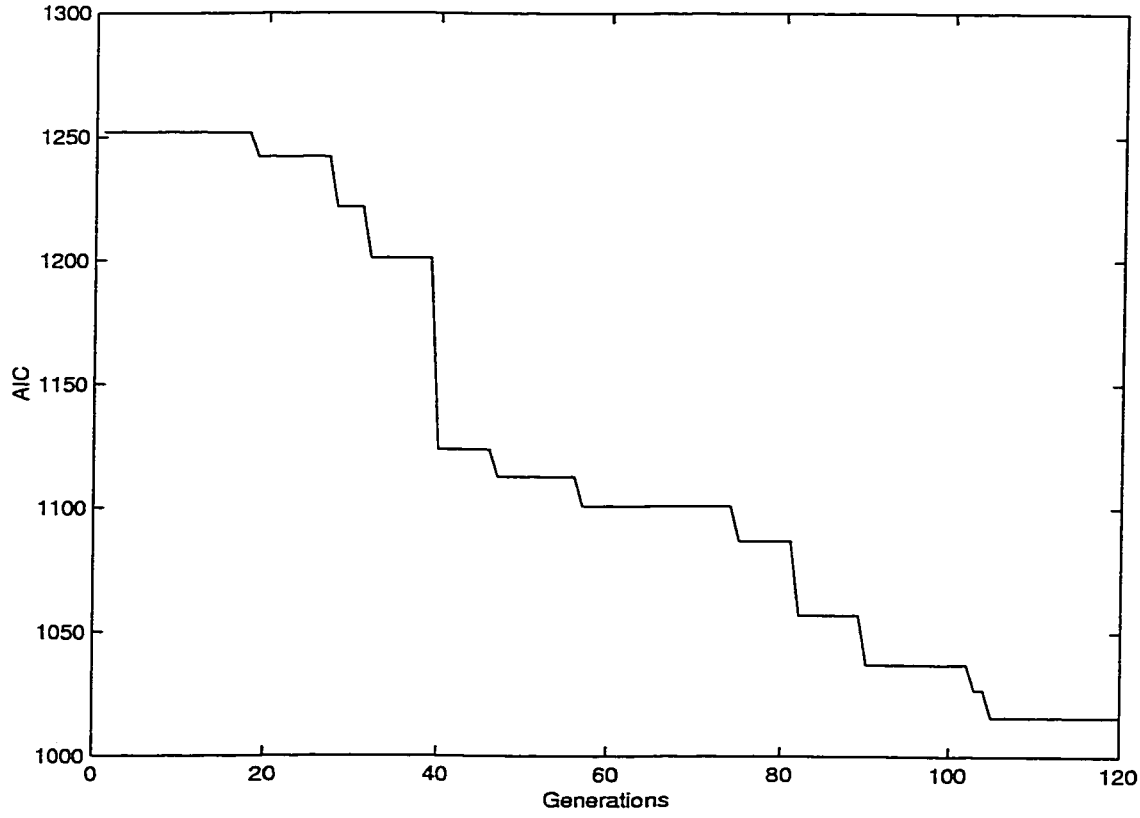

Figure 5.29: AIC Plotted against the Number of Generations

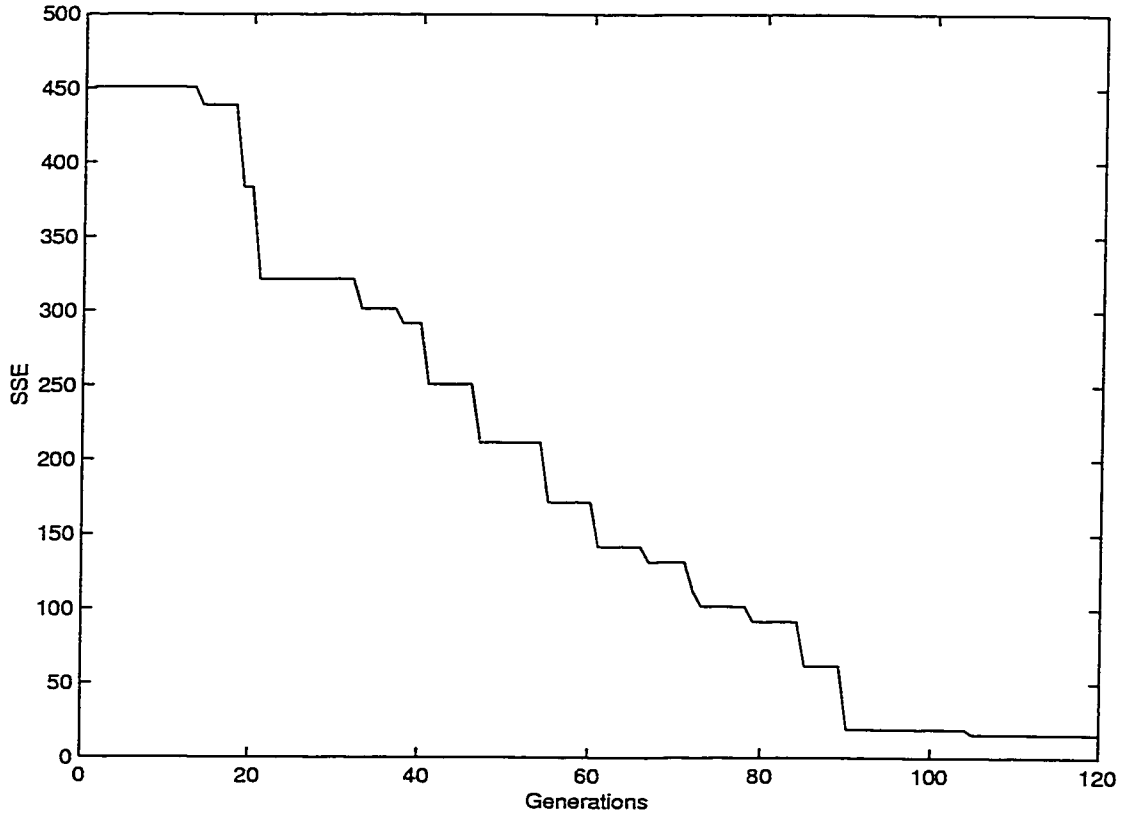

Figure 5.30: Sum Square Error Plotted against the Number of Generations

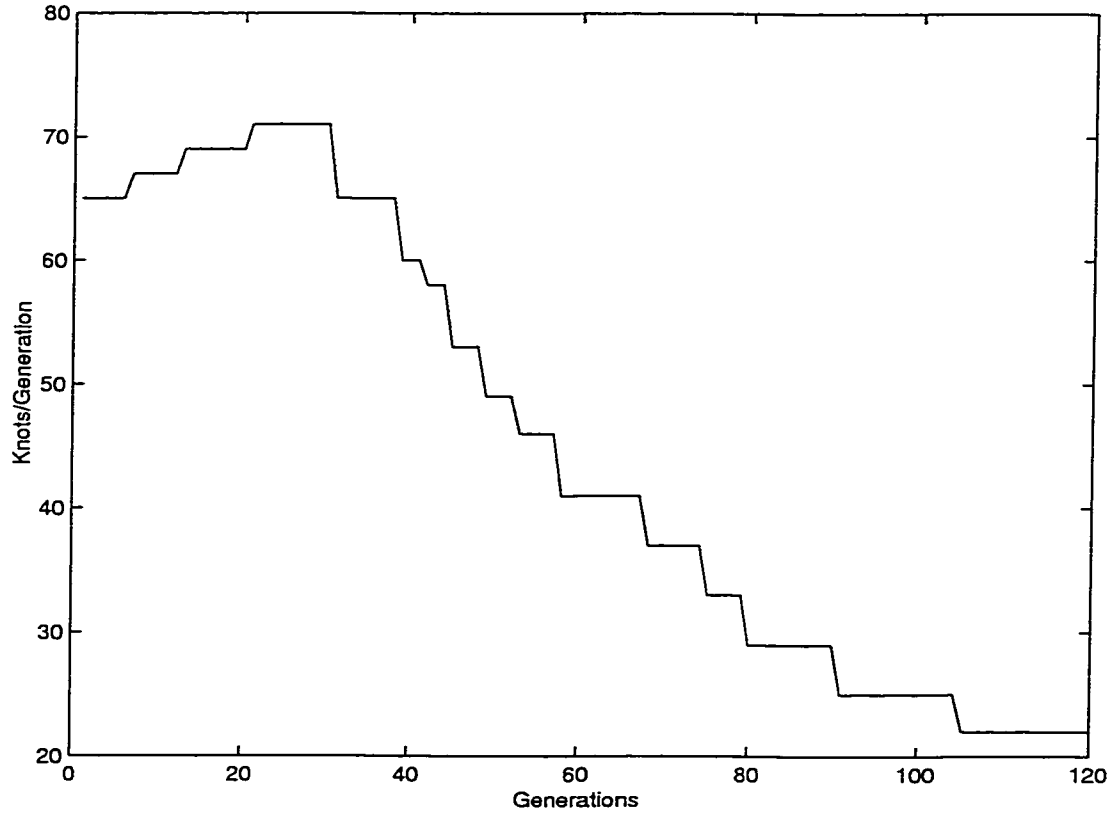

Figure 5.31: Knots Plotted against the Number of Generations

5.15 Surface Fitting

A step ahead of data fitting for curves is the surface fitting over a mesh of data. The scheme we used in our algorithm for surface fitting is to calculate individual NURBS curves by considering the rows and columns of the mesh data one by one. Thus calculations are done along parameter t and the parameter u one after the other. First the calculations are done along the parameter t by taking the rows of the mesh data one by one. Then the columns are considered one by one for the calculations along the parameter$u$. This scheme makes the surface fitting problem similar to the curve fiting. The difference is that in this case we have to consider the optimization of the individual curves of the mesh data one by one and iterate over the total number of generations of our GA to get an optimized mesh. Figure 5.32 shows the basic building blocks of our implemented System for surface fitting.

The algorithm was tested for the surface shown in Figure 5.33.The control parameters used are given in Table (5.5).

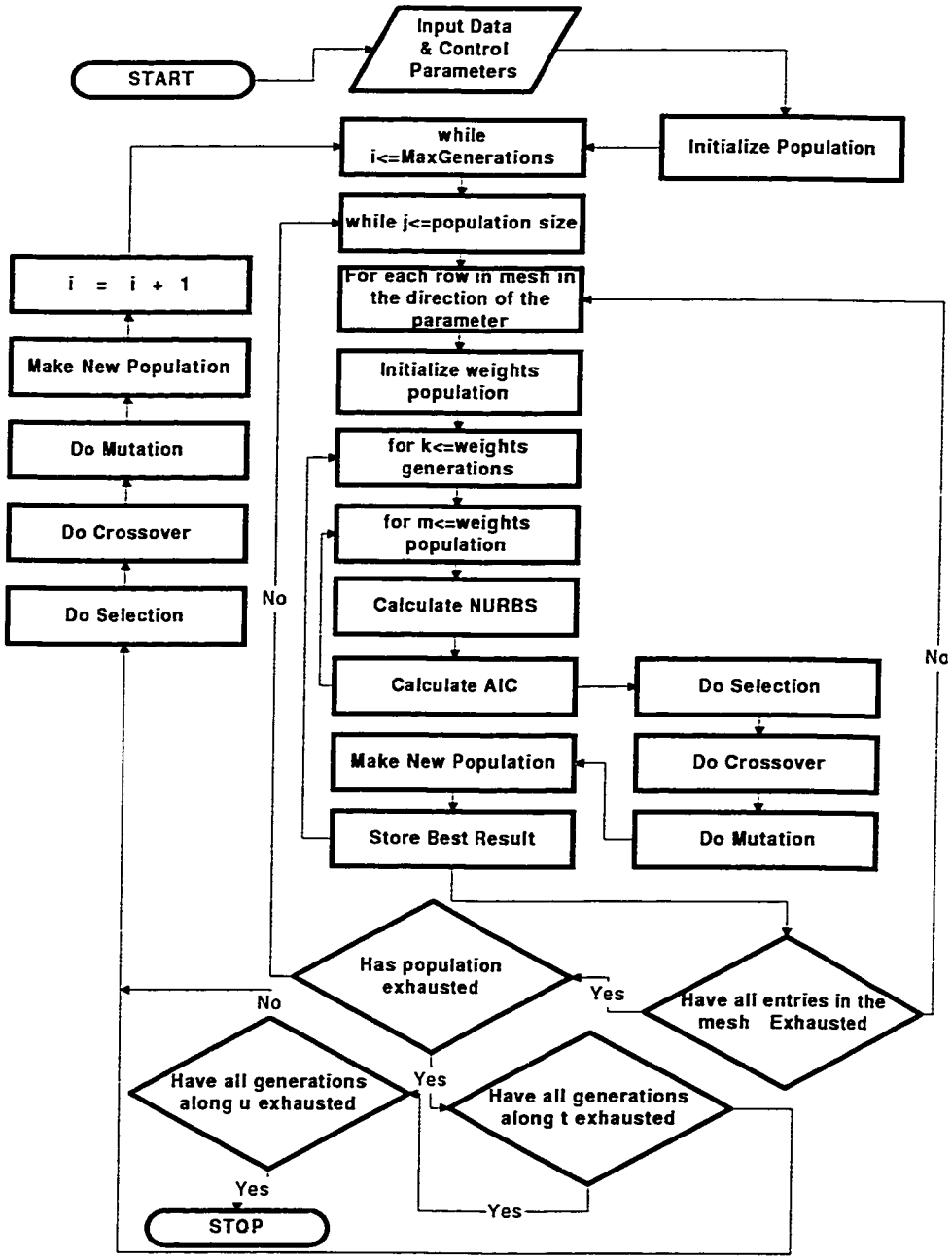

Figure 5.32: Flow Diagram of Data Fitting System for Surfaces

Table 5.5: Input Parameters for Surface Fitting

Mesh size	30 x 30
Noise Variance	1
No. of Generations	200
Knot Ratio	0.5
Population size	35
Generations for weights	30
Population size for weights	25
Crossover rate	0.7
Mutation rate	0.001
Selection	Roulette-wheel

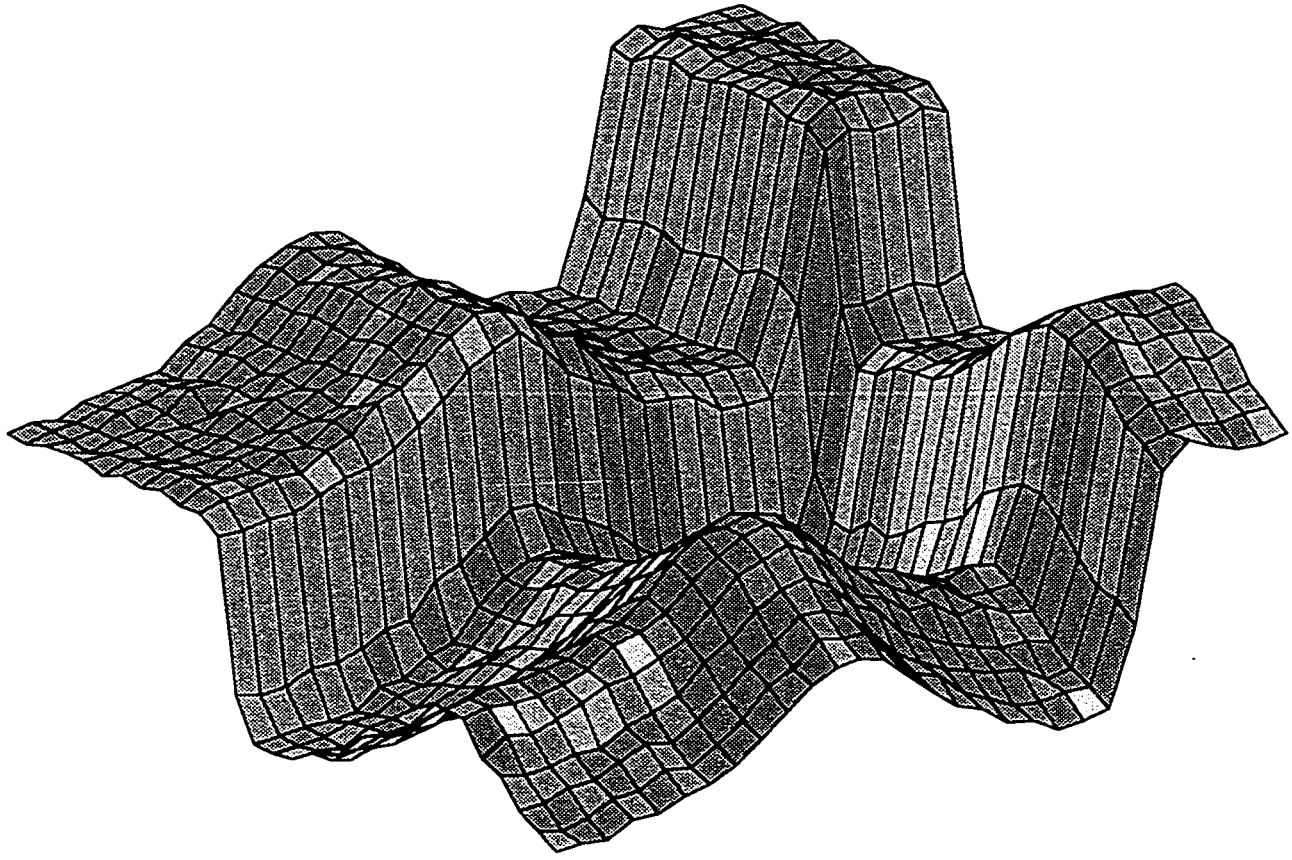

Figure 5.33: Input Surface for Data Fitting

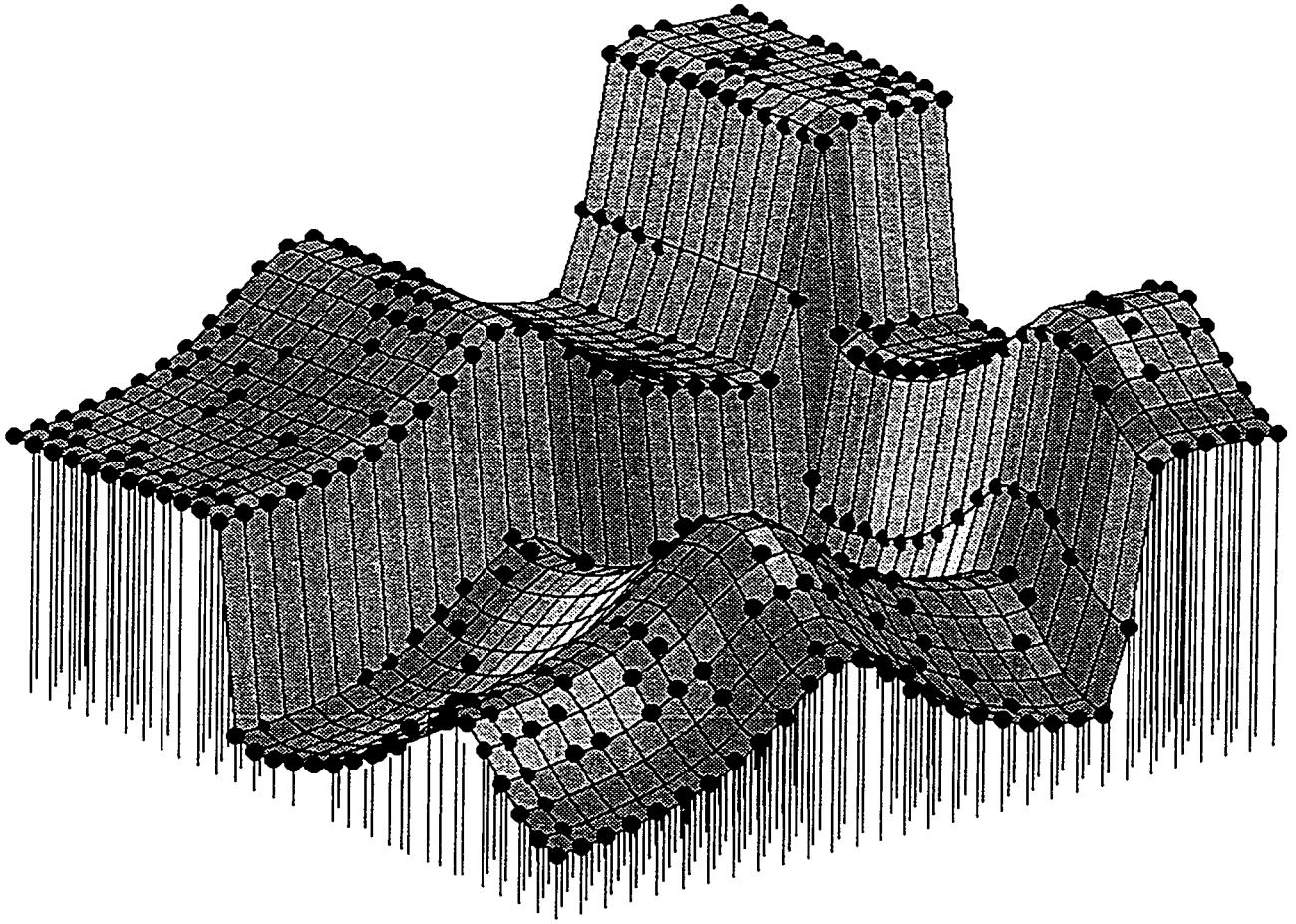

Figure 5.34: Approximated Surface after Convergence

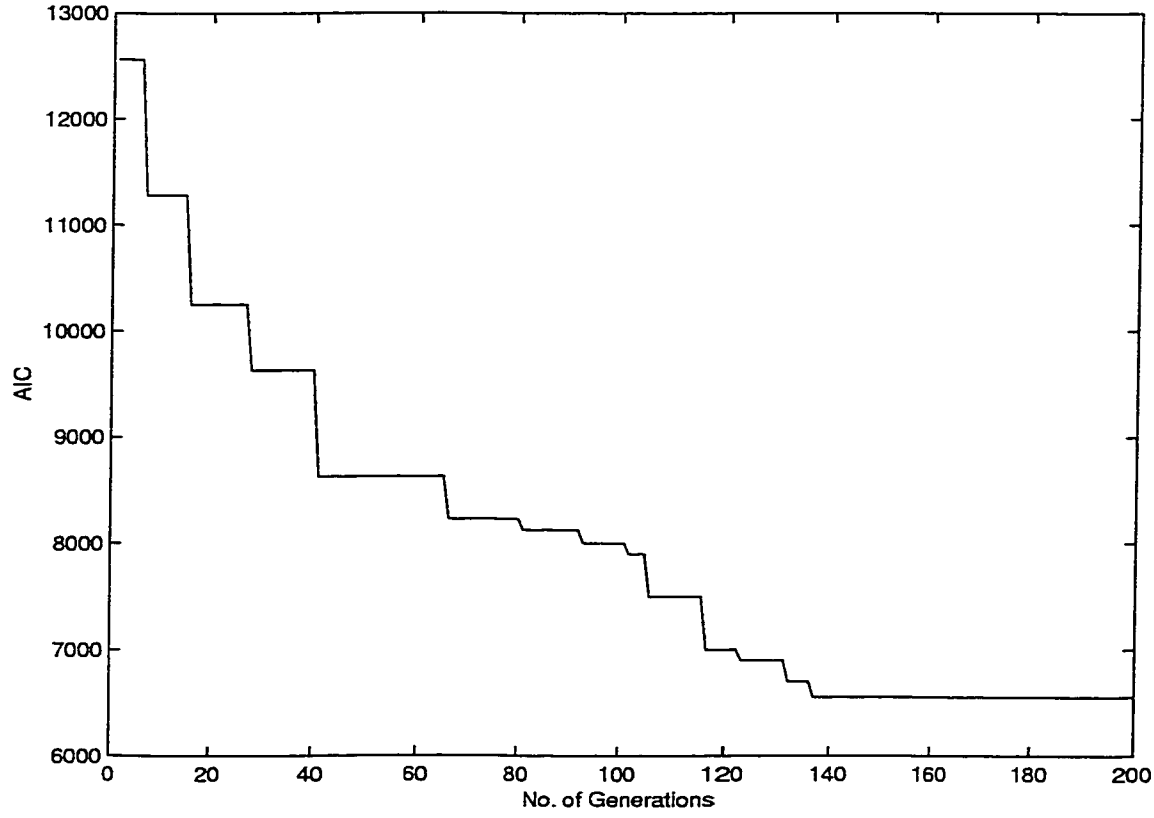

Figure 5.35: AIC Plotted against the Number of Generations

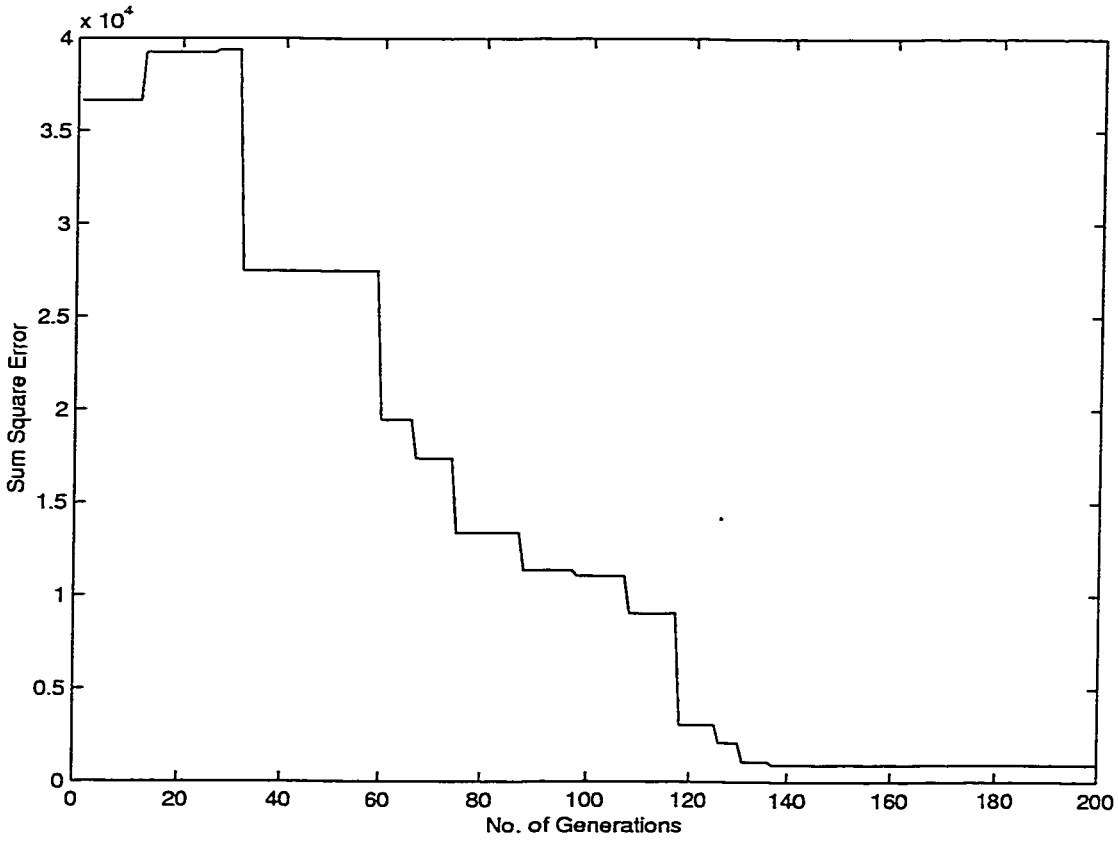

Figure 5.36: Sum Square Error Plotted against the Number of Generations

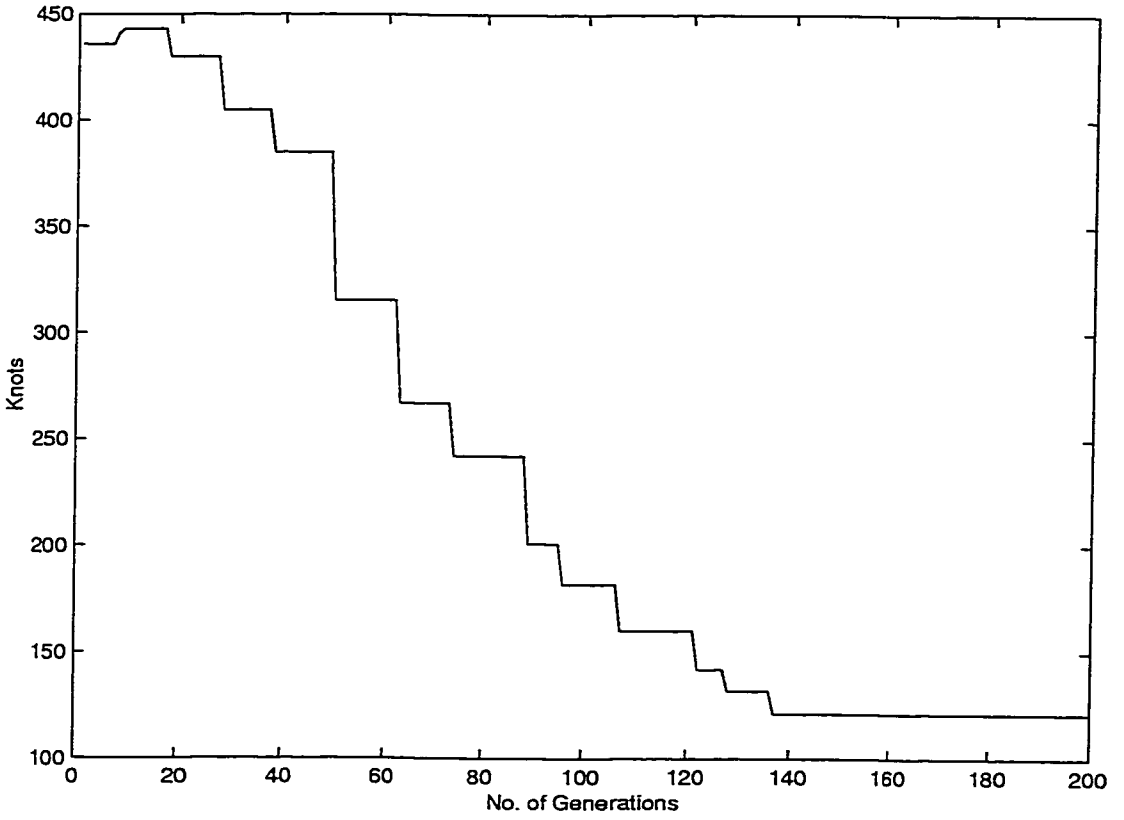

Figure 5.37: Knots Plotted against the Number of Generations

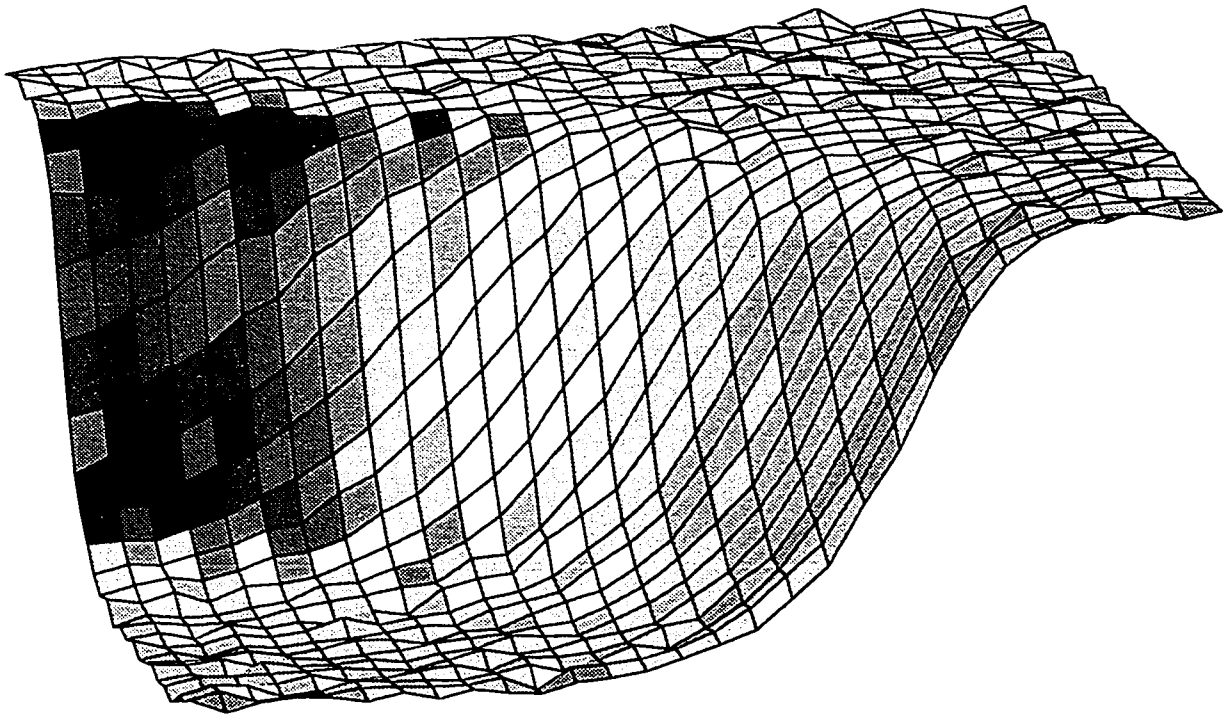

Figure 5.38: Input Noisy Surface

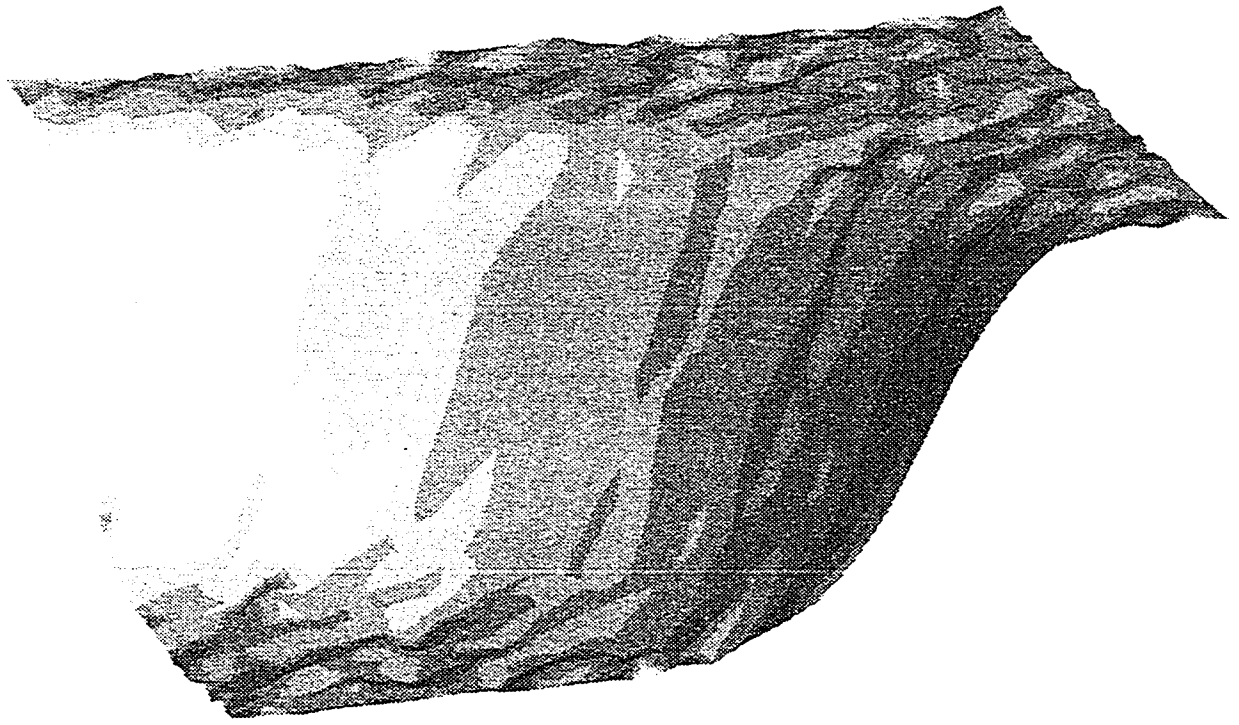

Figure 5.39: Input Noisy Surface after Shading

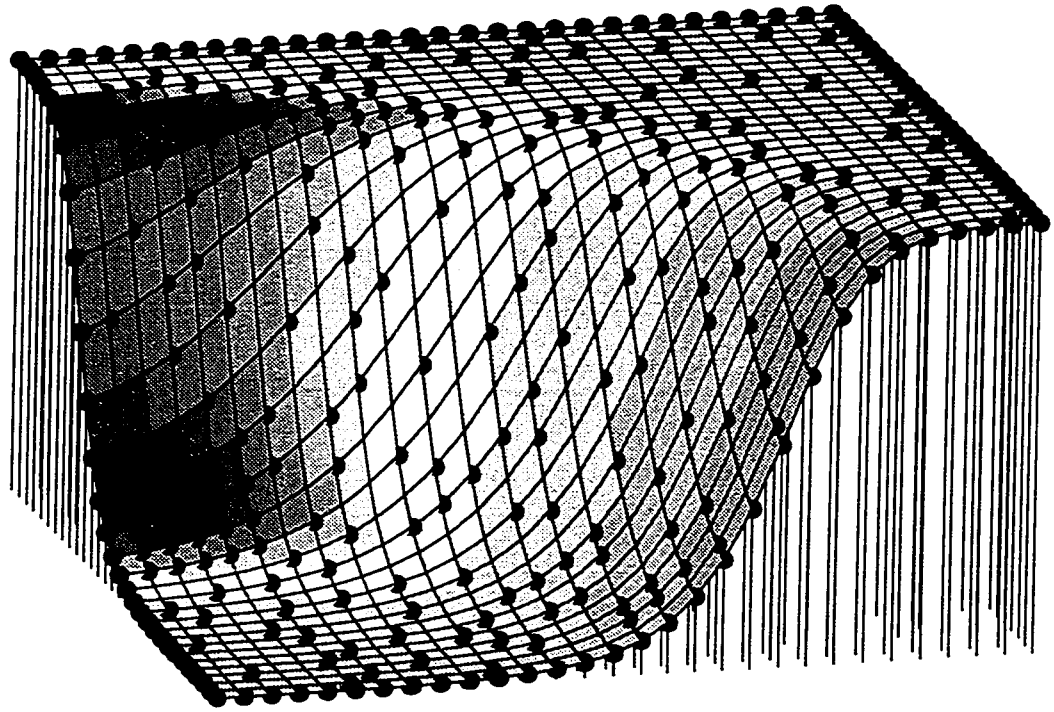

Figure 5.40: Approximated Surface after Convergence

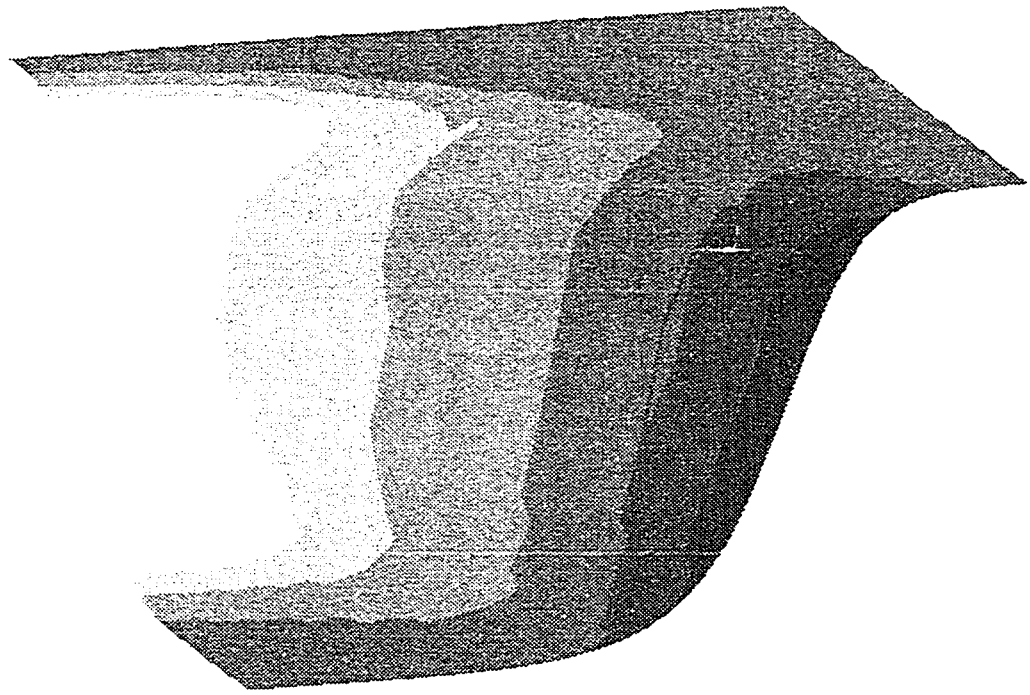

Figure 5.41: Approximated Surface after Shading

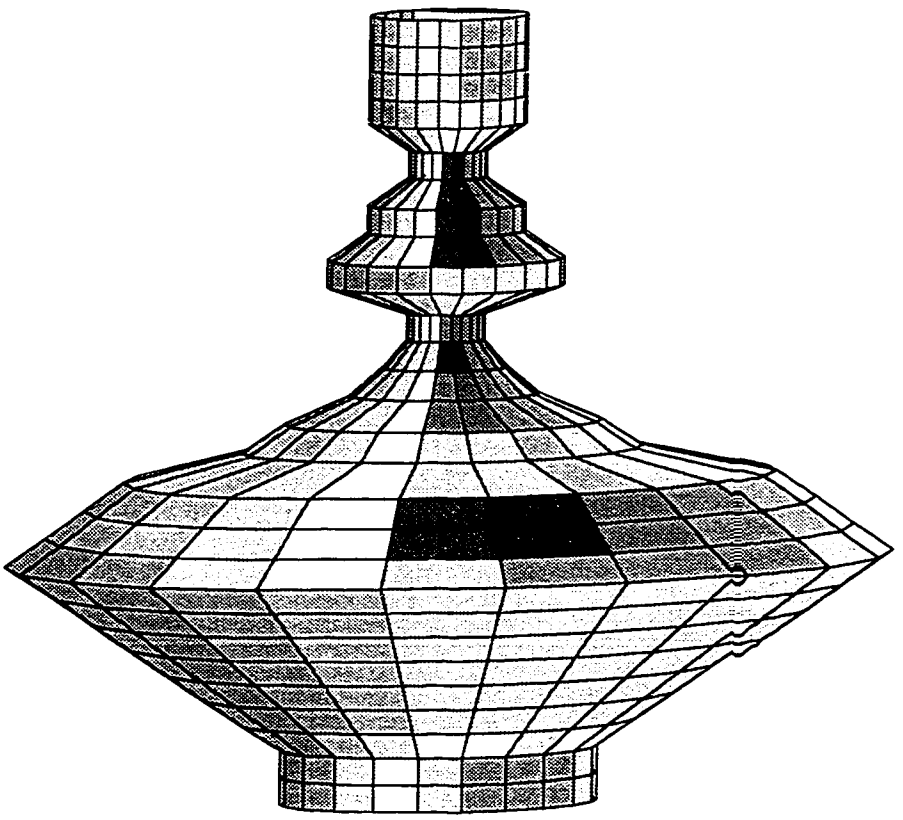

Figure 5.42: Input Surface (A Flask)

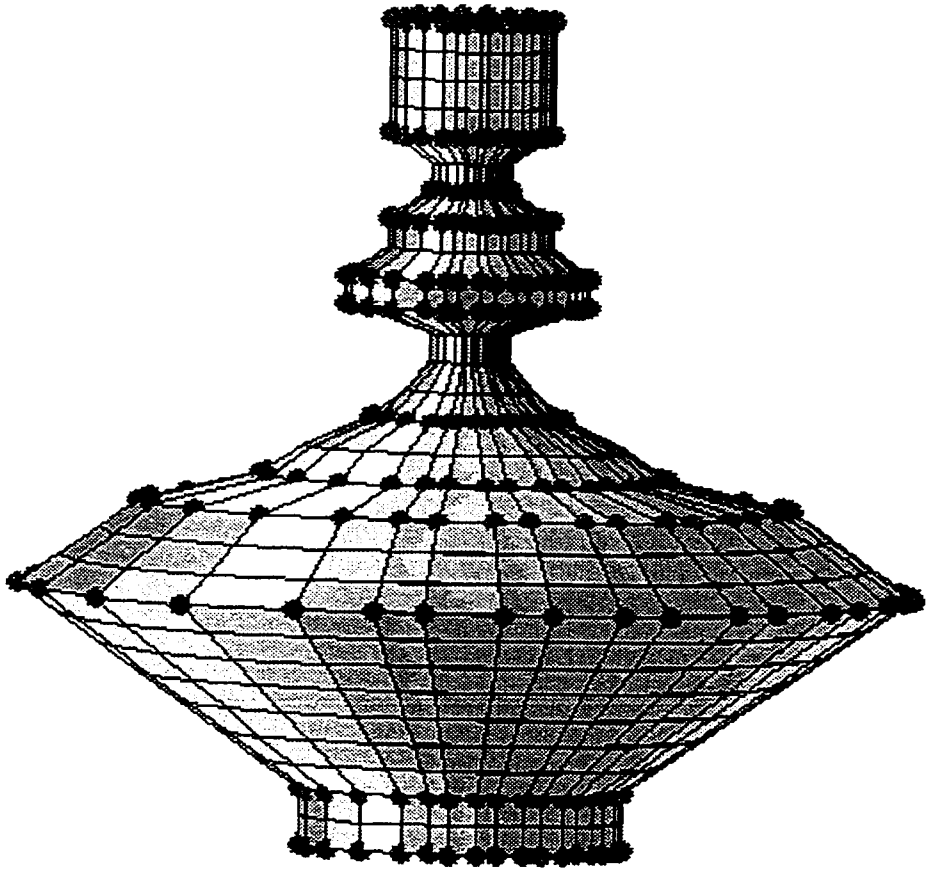

Figure 5.43: Approximated Surface after Convergence

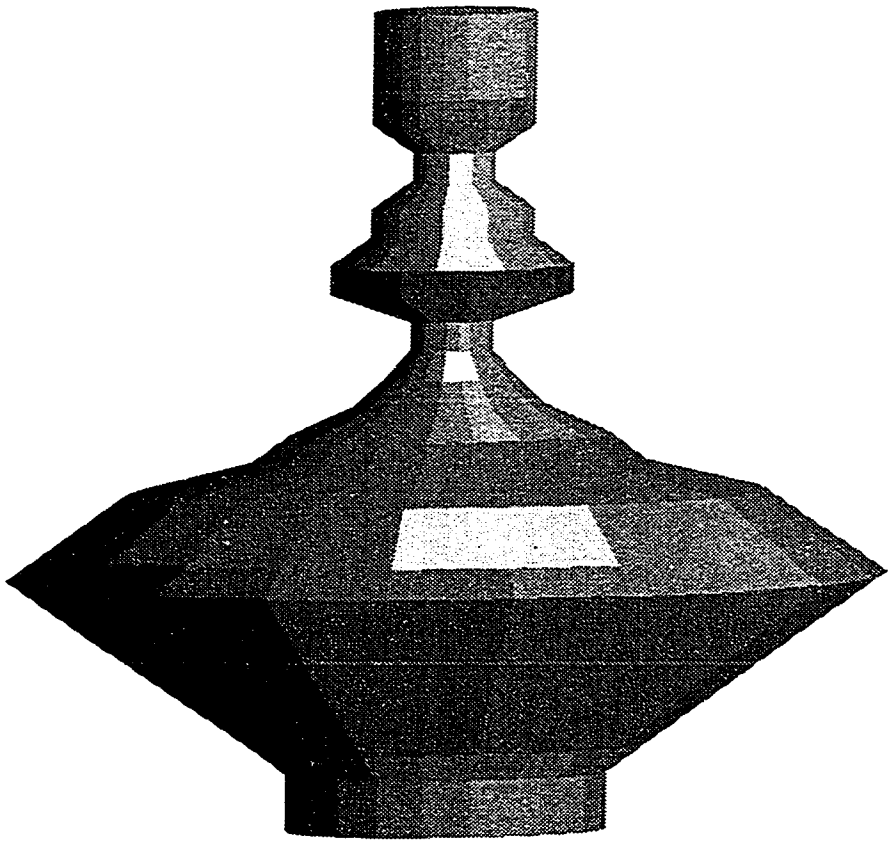

Figure 5.44: Approximated Surface after Shading

Chapter 6

Evaluation and Analysis

6.1 Representation of Fonts

There are two fundamental techniques for storing fonts in computer.

1. Bitmap

2. Outline

6.1.1 Bitmap

A bitmap is characterized by the width and height of the image in pixels. Number of bits per pixel determines number of shades of gray or colors that can be represented. For example one pixel per bit image has only two possible values of bit i.e. bit is either on (1) or off (0). Thus it has only two possible colors (black or white). Bitmap

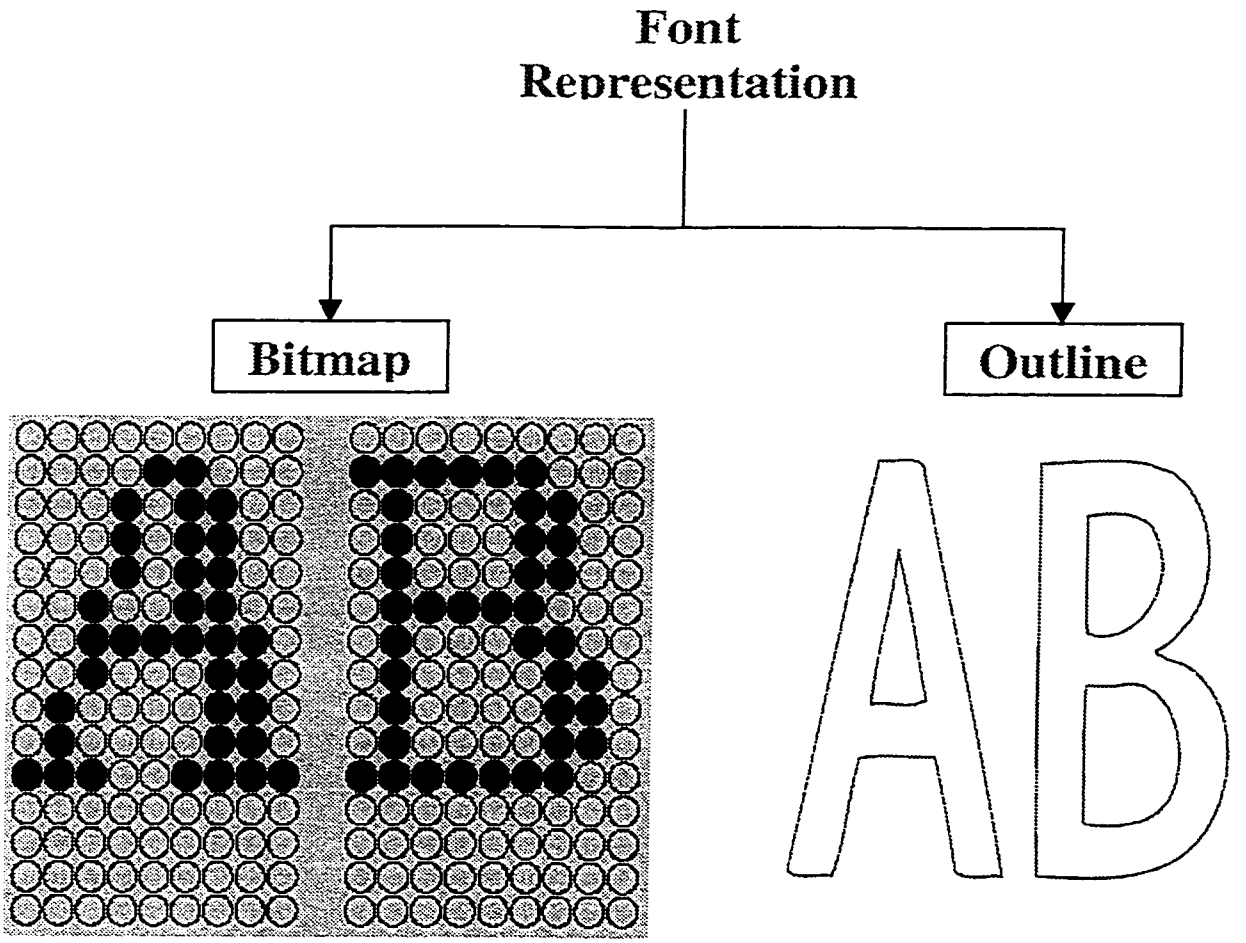

Figure 6.1: Font Representation Methods

graphics are referred as raster graphics.

In this method each character in a given font is specified as a small rectangle bitmap. Information about height of the characters and the amount of space to be placed between adjacent characters is also stored. Generating a character requires to copy those pixels values from font cache into the frame buffer and then on to the screen at the desired position.

The bitmaps for the font cache are usually created by scanning in enlarged pictures of characters from typesetting fonts in various sizes. A paint program is used to specify the individual pixels in each character's bit map. Alternatively, the type designer can use a paint program to create fonts from scratch. Since small bitmaps do not scale well, it is necessary to define more than one bit map for each character to provide various standard sizes.

6.1.2 Outline Representation

Fonts can be represented by outline using spline. Splines are piecewise polynomial parametric curves, generated by varying a parameter over a specified range. If $x(t)$ and $y(t)$ are two functions that supply points $(x(t), y(t))$ along a curve as t is varied then mathematically we can write:

$$x(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + \ldots + b_n t^n \tag{6.1}$$

where a_i and b_i are coefficients and n is the order of the polynomial

In equation (6.1) when n is 2, the polynomials are called *conic*; when n is 3, they are *cubic*.

6.1.3 Drawbacks of Bitmap Representation of Fonts

- It requires a distinct font cache for each combination of font, size, and face for each different resolution of display or output device. For example a single font in six different point sizes and four faces (normal, bold, italic, bold italic) requires 24 font caches.
- Bitmap representation of font behaves unpredictably when subject to Affine Transformations, thereby giving an unacceptable shape and resolution.
- Bitmap fonts always appear the same even if the resolution of device is enhanced

6.1.4 Advantages of Outline Representation

- In the outline representation characters are represented in an abstract device independent way.

- Piecewise polynomial functions i.e. splines provide smooth curves.

- These splines are used to store the outline of characters. Although each character definition may take more space than its representation in a font cache but multiple sizes may be derived from a single stored representation by suitable scaling.

- Different type faces can also be derived e.g. italics may be derived by shearing the original outline.

- Another advantage of storing characters in a device independent form is that the outlines may be arbitrarily translated, rotated, scaled, and clipped.

- In addition to all the above advantages, instead of saving all the points along the splines, some important points can be saved so that the font outline could be reproduced by fitting the spline to those saved points.

  These points in our system have been termed as it knots. The following table shows the total number of points and the knots obtained after running our algorithm.

Table 6.1: Contour Data and the Knots Obtained

Figure #	# of Boundary Points	Points after Decimation	# of Corners	# of knots
5.5	1640	410	12	54
5.8	689	344	11	33
5.11	320	160	9	22

According to the Table (6.1), in case of 'Ali', 54 knots constitute 13% of the decimated boundary points. It shows that 83% saving in terms of boundary points is obtained.

Similarly, in case of 'Pound', 33 knots make 9.5% of the decimated boundary points, which consequently results in 90.5% saving in terms of points.

In case of 'Aich', 22 knots make about 14% of the decimated boundary points, which in turn results in a saving of 86% in terms of boundary points.

6.2 Surface Representation

In the case of surface,as shown in the Table (6.2)

So, in this case 121 knots make about 13.5% of the total input data, which results in saving of about 86.5% in terms of data points.

6.3 Comparison with the Yoshimoto's Algorithm

The paper titled *"Automatic Knot Placement by a Genetic Algorithm for Data Fitting with a Spline"* [34] presents a GA-based algorithm with an objective of simultaneous determination of the number of knots and their locations. In this work, they have used a non-parametric form of spline with no optimization of weights associated with the control points.

In our work, we have developed and implemented a *"Nested Genetic Algorithm"*,

in which we have used NURBS(parametric form). The NURBS are the splines which can define conic section. That is why they have been accepted as a standard in various Computer Graphics packages like IEGS,PHIGS+. The advantage with the parametric form is that is can represent a closed figures. In additions to this, our work also deals with the optimization of weights, which play an important role in the modification of shapes.

This shows that our algorithm is more flexible and can be applied in a more general fashion to every kind of shapes including curves and surfaces.

Table 6.2: Input Data Size and the Knots Obtained

Mesh size	30×30
Noise Variance	1
# ok knots obtained	121

Chapter 7

Conclusion and Future Work

7.1 A Bird's Eye View of the Implemented System

A method for datafitting with NURBS both for curves and surfaces using genetic algorithm has been developed. In case of curves, we tested and presented results in the context of fonts but the method is equally applicable to any data (noisy or noiseless).

The most important aspect of our method is the automatic determination of adequate number and location of knots simultaneously. The method also determines the weights associated with the control points.

In case of fonts, the some important steps of our system are:

- Input the control parameters
- Getting digitized image
- Extracting Contour (Boundary)
- Detecting Corner Points
- Conservation of corner points
- Application of Nested Genetic Algorithm for NURBS fitting in a iterative manner.

The objective of our curve fitting process is to approximate the digitize curve with parametric curve in the best way. Our genetic algorithm is aided by a corner detection algorithm so that these points could not be lost in the subsequent generations as they are important in capturing fonts' outline with required accuracy.

In case of surfaces, the important steps of the system are:

- Input mesh data
- Input the control parameters
- Application of Nested Genetic Algorithm for NURBS fitting along parameter t.
- Application of Nested Genetic Algorithm for NURBS fitting along parameter u.

Both the systems have been designed in a modular way. The output of one module is input to next module. Because of the modular design approach any module can be modified without effecting other modules of the system. This makes our system flexible and adaptable for future enhancements.

7.2 Future Work

- **Distributed/Parallel Processing**: The nature of our system makes it suitable for distributed/parallel processing. If the character has more than one piece then distributed/prallel processing can be applied at piece level. Corner detection can be done separately for each piece. Also the calculation of NURBS or the fitness values for a group of chromosomes can be assigned to separate processors to gain significant speedup.

- **Parallel Genetic Algorithm**: It will be interesting to investigate and extend the method to Parallel Genetic Algorithm.The population can be divided into subpopulations each assigned to a separate processor running genetic algorithm of its own and then fitter chromosomes could be transferred between them through migration to enhance the search.

- **Other Heuristics**: The algorithm can be tested using other heuristics like Tabu Search or Simulated Annealing can be applied.

Bibliography

[1] H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, AC-19(6):716–723, 1974.

[2] W.S. Yoo B.K. Choi and C.S. Lee. Matrix representation of NURBS Curves and Surface. *Computer Aided Design*, 22(4), 1990.

[3] Dmitry Chetverikov and Zsolt Szabo. A Simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves. *Proc. 23rd Workshop of the Australian Pattern Recognition Group*, pages 175–184, 1999.

[4] E. R. Davies. Locating objects from their point features using an optimal hough-like accumulation technique. *Pattern Recognition Lett*, 13:113–121, 1992.

[5] P. Dierckx. *Curve and Surface Fitting with Splines.* Oxford University Press, 1993.

[6] L. Dreschler and H. Nagel. Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequence of a street scene. *Proc. IJCSI*,

pages 692–697, 1982.

[7] Jos L. Libeiro Filho and Philip C. Treleaven. Genetic Algorithm Programming Environments. *IEEE Computer*, pages 28–43, June 1994.

[8] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[9] J.A. Gregory, M. Sarfraz, and P. K. Yuen. Interactive Curve Design using C2 Rational Splines. *Computers and Graphics*, 18(2):153–159, 1994.

[10] M. H. Han and D. Jang. The use of maximum curvature points for recognition of partially occluded objects. *Pattern Recognition*, 23:21–23, 1990.

[11] David L. B. Jupp. Approximation to data with splines with free knots. *SIAM J. Numer. Anal*, 15(2):328–343, 1978.

[12] Murtaza Ali Khan. An Efficient Font Design Method. Master's thesis, KFUPM, 2000.

[13] Frederick L. Kitson. An algorithm for curve and surface fitting using B-Splines. *IEEE Conference*, 1989.

[14] Y.K. Kwok and I. Ahmad. Efficient scheduling of Arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *Journal of parallel and distributed computing*, pages 58–77, 1997.

[15] Wayne Tiller Les Piegel. *The NURBS Book.* Springer, 2 edition, 1997.

[16] H. C. Liu and M.D. Srinath. Corner detection from chain-code. *Pattern Recognition*, pages 51–68, 1990.

[17] T. Lyche and K. Morken. A data reducttion strategy for splines wit applications to the approximation of functions and data. *IMA Journal of Numerical Analysis*, 8(2):185–208, 1988.

[18] G. Renner Markus, A. and J. Vancza. Spline interpolation with genetic algorithms. *Proc. Int. Conf. on shape, Modeling and applications, IEEE Computer Society Press*, pages 47–54, 1997.

[19] Lalit M. Patnaik M.Srinivas. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, 24(4):656–667, 1994.

[20] Lalit M. Patnaik M.Srinivas. Genetic Algorithms: A Survey. *IEEE Computer*, pages 17–26, June 1994.

[21] Les Piegel. On NURBS: A Survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, Jan. 1991.

[22] M. J. D. Powell. *Curve fitting by splines in one variable, in one variable, in Numerical Approximation to Functions and Data,.* Athlone Press, London, j. g. hayes (ed.) edition, 1970.

[23] Toni Prahasto and Sanjeev Bedi. Optimization of knots for the multi curve B-spline approximation . *IEEE Conference*, 2000.

[24] Azhar Quddus. *Curvature Analysis Using Multiresolution Techniques.* PhD thesis, KFUPM, 1998.

[25] John R. Rice. *Numerical Methods, Software and Analysis.* Academic Press,New York, second ed. edition, 1993.

[26] Javier Sanches-Rayes. A simple technique for NURBS shape Modification. *IEEE Computer Graphics and Applications*, Jan-Feb 1997.

[27] M. Sarfraz. A C2 rational cubic spline alternative to the NURBS. *Computers and Graphics*, 116(1):69–77, 1992.

[28] M. Sarfraz. Cubic Spline Curves with Shape Control. *Computers and Graphics*, 18(5):707–713, 1994.

[29] M. Sarfraz, M. N. Haque, and M. A. Khan. Capturing outlines of 2d images. *PDPTA' 2000 Conference, Las Vegas, USA.*, 2000.

[30] M. Sarfraz and M. A. Khan. Towards automation of capturing outlines of arabic fonts. *Proceeding of WICS 2000 Workshop*, 2000.

[31] Yung-Nien Sun Shu-Chien Huang. Polygonal Approximation using genetic algorithm. *IEEE Conference*, 1996.

[32] Miroslav Trajkovic and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, February 1998.

[33] M.Moriyama Yoshimoto, F. and T.Harada. A method of plane-data fitting with a genetic algorithm. *Proc. of the Iasted International Conference on Computer Graphics and Imaging*, pages 21–31, 1998.

[34] M.Moriyama Yoshimoto, F. and T.Harada. Automatic Knot Placement by a Genetic Algorithm for Data Fitting with Spline. *IEEE Conference*, 1999.

[35] T. Harada Yoshimoto, F. and Y. Aoyama. Data Fitting using a genetic algorithm with real number genes. *Proc. of the Iasted International Conference on Computer Graphics and Imaging*, 2000.

Vita

- Syed Arshad Raza.
- Born in Hyderabad, Pakistan on August 10, 1973.
- Received Bachelor of Engineering (B.E) degree in Mechanical Engineering from N.E.D University of Engineering and Technology, Karachi, Pakistan in 1997.
- Joined King Fahd University of Petroleum and Minerals in September 1998.
- Publication: Sarfraz, M.,and Raza, S. A.(2001),"Capturing outline of Fonts using Genetic Algorithm and Splines',To appear in the Proceedings of IEEE International Conference on Information Visualization,being held in UK.
- Email: saraza@ccse.kfupm.edu.sa