A Study on B-Spline & β-Spline and their Applications in Image Compression Problem.

A dissertation submitted in partial fulfillment of
the M.Tech.(Computer Science) degree of
The Indian Statistical Institute.

by

Arindam Das

under the supervision of

Mr. S. N. Biswas.
Machine Intelligence Unit.
Indian Statistical Institute.

July 28, 1997

CERTIFICATE

This is to certify that the thesis entitled *"A Study on B-spline and β-spline with their applications in Image Compression Problem"*, which is submitted by **Arindam Das** ,final year student of **Computer Science** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology(Computer Science)** ,embodies the work carried by him during the year **1996-97**, under my guidance.This thesis fulfills the purpose for which it was written.

Mr. S. N. Biswas
MIU
I.S.I.Calcutta

Date: 28 July, 1997

ACKNOWLEDGEMENTS

I wouuld like to thank the following persons for the support and inspiration they have provided me during the course of my work :

Dr. Probal Sengupta : Who helped me during the mid-term evaluation, and allowed me to work in the CVPR Silicon Graphics computer.

CSSC : Who was finally able to load the X Window System in the Sun workstation.

Companions : To all my class-mates.

ARINDAM DAS
MTC9502

Date: 25 July, 1997
Place: I.S.I.Calcutta

Index

Chapter 1

A Brief Introduction To B-Spline & β-Spline

A *spline* is defined analytically as a set of polynomials over a knot vector. A *knot vector* is a vector of real numbers, called *knots*, in nondecreasing order; that is

$$u = [u_0, u_1, \ldots, u_q]$$

such that,

$$u_{i-1} \leq u_i, i = 1, 2, \ldots, q$$

The two types of spline surfaces studied are the B-spline and the β-spline. Of the B-splines studied, only *Nonrational* B-splines are considered.

1.1. B-splines

A spline of order k (degree $k-1$) is defined mathematically as a piecewise $(k-1)^{st}$-degree polynomial that is C^{k-2} continuous; i.e., it is a polynomial of degree at most $k-1$ on each interval $[u_{i-1}, u_i)$, and its position and first $k-2$ derivative are continuous.

1.1.1. B-spline curve

The i^{th} B-spline basis function of order k(degree $k-1$) for the knot vector $[u_i, \ldots, u_{i+k-1}]$ will be denoted $N_{i,k}(u_i, \ldots, u_{i+k-1}; u)$, and can be expressed as the following recurrence relation :

$$N_{i,1}(u_i, u_{i+1}; u) = \begin{cases} 1 & \text{for } u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

$$N_{i,k}(u_i,\ldots,u_{i+k-1};u) = \frac{u-u_i}{u_{i+k-2}-u_i}N_{i,k-1}(u_i,\ldots,u_{i+k-2};u) + \frac{u_{i+k-1}-u}{u_{i+k-1}-u_{i+1}}N_{i+1,k-1}(u_{i+1},\ldots,u_{i+k-1};u),$$
$$\text{for } k>1 \;\&\; u_i \le u < u_{i+1} \quad (1.2)$$

In other words, the above equation means that the B-spline of order k in the i^{th} span is the weighted average of the B-splines of order $k-1$ on the i^{th} and $(i+1)^{st}$ spans, each weight being the ratio of the distance between the parameter and the end knot to the total length of the $k-1$ spans.

Although, the values of the knots are so unconstrained, an especially useful special case is that of uniform knot spacing, where $u_i = i$. For the case $k=4$, this generates the uniform cubic B-spline basis function :

$$N_{i,4} = \begin{cases} 0 & u'_i < 0 \\ \frac{1}{6}u'^3_i & u'_{i+1} < 0 \le u'_i \\ \frac{1}{6}(1+3u'_{i+1}+3u'^2_{i+1}-3u'^3_{i+1}) & u'_{i+2} < 0 \le u'_{i+1} \\ \frac{1}{6}(4-6u'^2_{i+2}+3u'^3_{i+2}) & u'_{i+3} < 0 \le u'_{i+2} \\ \frac{1}{6}(1-u'_{i+3})^3 & u'_{i+4} < 0 \le u'_{i+3} \\ 0 & 0 \le u'_{i+4} \end{cases} \quad (1.3)$$

where, $u'_{i+j} = u - u_{i+j}, \qquad j = 0,\ 1,\ 2,\ 3.$

1.1.2. B-spline surface

B-spline surface is a generalization of the B-spline curve. Here, from a 2-dimensional array of control points $\{\mathbf{V}_{i,j}\}^{m\times n}$, a B-spline surface, composed of a set of surface patches, is constructed. The k^{th}-order, i^{th} basis function for the parameter u is $N_{i,k}(u)$, which is the same as that used for B-spline curves. The $(i,j)^{th}$ patch is given by the equation :

$$\mathbf{Q}_{i,j}(u,v) = \sum_{r=0}^{k-1}\sum_{s=0}^{l-1} \mathbf{V}_{i+r,j+s}\ N_{i+r,k}(u)\ N_{j+s,l}(v) \quad (1.4)$$

$$0 \leq u \leq 1, 0 \leq v \leq 1; i = 0, \ldots, m-k-2; j = 0, \ldots, n-l-2;$$

1.2. β-splines

The β-spline is a departure from B-spline in that its derivation is based on fundamental geometric measures rather than abstract algebraic quantities. To be specific, two new degrees of freedom — *bias* and *tension* — are obtained by means of a cubic β-spline. For appropriate values of bias and tension, the β-spline reduces to a uniform cubic B-spline, demonstrating that the β-spline is actually a generalization of the B-spline.

1.2.1. β-spline curve

Given a control polygon in 3-space : $[\mathbf{V}_0, \mathbf{V}_1, \ldots, \mathbf{V}_m]$, the β-spline curve defined by the sequence is composed of $m-2$ curve segments, the i^{th} of which is given by :

$$\mathbf{Q}_i(u) = \sum_{r=-2}^{1} \mathbf{V}_{i+r} b_r(\beta_1, \beta_2; u), 0 \leq u \leq 1, i = 2, \ldots, m-1 \qquad (1.5)$$

The expression $b_k(\beta_1, \beta_2; u)$ is a cubic polynomial called the k^{th} β-spline basis function. The four basis functions are derived so as to preserve continuity of the unit tangent and curvature vectors. For all values of β_1 and β_2, the basis functions have the following representation :

$$b_{-2}(\beta_1, \beta_2; u) = \frac{1}{\delta} 2\beta_1^3 (1-u)^3 \qquad (1.6)$$

$$\begin{aligned} b_{-1}(\beta_1, \beta_2; u) = \frac{1}{\delta}(&2\beta_1^3(u^3 - 3u^2 + 3u) + 2\beta_1^2(u^3 - 3u^2 + 2) \\ &+2\beta_1(u^3 - 3u + 2) + \beta_2(2u^3 - 3u^2 + 1)) \end{aligned} \qquad (1.7)$$

$$b_0(\beta_1, \beta_2; u) = \frac{1}{\delta}(2\beta_1^2(-u^3 + 3u^2) + 2\beta_1(-u^3 + 3u)(1.8)$$

$$+\beta_2(-2u^3 + 3u^2) + 2(-u^3 + 1)) \qquad (1.9)$$

$$b_1(\beta_1, \beta_2; u) = \frac{1}{\delta} 2u^3 \qquad (1.10)$$

where,

$$\beta_1 = \text{Bias}$$

$$\beta_2 = \text{Tension}$$

$$\delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2 \qquad (1.11)$$

1.2.2. β-spline surface

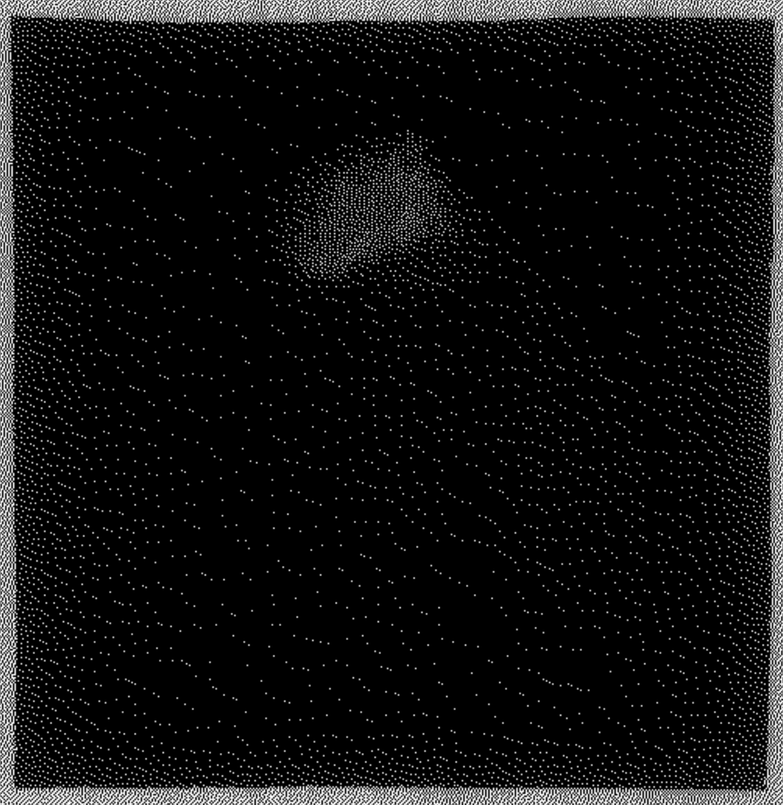
A β-spline surface is described by a control graph in 3-space :$[\mathbf{V}_{00}, \mathbf{V}_{01}, \ldots, \mathbf{V}_{mn}]$. In a manner analogous to the β-spline curve, a β-spline surface is composed of a set of surface patches, the $(i,j)^{th}$ patch being defined as :

$$\mathbf{Q}_{i,j}(u,v) = \sum_{r=-2}^{1} \sum_{s=-2}^{1} \mathbf{V}_{i+r,j+s} b_r(\beta_1,\beta_2;u)\ b_s(\beta_1,\beta_2;v) \qquad (1.12)$$

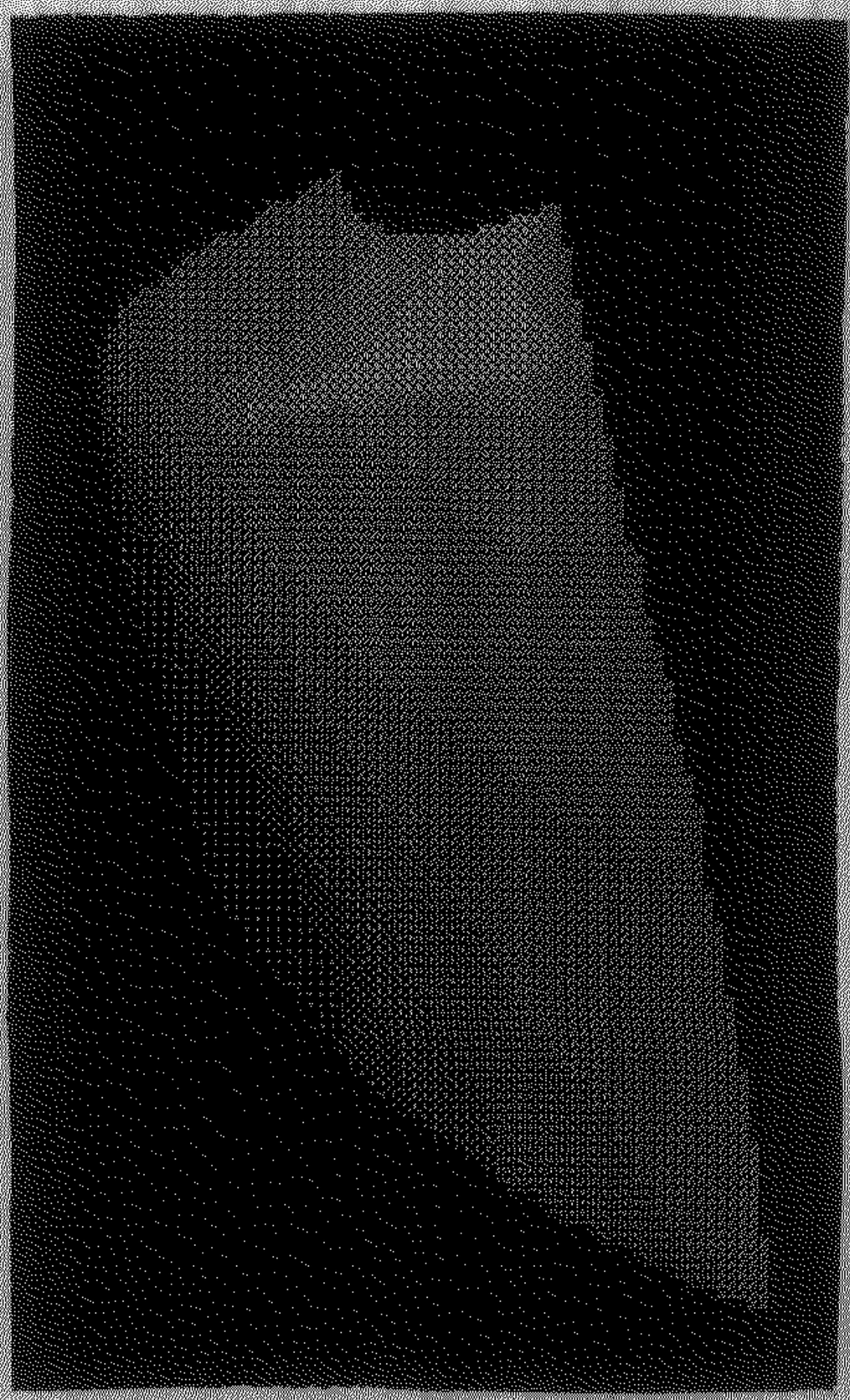
$$0 \le u \le 1, 0 \le v \le 1; i = 2,\ldots,m-1; j = 2,\ldots,n-1;$$

1.3. Result and Discussions

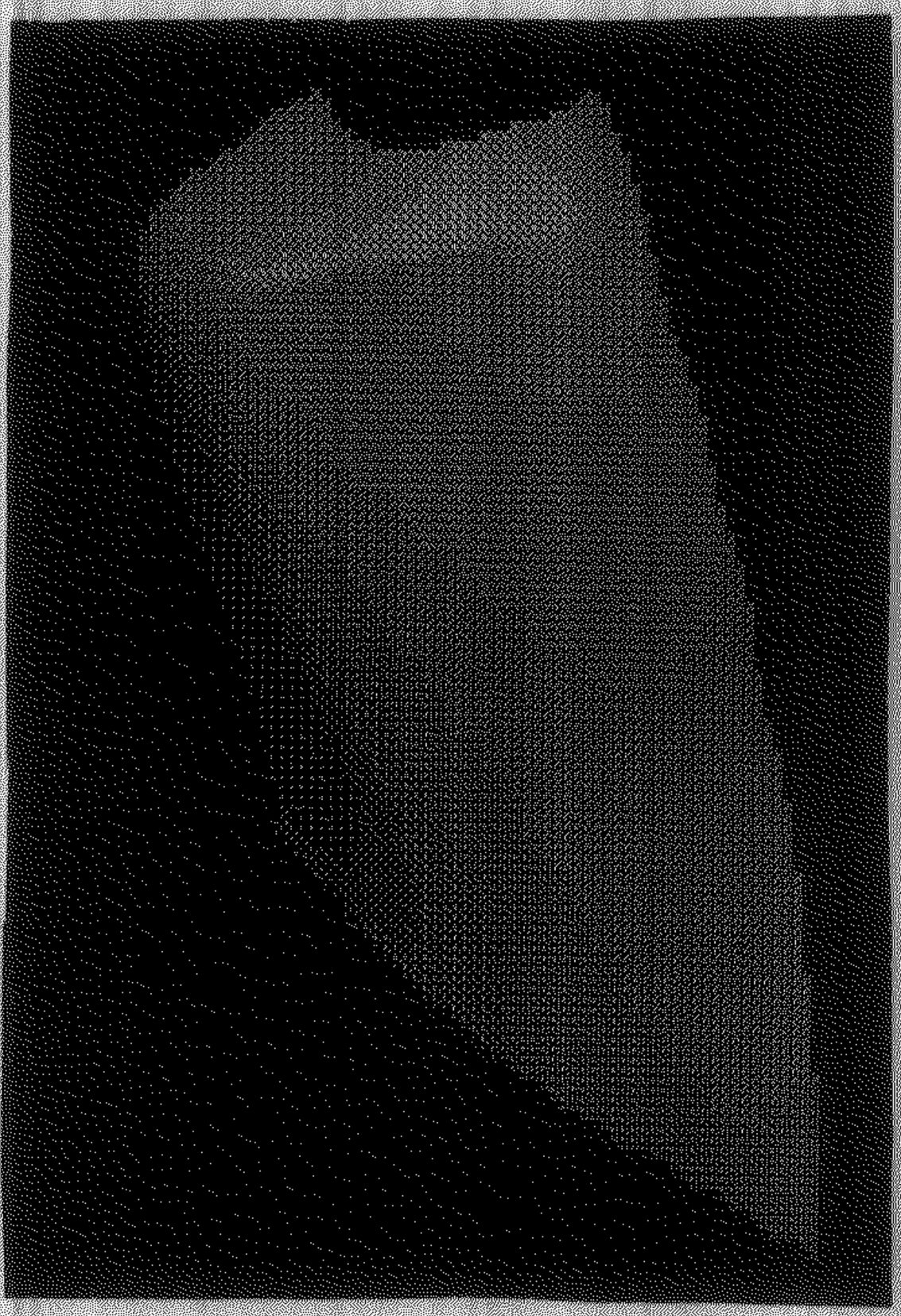
In computer graphics, free-form design of curves and surfaces are very often required. *B*-splines are extensively used to serve the purpose because of its local control. A cubic B-spline curve can be generated using eqn.1.3, while a cubic surface can be generated using eqn.1.3 and 1.4. To get curves and surfaces of any other degree, eqn. 1.1 and 1.2 can be used to get the basis functions.



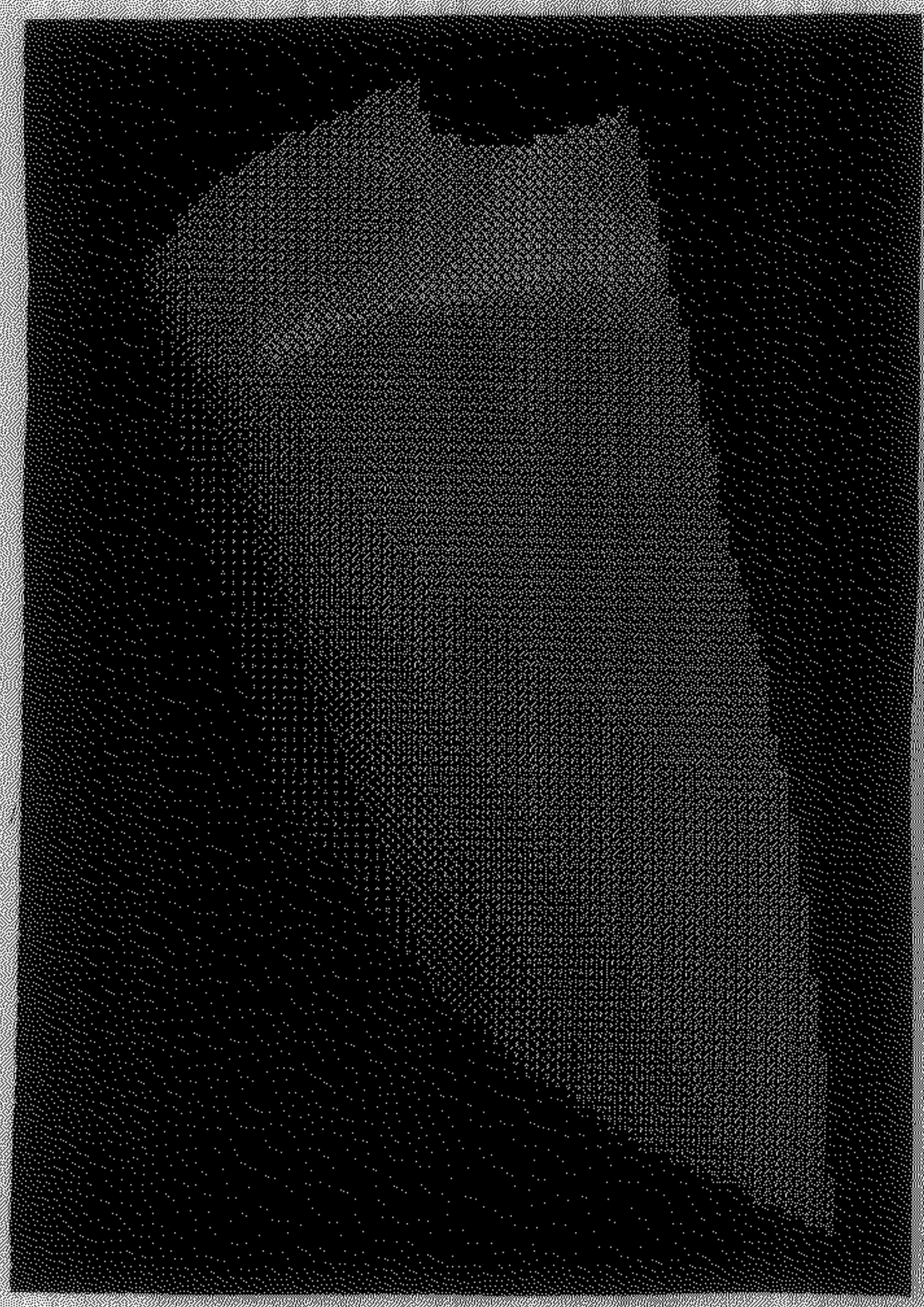
Cubic B-spline surface



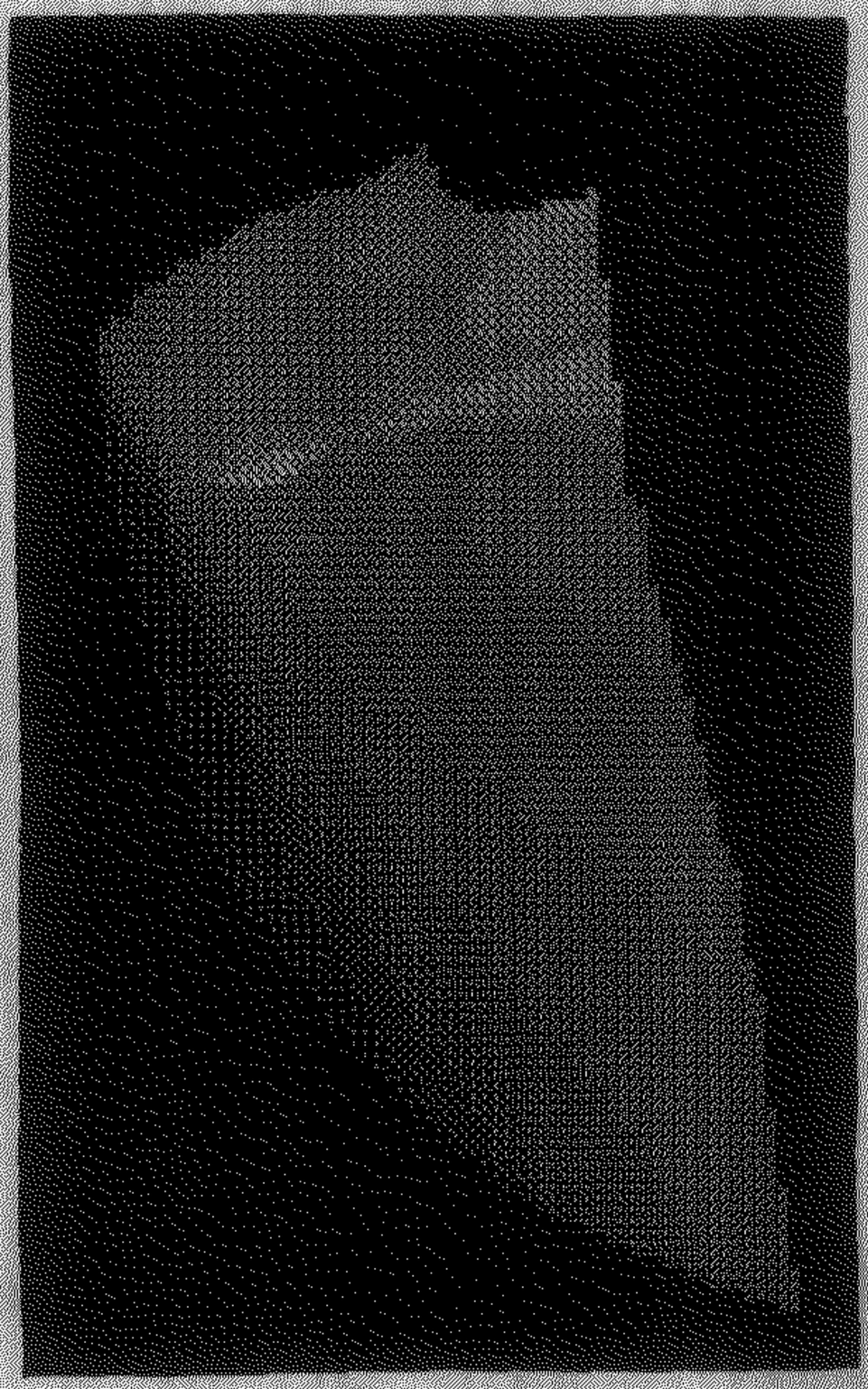
β-spline surface
$\beta_1 = 1, \beta_2 = 0$;



β-spline surface
$\beta_1 = 1$; $\beta_2 = 5$;



β-spline surface
$\beta_1 = 2,\ \beta_2 = 5$



β-spline surface
$\beta_1 = 0.2 \beta_2 = 5$)

Chapter 2

Image Segmentation

For application of spline surfaces in image compression problem, we first of all segment the image into uniform region and subsequently approximation by spline function. One approach to segmentation is *Region Growing*. In this method, a set of highly uniform regions, e.g., connected components of constant graylevel, or, more generally, regions extracted by some preliminary segmentation process, is constructed. Growth then takes place by starting with one of this regions and merging neighboring regions with it, one at a time. The choice of which neighbor to merge should depend both on the similarity of the regions and on the size and shape of the resulting merged region. The basic formulation for region-oriented segmentation is given below.

2.1. Basic Criteria for Segmentation

Let R represent the entire region. One may view segmentation as a process that partitions R into k subregions, $R_1, R_2, \ldots, R_k$, where the i^{th} region contains n_i pixels, $p_{i1}, p_{i2}, \ldots, p_{in_i}$, such that :

- $\diamond$ $\cup_{i=1}^{k} R_i = R$
- $\diamond$ R_i is a connected region, $i = 1, 2, \ldots, k$
- $\diamond$ $R_i \cap R_j = \phi, \qquad \forall i, j, \quad i \neq j$
- $\diamond$ $\wp(R_i) < \theta, \qquad \forall i = 1, 2, \ldots, k$

- $\wp(R_i \cup R_j) \geq \theta, \qquad \forall i \neq j$

where,

$$\theta = \text{The threshold for segmentation}$$

$$\wp(R_i) = \sqrt{\frac{1}{n_i - 1} \sum_{j=1}^{n_i} (p_{ij} - \mu(R_i))^2}$$

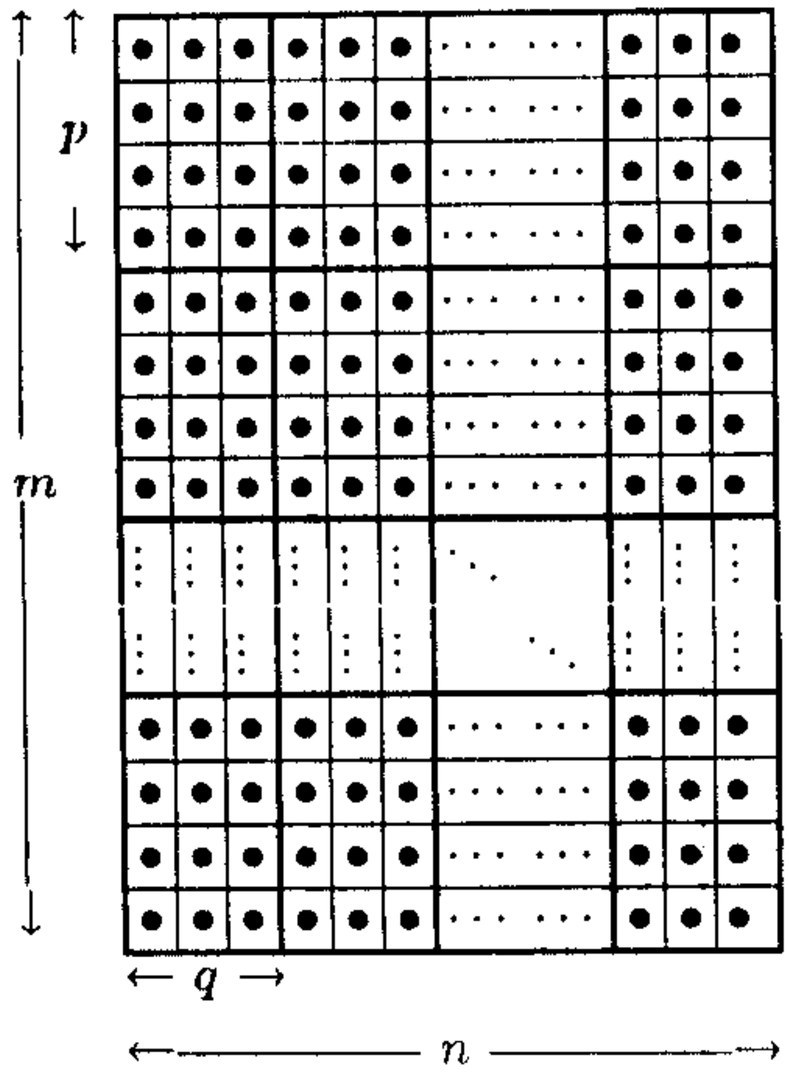
$$\mu(R_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} p_{ij}$$

2.2. Adaptive Region Growing Technique

The segmentation method followed is the *Adaptive Region Growing Technique* used by Kocher & Leonardi. In this method, the entire image is initially split up into a number of small regions. These regions are then merged sequentially, according to some criteria, and the merging continues till no more regions can be merged.

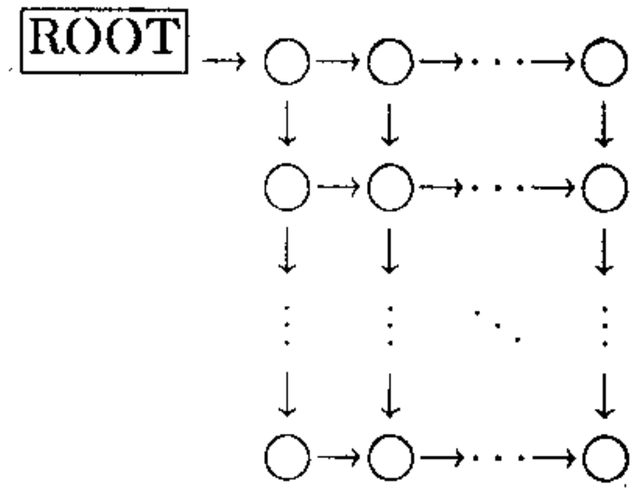
2.2.1. Image Splitting

First of all, the input image is split up into a number of small 4-connected rectangular regions. For an $m \times n$ image, and for initial rectangular region size $p \times q$, we show the split image as follows :



Let the size of the image be $m \times n$, and the size of each initial rectangular region be $p \times q$. The image is partitioned into such rectangular regions as shown above. So, initially each row of the splitted up image contains $\lfloor \frac{n}{q} \rfloor$ segments, and each column contains $\lfloor \frac{m}{p} \rfloor$ segments. Each such segment contains $p \times q$ pixels, except in the last segments of each row or, column. The last segment contains more than $p \times q$ pixels, provided mn is perfectly divisible by pq Otherwise, the last segment contains more than $p \times q$ pixels, so that the $m \times n$ image can be integrally decomposed.

Thus, we have an array ($\lfloor \frac{m}{p} \rfloor \times \lfloor \frac{n}{q} \rfloor$) of regions. Each region is represented by a node, comprising of the boundary of that region and the number of pixels contained in that region. Then, a directed acyclic graph (DAG) is made out of the above nodes in the manner shown. Two nodes are said to be adjacent if there is a link from one to the other in the DAG.



2.2.2. Merging Algorithm

Region growing is carried out by merging the nodes, each of which represent a region in the image, of the above DAG in the following manner :

Step 1: The average value of the pixels of each node i.e.$\mu(R_i)$ are calculated and are assigned to the nodes.

Step 2: The value of dissimilarity between the pixels of every two adjacent nodes, representing the regions R_i and R_j, $\wp(R_i \cup R_j)$ are calculated and assigned to the corresponding links.

Step 3: A breadth-first topological search of the DAG is made to find out the link, containing the minimum value of $\wp$, and the corresponding two adjacent nodes.

Step 4: The two adjacent nodes containing the minimum link are then merged in the following manner :

Let a be the node which has the minimum link as the outlink, and b be the node which has the minimum link as the inlink. Let $a^i = \{a_u^i\}$ be the set of nodes from which there are in-links to a, and $a^o = \{a_v^o\}$ be the set of nodes to which a has got out-links. Likewise, let $b^i = \{b_s^i\}$ be the set of nodes from which there are in-links to b, and $b^o = \{b_t^o\}$ be the set of nodes to which b has got out-links. The links in the neighbourhood of a and b are then modified in the following manner:

Step 4a: The links from a^i to a are kept as they are.

Step 4b: The links from $a^i \cap b^i$ to b are destroyed, and the elements of $a^i \cap b^i$ are deleted from b^i.

Step 4c: Let $b^i_\alpha \in b^i - a^i$.

If b^i_α be at a topological level lower than that of a, then make a link from a to b^i_α, include b^i_α in the set a^o, destroy the link from b^i_α to b, and delete b^i_α from the set b^i.

Otherwise, the link from b^i_α is made to point towards a, b^i_α deleted from the set b^i, and b^i_α included in the set a^i.

Step 4d: $b^o - a^o$ is added to the set a^o, and the links from a to $b^o - a^o$ are established. The elements of the set b^o are deleted, and the links from b to them are destroyed.

a and b are then merged to form a single node, consisting of the initial rectangular regions of both a and b. The link from a to b is deleted, and b is deleted from the set a^o. μ for a is calculated and put into it. $\wp$s are updated for all the links from a^i to a and from a to a^o.

Step 5: The above process of merging is carried on till no further nodes can be merged. This terminal condition can be obtained from the value of θ, which is to be specified by the user.

2.2.3. Contour Tracking

Once the merging is complete, we are left with the DAG,where each node represents a region which satisfies the properties as mentioned in 2.1. Each segmentated region has a list of the initial rectangular regions of size $p \times q$. We extract the contour from this list. An algorithmic description is as follows :

Step 1: Each node is given an unique identity number(ID).

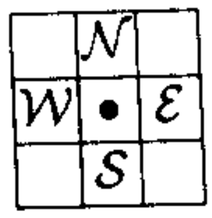
Step 2: An array A, having the same dimensions as the image, is constructed.

Step 3: The positions, in the array A, belonging to the initial rectangular regions, are colored with the ID of the node to which the corresponding initial rectangular region belongs.

Step 4: This array A is scanned, row-wise, from the top left position. During the scan, at each position we check whether the IDs corresponding

to its four 4-connected neighbours of that position are all equal to its ID of that corresponding position. If they are all equal, then the present position is an interior position, and we proceed with our scan. Otherwise, this position is a boundary position, and we see whether this boundary has been already considered. If it has been already considered, then we proceed with our scan. Else, we track the boundary in the following manner :

Step 4.a: The 4-neighbours of a position are named as shown.



We compare the color of a boundary position with those of its neighbours, and choose the next boundary position in the following manner.

Step 4.a.i: If $\mathcal{W}$ matches, but $\mathcal{N}$ doesn't match, we move to the left and select $\mathcal{W}$ as our next boundary position.

Step 4.a.ii: If $\mathcal{S}$ matches, but $\mathcal{W}$ doesn't match, we move down and select $\mathcal{S}$ as our next boundary position.

Step 4.a.iii: If $\mathcal{E}$ matches, but $\mathcal{S}$ doesn't match, we move to the right and select $\mathcal{E}$ as our next boundary position.

Step 4.a.iv: If $\mathcal{N}$ matches, but $\mathcal{E}$ doesn't match, we move up and select $\mathcal{N}$ as our next boundary position.

Step 4.a.v: If all of them matches, we disregard the current position as a boundary position (if we are considering 8-connected contour; otherwise, not), and consider our last move.

- ◇ If our last move was to the left, we move up and select $\mathcal{N}$ as our next boundary position.
- ◇ If our last move was downwards, we move left and select $\mathcal{W}$ as our next boundary position.
- ◇ If our last move was to the right, we move down and select $\mathcal{S}$ as our next boundary position.
- ◇ If our last move was upwards, we move right and select $\mathcal{E}$ as our next boundary position.

Step 4.b: We go on finding the next boundary position and marking them as traversed, till we reach the starting position of the contour.

Once a boundary is tracked, it is added to the list of boundaries for that particular region. Then, scan follows for the other boundaries.

2.3. Result and Discussions

We have applied the adaptive region growing technique for segmentation on three different types of input images. Two of them are of size 64×64, and have 32 graylevels, while the third one is of size 256×256 and have 256 graylevels. The number of regions corresponding to different regions for different threshold values are tabulated as follows.

Input Images	Threshold θ	No. of Regions
Lena	8	1926
	9	1693
	10	1493
Lincoln	12	166
	13	162
	14	153
Plane	15	60
	17	50
	19	44

The images are shown as follows.

Lena
Input Image

Chapter 3

Encoding

After we obtain the segmented image, we start the process of encoding each regio. The contour of each region is encoded by a random-walk encoding method. While, the graylevels of each region is encoded by parametric surfaces.

3.1. Contour Encoding

The random-walk method for encoding a 4-connected contour was proposed by Kocher and Kunt. However, we have extended to the case of a 8-connected contour.

3.1.1. Terminology

Each contour is represented by a starting point and a string of symbols, each symbol giving an information about the next pixel in the contour. The following set of three symbols — R:denoting a right turn, L:denoting a left turn, S:denoting a straight move, is used.

P_0 = Starting point of a contour
M_k = k^{th} symbol of the contour
P_k = Contour pixel having the symbol M_k
$P_k(i)$ = Row-value of pixel P_k
$P_k(j)$ = Column-value of pixel P_k
$\delta_k(i)$ = $P_k(i) - P_{k-1}(i)$
$\delta_k(j)$ = $P_k(j) - P_{k-1}(j)$

3.1.2. The 4-connected method

In the 4-connected method used by Eden and Kocher, M_k can take the values R, L or, S. Accordingly, we obtain the following configurations :-

• For $M_k = R$:

P_{k-1}	P_k	
	P_{k+1}	

$$\delta_{k+1}(i) = \delta_k(j)$$
$$\delta_{k+1}(j) = \delta_k(i)$$

• For $M_k = L$:

	P_{k+1}	
P_{k-1}	P_k	

$$\delta_{k+1}(i) = -\delta_k(j)$$
$$\delta_{k+1}(j) = \delta_k(i)$$

• For $M_k = S$:

P_{k-1}	P_k	P_{k+1}

$$\delta_{k+1}(i) = \delta_k(i)$$
$$\delta_{k+1}(j) = \delta_k(j)$$

The above three configurations can be applied to the following four grid orientations to obtain twelve contour patterns. The four grid orientations are as follows :

Grid 1

$i-1, j-1$	$i-1, j$	$i-1, j+1$
$i, j-1$	i, j	$i, j+1$
$i+1, j-1$	$i+1, j$	$i+1, j+1$

Grid 2

$i-1, j+1$	$i, j+1$	$i+1, j+1$
$i-1, j$	i, j	$i+1, j$
$i-1, j-1$	$i, j-1$	$i+1, j-1$

Grid 3

$i+1, j+1$	$i+1, j$	$i+1, j-1$
$i, j+1$	i, j	$i, j-1$
$i-1, j+1$	$i-1, j$	$i-1, j-1$

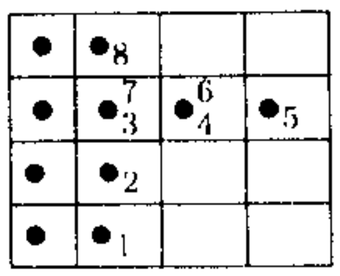
Grid 4

$i+1, j-1$	$i, j-1$	$i-1, j-1$
$i+1, j$	i, j	$i-1, j$
$i+1, j+1$	$i, j+1$	$i-1, j+1$

If we take any three consecutive pixels from a contour, it will obviously belong to one of the twelve contour patterns. There is one assumpsion that, the contour should belong to a region which is formed by merging sub-regions of minimum dimension 2×2. On the contrary, if we had allowed

initial sub-regions of dimension less than 2×2, then we could easily have obtained a region as shown.

The contour has to be traversed as numbered. It can be easily seen that no move exists for the point having mark five. Moreover, there are two points which require two moves. So, such contours are not acceptable.



3.1.3. The 8-connected method

In the 8-connected method, our convention for the sence of the move is anti-clockwise for the outer (bounding) contour and clockwise for the inner (hole) contour. Based on this convention and assumption for the 4-connected case, we generate the following four (three for P_{k-1} and P_k being 4-connected and only one for them being just 8-connected) configurations :-

- For $M_k = L$:

	P_{k+1}	
P_{k-1}	P_k	

$$\delta_{k+1}(i) = -\delta_k(j)$$
$$\delta_{k+1}(j) = \delta_k(i)$$

- For $M_k = R$:

P_{k-1}	P_k	
		P_{k+1}

$$\delta_{k+1}(i) = \delta_k(j) + \delta_k(i)$$
$$\delta_{k+1}(j) = \delta_k(j) - \delta_k(i)$$

- For $M_k = S$:

Parallel

P_{k-1}	P_k	P_{k+1}

Diagonal

	P_k	P_{k+1}
P_{k-1}		

$$\delta_{k+1}(i) = \delta_k(i)\phi(\delta_k(i) + \delta_k(j))$$
$$\delta_{k+1}(j) = \delta_k(j)\phi(\delta_k(j) - \delta_k(i))$$

where,

$$\phi(x) = \begin{cases} 1, & \text{if } x \neq 0; \\ 0, & \text{otherwise.} \end{cases}$$

The above four configurations can be applied to the following four grid orientations to obtain sixteen contour patterns. The four grid orientations are as follows :

Grid 1

$i-1, j-1$	$i-1, j$	$i-1, j+1$
$i, j-1$	i, j	$i, j+1$
$i+1, j-1$	$i+1, j$	$i+1, j+1$

Grid 2

$i-1, j+1$	$i, j+1$	$i+1, j+1$
$i-1, j$	i, j	$i+1, j$
$i-1, j-1$	$i, j-1$	$i+1, j-1$

Grid 3

$i+1, j+1$	$i+1, j$	$i+1, j-1$
$i, j+1$	i, j	$i, j-1$
$i-1, j+1$	$i-1, j$	$i-1, j-1$

Grid 4

$i+1, j-1$	$i, j-1$	$i-1, j-1$
$i+1, j$	i, j	$i-1, j$
$i+1, j+1$	$i, j+1$	$i-1, j+1$

If we take any three consecutive pixels from a contour, it will obviously belong to one of the sixteen contour patterns. We use the assumption made in the 4-connected case to get rid of the extraneous cases which arise due to 8-connectivity.

0.0.1 Case Elimination

When P_{k-1} and P_k are 4-connected :

Case I :

P_{k+1}		
P_{k-1}	P_k	

Reason:Due to our initial assumption we cannot have such a region. Because, the common 8-connected component of P_{k-1}, P_k and P_{k+1} should at least be inside the same region.

Case II :

		P_{k+1}
P_{k-1}	P_k	

Reason:By the segmentation procedure that we have followed, the initial regions are rectangular in shape and the boundary between any two such regions are of the same size & totally aligned. Since, our initial assumption doesn't allow us to have initial rectangular regions of dimension less than 2×2; either the common 4-connected points of P_k and P_{k+1} must also be inside the same region or, P_{k+1} cannot be a contour point.

Case III :

P_{k-1}	P_k	
	P_{k+1}	

Reason:We can replace the move M_k by making $M_{k-1} = R$ and taking P_{k-2}, P_{k-1} and P_{k+1} as three consecutive pixels on the contour (i.e. diregarding P_k as a contour point).

Case IV :

P_{k-1}	P_k	
P_{k+1}		

Reason:Same as Case I.

When P_{k-1} and P_k are only 8-connected :

Case I :

P_{k+1}	P_k	
P_{k-1}		

Reason:Same as Case I of the 4-connected case.

Case II :

P_{k+1}		
	P_k	
P_{k-1}		

Reason:By the segmentation procedure that we have followed, the initial regions are rectangular in shape and the boundary between any two such regions are of the same size & totally aligned. Since, our initial assumption doesn't allow us to have initial rectangular regions of dimension less than 2×2; we cannot have a region where a single pixel protrudes out of a straight contour.

Case III :

	P_{k+1}	
	P_k	
P_{k-1}		

Reason:Same as Case II of the 4-connected case.

Case IV :

		P_{k+1}
	P_k	
P_{k-1}		

Reason:Same as Case II of the 4-connected case.

Case V :

	P_k	
P_{k-1}		P_{k+1}

Reason:Same as Case II of the 8-connected case.

Case VI :

	P_k	
P_{k-1}	P_{k+1}	

Reason:Same as Case I of the 4-connected case.

3.1.4. Expression for Computing the no. of Contour Bits

Let –

L_n = Total number of moves having label L

R_n = Total number of moves having label R
S_n = Total number of moves having label S

Then the total number of contour bits is given by the equation :

$$2\ (L_n + R_n + S_n)\ -\ max(L_n,\ R_n,\ S_n)$$

3.2. Graylevel Encoding

In this chapter, we consider the problem of approximating a 2-dimensional signal using analytic functions. After we obtain the regions from the segmentation procedure, the signal within each region is approximated by a 2-dimensional polynomial function. The constraint of the approximation being that, the total root mean square approximation error calculated over the entire image must be minimized. Two cases are considered here — one is the approximation by a quadratic function, proposed by Kocher and Leonardi, while the other being a cubic spline approximation.

3.2.1. Terminology

(x_i, y_i) = Co-ordinates of the i^{th} pixel in region R, $i = 1, 2, \ldots, n$
$g(x_i, y_i)$ = Original gray value of the i^{th} pixel in region R
$\hat{g}(x_i, y_i)$ = Approximated gray value of the i^{th} pixel in region R
$\mathbf{g}$ = $(g(x_1, y_1), g(x_2, y_2), \ldots, g(x_n, y_n))^T$
$\hat{\mathbf{g}}$ = $(\hat{g}(x_1, y_1), \hat{g}(x_2, y_2), \ldots, \hat{g}(x_n, y_n))^T$
$\mathbf{e}$ = The error vector, $(e_1, e_2, \ldots, e_n)^T$
Γ_0 = Total number of bits in the original image
Γ = Total number of bits in the compressed image
γ = Γ_0 / Γ

3.2.2. Quadratic Case

The polynomial approximation function, used by Kocher & Leonardi to parametrize the graylevels of a particular region of a given image, is a two-dimensional quadratic function. The analytic function chosen, is given as:

$$\hat{g}(x, y) = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 y^2 + b_5 xy$$

For all the points in the region R, the above equation can be written in the matrix form as :

$$\hat{\mathbf{g}} = \mathbf{x}\ \mathbf{b}$$

where,

$$\mathbf{b} = (b_0, b_1, \ldots, b_5)^T$$

$$\mathbf{x} = \begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2^2 & y_2^2 & x_2 y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & x_n^2 & y_n^2 & x_n y_n \end{pmatrix}$$

The mathematical relations between the original gray values, the polynomial co-efficients and the approximation error can be written as:

$$\mathbf{g} = \mathbf{x}\mathbf{b} + \mathbf{e}$$

By least-square minimization of the above error $\mathbf{e}$, we get :

$$\mathbf{b} = (\mathbf{x}^T\mathbf{x})^{-1}(\mathbf{x}^T\mathbf{g})$$

3.2.3. Cubic Spline Case

The polynomial approximation function, used by Kocher & Leonardi to parametrize the graylevels of a particular region of a given image, has been extended to the case of a cubic spline function. However, instead of using control points, we have used a parametric form, as this will enable us to reduce the compression cost. Moreover, for 4×4 array of control points, the Cubic B-spline reduces to cubic spline. The analytic function chosen, is given as:

$$\begin{aligned} \hat{g}(x,y) &= b_0 + b_1 y + b_2 y^2 + b_3 y^3 \\ &+ b_4 x + b_5 xy + b_6 xy^2 + b_7 xy^3 \\ &+ b_8 x^2 + b_9 x^2 y + b_{10} x^2 y^2 + b_{11} x^2 y^3 \\ &+ b_{12} x^3 + b_{13} x^3 y + b_{14} x^3 y^2 + b_{15} x^3 y^3 \end{aligned}$$

For all the points in the region R, the above equation can be written in the matrix form as :

$$\hat{\mathbf{g}} = \mathbf{x}\ \mathbf{b}$$

$$\text{where, } \mathbf{b} = (b_0, b_1, \ldots, b_{15})^T$$

$$\mathbf{x} = \begin{pmatrix} 1 & y_1 & y_1^2 & y_1^3 & x_1 & x_1y_1 & x_1y_1^2 & x_1y_1^3 & \cdots & x_1^3y_1^3 \\ 1 & y_2 & y_2^2 & y_2^3 & x_2 & x_2y_2 & x_2y_2^2 & x_2y_2^3 & \cdots & x_2^3y_2^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & y_n & y_n^2 & y_n^3 & x_n & x_ny_n & x_ny_n^2 & x_ny_n^3 & \cdots & x_n^3y_n^3 \end{pmatrix}$$

The mathematical relations between the original gray values, the polynomial co-efficients and the approximation error can be written as :

$$\mathbf{g} = \mathbf{x}\mathbf{b} + \mathbf{e}$$

By least-square minimization of the above error $\mathbf{e}$, we get :

$$\mathbf{b} = (\mathbf{x}^T\mathbf{x})^{-1}(\mathbf{x}^T\mathbf{g})$$

3.3. Result and Discussions

Using the above mentioned two methods for contour encoding on the same input images as those used for segmentation, we get the following results :

Input Image	Threshold θ	4-connected case		8-connected case	
		Bits/Pixel	No. of Bits	Bits/Pixel	No. of Bits
Lena	8	1.28844	35557	1.30038	34460
	9	1.27788	33281	1.28996	32196
	10	1.26931	31300	1.28189	30200
Lincoln	12	1.29245	2740	1.30112	2679
	13	1.29278	2720	1.30211	2655
	14	1.29108	2648	1.30121	2579
Plane	15	1.23445	1548	1.24561	1491
	17	1.22074	1460	1.23178	1403
	19	1.21552	1410	1.22704	1351

The results of fitting the quadratic patches on the images, mentioned in the segmentation chapter, are produced as follows. Although, we have studied the possibility of applying cubic splines to the segmented images, we are producing few results as we have failed to obtain consistent satisfactory result. For the quadratic case, 12 bits are sufficient for coding a single

parameter, resulting in 72 bits for graylevel encoding of each region.So, in this case, the expression for the compression ratio γ comes as :

$$\gamma = 72 \times \text{No. of regions} + \text{No. of contour points}$$

For the Lena image this value is from 3 to 3.8.

Lena
Quadratic Case
$\theta = 9$

Lena
Quadratic Case
$\theta = 10$

Conclusions

The software for the entire project has been developed in the X Window System environment. It is an interactive software with its own graphical user interface. Moreover, for the display of the spline surfaces, a 2-dimensional surface renderer has also been developed. The primary motive behind the development of such a software is to have an experience in the practical aspects of image processing and computer graphics.

As far as the theoretical aspects are concerned, the implementation of the segmentation algorithm using a DAG has been developed by us. This algorithm can be extended to a parallel segmentation algorithm if a suitable algorithm for generating disjoint mergable neighborhoods can be found.

The application of splines to the regions has not been so much of a success due to the lack of a suitable error minimization procedure. It seems that, at least some sort of adaptive gradient descent algorithm is required to get satisfactory result in this case.

References

[1] Rosenfeld, A. and A. C. Kak, *Digital Picture Processing, Volume 2*, Academic Press, Inc.

[2] Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company.

[3] Foley, D. J., A. van Dam, S. K. Feiner and J. K. Hughes, *Computer Graphics, Principles and Practice*, Addison-Wesley Publishing Company.

[4] Kocher, M. and R. Leonardi, *Adaptive Region Growing Using Polynomial Functions For Image Approximation*, Signal Processing(North-Holland) 11 (July 1986) 47-60

[5] Barsky, Brian A., *A Description and Evaluation of Various 3-D Models*, IEEE CG&A (January 1984)