Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

Title of Thesis: A non-invasive technique for burn area measurement
Jong-Daw Yu, Master of Science
Thesis directed by: Stanley Dunn, Ph.D
Rutgers University
Biomedical Engineering Department
Peter Engler, Ph.D.
New Jersey Institute of Technology
Electrical Engineering Department

Abstract

The need for a reliable and accurate method for assessing the surface area of burn wounds currently exists in the branch of medicine involved with burn care and treatment. The percentage of the surface area is of critical importance in evaluating fluid replacement amounts and nutritional support during the 24 hours of postburn therapy. A noninvasive technique has been developed which facilitates the measurement of burn area. The method we shall describe is an inexpensive technique to measure the burn areas accurately.

Our imaging system is based on a technique known as structured light. Most structured light computer imaging systems, including ours, use triangulation to determine the location of points in three dimensions as the intersection of two lines: a ray of light originating from the structured light projector and the line of sight determined by the location of the image point in the camera plane. The geometry used to determine 3D location by triangulation is identical to the geometry of other stereo-based vision system, including the human vision system.

Our system projects a square grid pattern from 35mm slide onto the patient. The grid on the slide is composed of uniformly spaced orthogonal stripes which may be indexed by row and column. Each slide also has square markers placed in between the lines of the grid in both the horizontal and vertical directions in the center of the slide.

Our system locates intersections of the projected grid stripes in the camera image and determines the 3D location of the corresponding points on the body by triangulation.

Four steps are necessary in order to reconstruct 3D locations of points on the surface of the skin: camera and projector calibration; image processing to locate the grid intersections in the camera image; grid labeling to establish the correspondence between projected and imaged intersections; and triangulation to determine three-dimensional position. Three steps are required to segment burned portion in image: edge detection to get the strongest edges of the region; edge following to form a closed boundary; and region filling to identify the burn region. After combining the reconstructed 3D locations and segmented image, numerical analysis and geometric modeling techniques are used to calculate the burn area. We use cubic spline interpolation, bicubic surface patches and Gaussian quadrature double integration to calculate the burn wound area.

The accuracy of this technique is demonstrated The benefits and advantages of this technique are, first, that we don't have to make any assumptions about the shape of the human body and second, there is no need for either the Rule-of-Nines, or the weight and height of the patient. This technique can be used for human body shape, regardless of weight proportion, size, sex or skin pigmentation.

The low cost, intuitive method, and demonstrated efficiency of this computer imaging technique makes it a desirable alternative to current methods and provides the burn care specialist with a sterile, safe, and effective diagnostic tool in assessing and investigating burn areas.

A Non-Invasive Technique for Burn Area Measurement

by
Jong-Daw Yu

Thesis submitted to the Faculty of the Graduate School
of the New Jersey Institute of Technology in partial
fulfillment of the requirements for the degree of
Master of Science
1989

APPROVAL SHEET

Title of Thesis : A Non-Invasive Technique of Burn Area Measurement

Name of Candidate : Jong-Daw Yu
Master of Science, 1989

Thesis and Abstract Approved :

Stanley Dunn, Ph.D — Date
Rutgers University
Biomedical Engineering Department

Peter Engler, Ph.D — Date
New Jersey Institute of Technology
Electrical Engineering Department

David Kristol, Ph.D — Date
New Jersey Institute of Technology
Biomedical Engineering Department

VITA

Name : Jong-Daw Yu

Permanent address :

Degree and date to be conferred : M.S., 1989

Date of birth :

Place of birth :

Collegiate institutions attended	Dates	Degree	Date of Degree
Chung-Yuang Christ. Univ.	9/80-5/84	B.S.	May 1984
New Jersey Institute of Technology	1/87-5/89	M.S.	May 1989

Major : Biomedical Engineering

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are many medical researchers and physicians who have, over the past century, studied the problem of estimating the surface area of the human body.

For the diagnosis and treatment of burn patients, knowing the surface area is important for establishing initial fluid replacements, nutritional requirements, and an initial prognosis for the patient [7]. Once the surface area of the burn wound can be accurately determined, the wound healing progression may be monitored in response to several topical healing ointments during the course of burn therapy which allows for the evaluation of pharmaceutical agents and their efficiency in reducing the wound area.

In 1915, Dubois and Dubois [14] modeled anatomic sections of the body into a series of cylinders each having an area equal to the circumference times the height. The final formula which they derived is :

$$Area = Weight^{0.425} * Height^{0.725} * 71.84 \tag{1.1}$$

The formula takes into account the large interpersonal difference in body fluid and stature since both weight and height are accounted for in the formula, and has since been accepted as a standard in determining the total body surface area.

In 1944 Lund and Browder [35] developed a burn chart and diagrams for both adults and children. The most widely used method for determining the surface area was developed in 1947, by Pulaski and Tennison called the Rule-of-Nines in which the various body segments are represented as percentages that are multiples of nine. The Rule-of-Nines was found to be in close agreement when compared with the Lund and Browder chart [33].

The disadvantages of both the Lund and Browder chart and the Rule-of-Nines are that they all require visual estimation on the part of the physician. Sizeable errors occur in estimating the percent total body surface area. The degree of the error varies widely among several observers estimating the area of the same burn patient when using the Rule-of-Nines method; thus attempts have been made to directly measure the burn areas and then determine the percentage total body surface area (%TBSA). During the last decade, many researchers have used the planimeter to estimate surface area. The recent research using such techniques include Nichart, Bryant, Edlich in 1985 [43], Fuller, Mansour, Engler and Shuster in 1985 [23], and Scott-Conner and Conner in 1988 [57]. But they all required the user to draw the contour of a burn portion on a diagram of the computer screen or a plotter.

In 1987, Miss Rath presented her master thesis, the Noncontacting Surface Area Assessment of Burn Wounds Using Computer Image Methods at NJIT [54]. Her technique involved projecting an orthogonal grid onto the surface to be measured, and capturing the image on video tape for processing by a digitizer. The drawbacks in her paper are: first, she had to measure the distance between projector and object and the distance between camera and object manually, in other words she had to compute the area of a pixel in linear units squared in the 3D Cartesian coordinate system; second, she as-

sumed the object or human body consisted of cylinders; third, she didn't mention how to get the contour of the burn area; fourth, she reduced the 3D problem to 2D which means she didn't consider the curvature or the normal vector of the surface.

Since we live in a three-dimensional world, one of the intuitive ways to obtain the three-dimensional properties of an object, in our case is the human body, is using sterometric techniques. The observable domain of an object is delineated by the surface encompassing the object. The technology to obtain three-dimensional spatial coordinates of a sample of points on the surface of an object is called surface digitization. The process of doing surface digitization without human interaction will be auto-surface digitization [50].

Basically, there are two types of sterometric techniques. One is the active binocular system, another is the passive stereo system. Applications of active sensing in the binocular system involve triangulation. The construction of an active binocular system is identical to that of a passive stereo system, expcet that one of the cameras is replaced by a controlled light source. The light source selectively illuminates the scene, and the illuminated portion can be readily detected in the image with triangulation to recover the spatial position. The light patterns can be points or lines [29], [8], or complicated pattern such as orthogonal grids [48],[47],[46],[52], [22],[20],[21],[39], [60], [25], [50], [51], [58], [59], or sets of parallel stripes [11],[65],[64], [66]. The light sources are laser, X-ray, or ordinary structured white light. The system described in this paper falls into the category of active binocular computer-image system. The advantage of our image system is that there is no biological risk of using lasers and/or X-rays, instead we are using a 35mm projector as a light source.

Chapter 2

Automatic Calibration for Camera and Projector

One generally does not know anything about the imaging geometry when looking at a picture. For example, the focal length and location of the camera are both unknown. However, people can produce an excellent qualitative description of scene content apparently without being able to deduce these imaging parameters. How the human visual system can accomplish this task is still a mystery. In order to recover depth, shape and other scene properties using known computational techniques, we must first construct a quantitative model describing the geometric relationship between the scene, the camera and the image.

One way to state this problem is: given a set of correspondences between a known three-dimension configuration of points and their locations in a single two-dimension image, determine the parameters of the camera that produced the image. Fischler and Bolles [19] show that if at least six correspondences between world points (with known 3D locations) and image points can be established , then it is possible to determine the twelve coefficients of the 3×4 collineation matrix that specifies the mapping (in homogeneous coordinates) from 3-space to 2-space.

Camera calibration is the determination of the correspondence between points in the camera image and points in physical space. Only when the camera is properly calibrated can its 2D raster coordinates be translated into real-world locations for the objects in its field of view. The same approach is used for projector calibration.

2.1 Assumptions

Before discussing the meaning of the camera calibration matrix and the projector calibration matrix, we have to make some assumptions about the mathematical model for camera and projector.

In the following derivation we first assume

A that the lens of the camera behaves as an ideal, thin lens;

B that the pixel matrix is aligned with the external coordinate system;

C that the reflecting surface is perfectly diffuse;

D that the pixel raster is ideal.

The ideal thin lens assumption states that the lens can be replaced by a pinhole [9], [10], [30]. The location of this equivalent pinhole is called the optical center of the lens.

The alignment assumption states that the pixel rows must be parallel to the plane of light, and that the columns hence are perpendicular. The diffuse-surface assumption requires that the intensity of the reflected light be independent of reflection angle, and the assumption of an ideal pixel raster requires that the camera be noise-free and of infinite resolution.

2.2 Camera Calibration

The camera calibration matrix represents the functional relationship between points in the 3D object-centered coordinate system and the pixel location of the projected point in the camera plane and it is a 3 $\times$ 4 matrix in homogeneous coordinates. Equation 2.1 represents the camera calibration matrix.

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 & t_7 \\ t_8 & t_9 & t_{10} & t_{11} \end{bmatrix} \tag{2.1}$$

The values of the camera calibration coefficients t_i depend on the orientation, translation, focal distance, and scaling of the camera relative to the three-dimensional scene.

An image plane is placed between the lens and the object so that the image plane is perpendicular to the optical axis of the lens and at a distance of one focal length from the lens center. The projection of an object point onto the image plane is determined as the intersection of the image with the line connecting the object point and the lens center (optical center of the lens).

Without loss of generality, we may assume that the z-axis in the object coordinate system is perpendicular to the image plane and the calibration box.

If f is the focal length of the camera and (x, y, z) the object point coordinates, then by using similar triangles we can express the image plane coordinates (u, v) as :

$$u = \frac{fx}{f + z} \tag{2.2}$$

$$v = \frac{fy}{f + z} \tag{2.3}$$

Equations 2.2 and 2.3 express u and v as nonlinear functions of x, y and z. Since

using these equations to compute the camera and projector calibration parameters involves solving simultaneous nonlinear equations which is very difficult, we shall use homogeneous coordinates as the mathematical representation with which one can express the perspective transformation as a linear transformation.

In homogeneous coordinates, a three-dimensional Cartesian vector (x, y, z) is represented by a four component vector (wx, wy, wz, w) where w is an arbitrary constant. The Cartesian coordinates of a point are recovered from the homogeneous coordinates by dividing each of the first three components by the last component.

The following matrix equation in homogeneous coordinates represents the camera image formation process which maps the point (x_i, y_i, z_i) in the objected centered coordinates onto the point (u_i, v_i) in the camera image plane

$$\begin{bmatrix} wu_i \\ wv_i \\ w \end{bmatrix} = \begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 & t_7 \\ t_8 & t_9 & t_{10} & t_{11} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{2.4}$$

These linear equations in the homogeneous coordinates of equation 2.4 are equivalent to the following system of two equations involving the Cartesian coordinates

$$\begin{bmatrix} t_0 - u_i t_8 & t_1 - u_i t_9 & t_2 - u_i t_{10} & t_3 - u_i t_{11} \\ t_4 - v_i t_8 & t_5 - v_i t_9 & t_6 - v_i t_{10} & t_7 - v_i t_{11} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.5}$$

The purpose of the camera calibration procedure is to solve for the unknown elements $t_0, \ldots, t_{11}$ of the camera calibration matrix.

If we know the coordinates (x_i, y_i, z_i) for 6 or more points and the coordinates of their projections (u_i, v_i) in the camera image plane then the unknown camera calibration matrix elements may be found by setting $t_{11} = 1$ and determining the reminding 11 elements of the calibration matrix as the least square solution of the overdetermined

linear system of equation 2.6

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 \\ x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 & -u_3x_3 & -u_3y_3 & -u_3z_3 \\ \vdots & & & & \vdots & & & & \vdots & & \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_nx_n & -u_ny_n & -u_nz_n \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2 & -v_2y_2 & -v_2z_2 \\ \vdots & & & & \vdots & & & & \vdots & & \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -v_nx_n & -v_ny_n & -v_nz_n \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \\ t_{11} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.6}$$

2.3 Projector Calibration

The elements of the projector calibration matrix depend on the position and orientation of the projector relative to the object-centered coordinates, the focal length of the projector, and a scale factor which takes into account the spacing of the grid stripes which are the projected pattern, which will be shown later.

Since we have already determined the camera calibration matrix, the object-centered coordinates of points in space (x_i, y_i, z_i) and the corresponding row and column of the projected grids (p_i, q_i), solving for the unknown elements of the projector calibration matrix is identical to solving for the unknown elements of the camera matrix

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -p_1x_1 & -p_1y_1 & -p_1z_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -p_2x_2 & -p_2y_2 & -p_2z_2 \\ x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 & -p_3x_3 & -p_3y_3 & -p_3z_3 \\ \vdots & & & & \vdots & & & & \vdots & & \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -q_1x_1 & -q_1y_1 & -q_1z_1 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -q_2x_2 & -q_2y_2 & -q_2z_2 \\ \vdots & & & & \vdots & & & & \vdots & & \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -q_nx_n & -q_ny_n & -q_nz_n \end{bmatrix} \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \\ q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \quad (2.7)$$

where

$$L = \begin{bmatrix} l_0 & l_1 & l_2 & l_3 \\ l_4 & l_5 & l_6 & l_7 \\ l_8 & l_9 & l_{10} & l_{11} \end{bmatrix}$$

is called projector calibration matrix.

By the same token, if we know the coordinates of the object points (x_i, y_i, z_i) and the coordinates of their corresponding projector points (p_i, q_i), the unknown projector calibration matrix elements may be found by setting $l_{11} = 1$ and determining the 11 elements of the projector calibration matrix as the least square solution of equation 2.7

2.4 Automatic Calibration Procedure

The automatic calibration procedure is one of the unique features in our system. Whenever we are dealing with stereophotogrammetry techniques, it is necessary for the camera/lighting configuration (in our case is camera and projector) to be calibrated; that is, for the precise relationship to be established between points on the image and their corresponding coordinates in the external world. The calibration must, in principle, be performed each time the camera and projector system are demounted or modified.

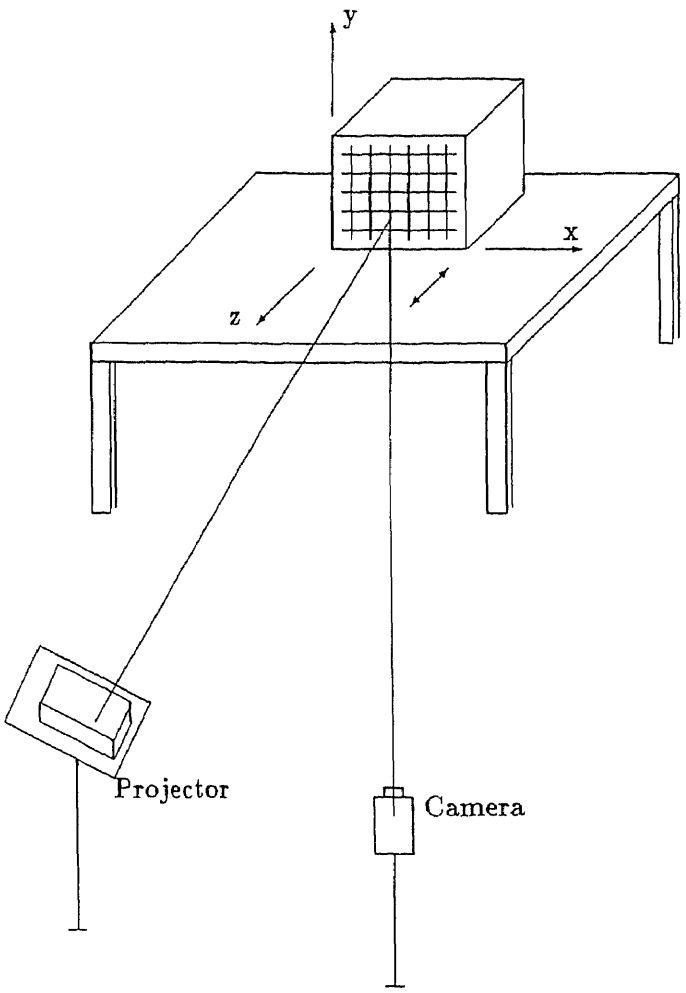

Figure 2.1: The calibration system setup

The purpose of automatic camera and projector calibration is to solve for the calibration matrices without human intervention such as giving initial values for certain parameters by guessing or choosing certain system parameters manually. Our method relies on grid labeling using the marked grid to obtain a set of 3D positions and their corresponding image plane coordinates for both camera and projector calibration. The setup of the system is shown in Figure 2.1.

The camera is calibrated first. A picture of the marked grid is pasted on the side

of a box which is called the "calibration box" and placed at two known object centered positions in the camera's field of view. Each image is processed by the sequence of steps described in Chapter 3 to determine the 2D locations of the intersections in the image plane. The 3D locations are determined from the grid labels, known scale of the pattern and known position of the box in the object centered coordinate system. The camera parameters are computed by equation 2.6.

In order to simplify the calculation, the stripes on the calibration box have one inch distance between adjacent stripes horizontally and vertically. See Figure 2.2.

The projector calibration is determined next. The picture of the marked grid on the box is replaced by a blank piece of paper, and the box is repositioned at the first position. The slide with the pattern shown in Figure 2.3 is placed in the projector and illuminates the box.

The camera image is saved, the box is repositioned at the second calibration position, and the second image is taken. Each image is processed by the procedure in Chapter 3 to determine the 2D location of the intersections and their grid labels. From the 2D location and the camera calibration matrix, each intersection is back projected to its location on the box in the object centered coordinate system. This is repeated for each intersection on both projector images . The 3D position for each detected intersection point on each projector image is calculated using

$$\begin{aligned} x_i &= (c_0 b_1 - c_1 b_0)/f \\ y_i &= (c_1 a_0 - c_0 a_1)/f \\ z_i &: \textit{the translation of the calibration box } (0, 0, z_t) \end{aligned} \tag{2.8}$$

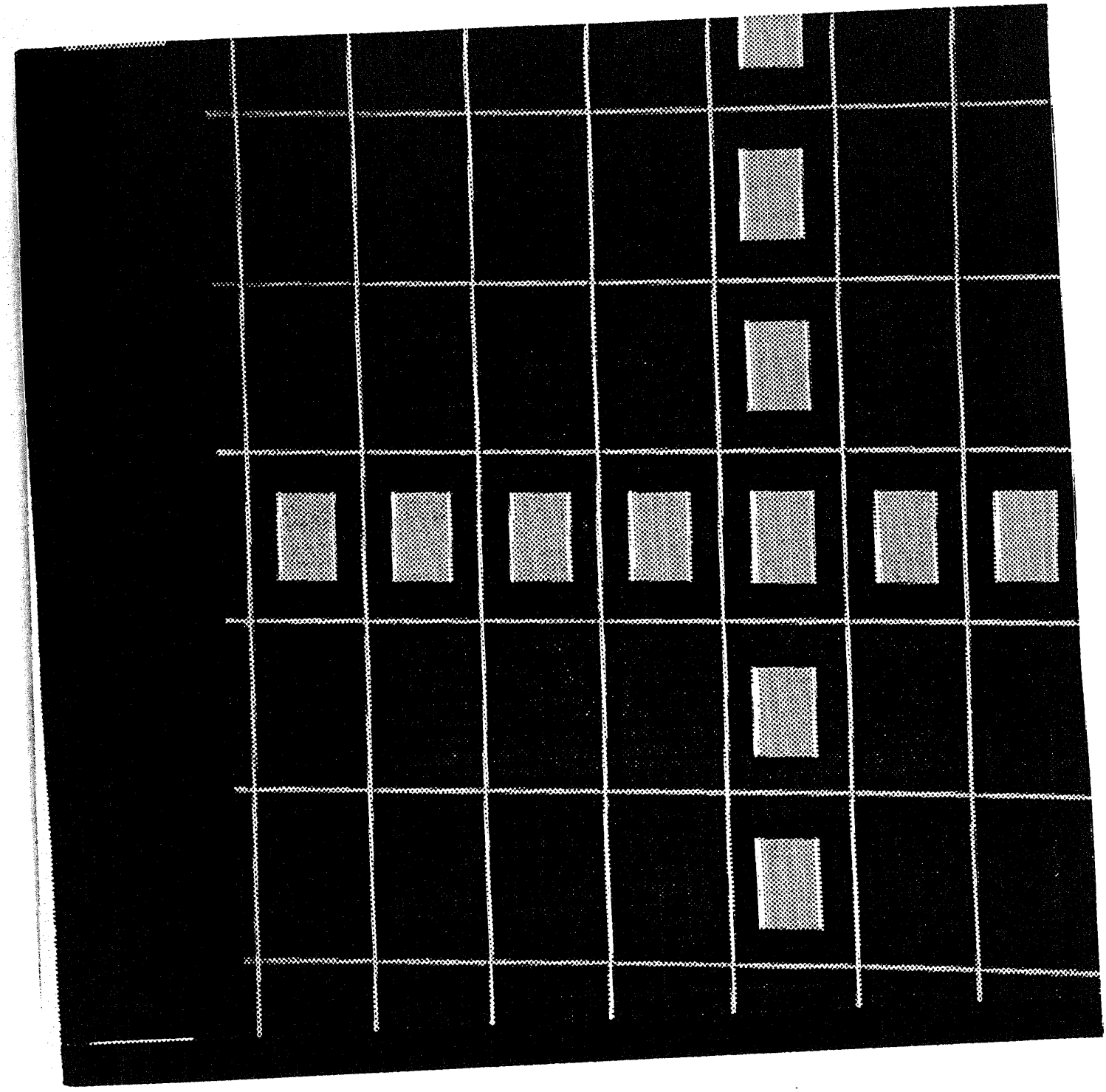

Figure 2.2: The pattern of camera calibration box

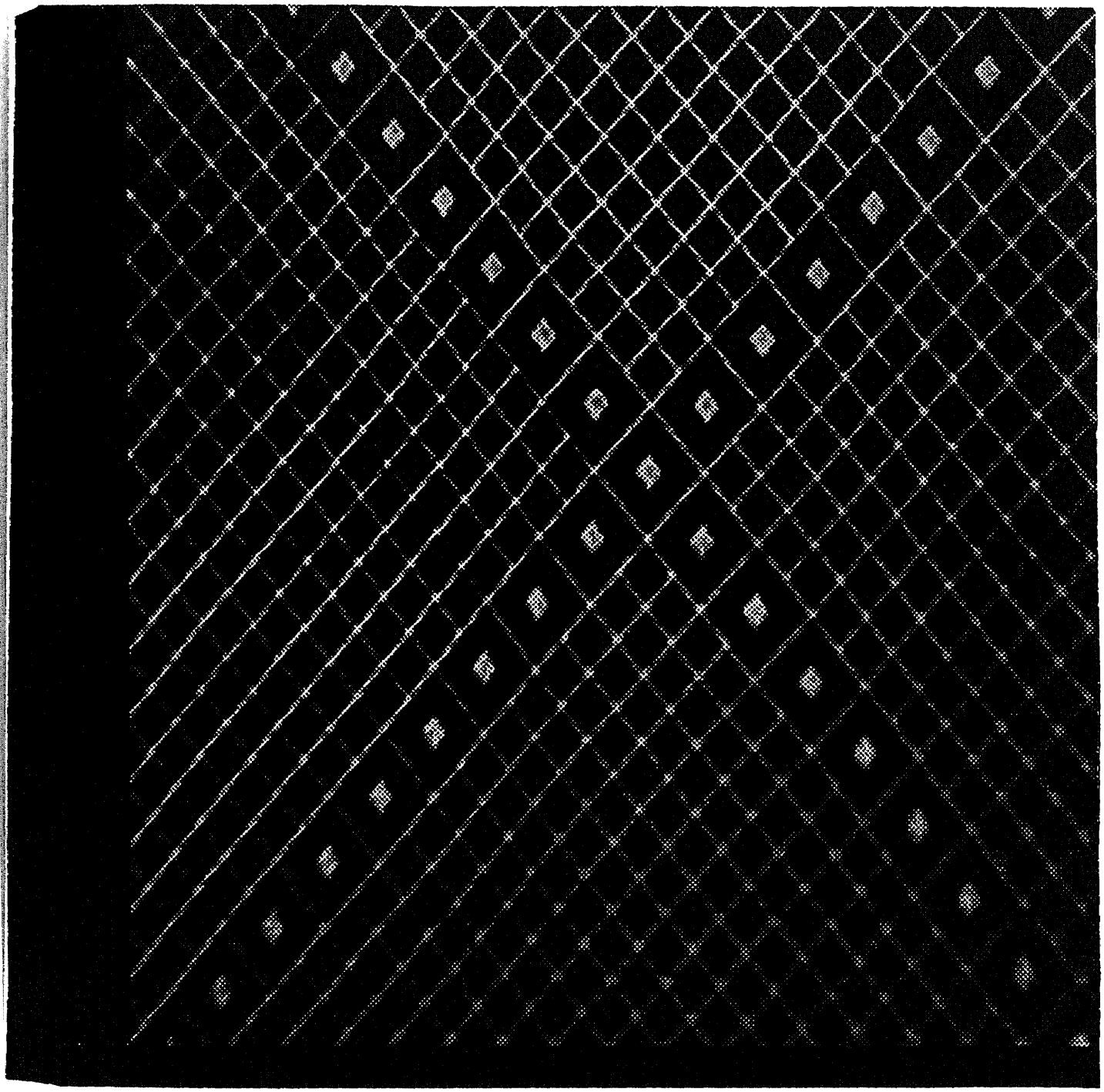

Figure 2.3: The pattern of projector slide

where

$$
\begin{aligned}
c_0 &= u_i + t_0 \times u_i \times z_t - t_2 \times z_t - t_3 \\
c_1 &= v_i + t_{10} \times v_i \times z_t - t_6 \times z_t - t_7 \\
b_0 &= t_1 - t_9 \times u_i \\
b_1 &= t_i - t_9 \times v_i \\
a_0 &= t_0 - t_8 \times u_i \\
a_1 &= t_4 - t_8 \times v_i \\
f &= a_0 \times b_1 - a_1 \times b_0
\end{aligned}
$$

From the grid labels and 3D locations, the projector calibration matrix is computed from equation 2.7.

Chapter 3

Computing Three Dimensional Coordinates

In order to calculate the surface area of a burn wound patient, we need to know how to get the 3D position by using our structured light system. In this chapter, we present algorithms and implementations due largely to Richard Keizer, a Ph.D candidate at Rutgers University, and reported in [16],[17],[31].

Because we use the white light as the structured light in our system, we need additional image processing to accurately locate the grid intersections in the camera image due to the poorer contrast of the white light stripes as compared to laser beams, and due to the fact that the projected white light stripes can not be simultaneously in sharp focus over the entire body. The following steps are used to find the 3D positions with structured light and can be utilized to accurately locate the structured white light grid intersections.

1. Segmentation.

2. Line Thinning.

3. Intersection Detection.

4. Grid Labeling.

5. Triangulation.

3.1 Segmentation

The first step is to extract the structured light stripes in the camera image by segmentation. We have used a two step segmentation procedure consisting of reflectance compensation followed by bimean thresholding.

3.1.1 Reflectance Compensation

This procedure removed the brightness variation due to the variations in the reflectivity of the human skin.

Theoretically, the image intensity due to reflected light from a surface patch illuminated by a single light source is given by

$$I(x,y) = L(x,y)R(x,y) \tag{3.1}$$

where $I(x,y)$ is the image intensity at pixel location (x,y), $L(x,y)$ is the intensity of the incident light at the corresponding point on the objects surface, and $R(x,y)$ is the reflectivity of the surface [28].

The reflectance compensation algorithm begins with estimating the term $R(x,y)$ in equation 3.1 and then dividing the image intensity by this estimate to obtain an estimate of the incident illumination $L(x,y)$. Assuming that $R(x,y)$ varies slowly compared to the grid stripes spacing, we can estimate $R(x,y)$ by the median of the intensities of the image pixels inside a neighborhood centered at (x,y). We recover the illumination by dividing the intensity by the estimate of the surface reflectivity. In practice dividing

$I(x, y)$ by the median is unstable in the background since the local median levels are small. In our system, we add the global median to the local (neighborhood) median before dividing. Thus, this can avoid overcompensating in the background.

3.1.2 Bimean Clustering

This procedure is used to extract the grid stripes from the compensated image.

Mathematically, this procedure computes the values of u and v such that

$$f = \sum_{i=1}^{n} min((x_i - u)^2, (x_i - v)^2) \tag{3.2}$$

and the value of k such that

$$f = \sum_{i=1}^{k} (x_i - u)^2 + \sum_{i=k+1}^{n} (x_i - v)^2 \tag{3.3}$$

is minimized where $x_1 \leq x_2 \leq \ldots \leq x_n$. Practically, the index k is the gray level above which the gray levels belong to the object, and the cluster of gray levels below the index k belongs to the background. In other words, Bimean clustering classifies image pixels according to their gray level as belonging to one of two classes: object (the stripes) and background (the body). The value of the threshold separating object from background is determined from the constraint that the sum of the intraclass distances to the means is minimized for both classes. This allows us to reliably extract small objects from large image.

3.2 Classical Thinning Algorithm

After the segmentation procedures, we make the grid stripes of the segmented image exactly one pixel wide by shrinking each stripe to its "*skeleton*". The algorithm we use is called the *classical thinning algorithm* [45].

3.3 Intersection Detection

After the thinning procedure, the stripes are exactly one pixel wide and are connected in the 8-neighbors sense.

The candidates for intersection detection are the pixels lying on the thinned stripes, examined in a raster scanned manner. Detection is based on a set of conditions on the set of nonzero pixels lying in a square neighborhood centered about each candidate pixel. The detection conditions are a set of heuristics designed to test the hypothesis that the nonzero pixels in the square neighborhood are the thinned image of two intersecting stripes.

Condition A: The set of nonzero border pixels in the square neighborhood centered about the candidate pixel consists of exactly four connected subsets called border-connected components. If condition A is satisfied, then the remaining conditions are tested for each combination of four border pixels with one pixel taken from each border-connected component.

Condition B: A set of constraints on the lengths of the shortest paths connecting the center candidate pixel with each of the four border pixels.

Condition C: The set of nonzero pixels in the square neighborhood is exactly the union of two paths connecting pairs of border pixels, and with the candidate pixel required to lie on each of these paths.

Condition D: A constraint on the curvature of each of these two paths.

Condition E: A constraint on the angles of each of the two paths.

If each of the conditions A through E is satisfied for a candidate pixel and a quadruple of border pixel with one pixel taken from each of the border-connected components, then the geometric intersection of the two stripes is taken to be the set theoretic intersection of the two paths. The location of the intersection will later be estimated as the centroid of this connected component of pixels.

3.4 Grid Labeling

The main purpose of this procedure is to determine the correct correspondence between labeled intersections on the slide and detected intersections in the camera image. We have found, using the marked grid, that reliable grid labeling can be performed without epipolar constraints and using only a small subset of the constraints used by Stockman and Hu [58]. This is an advantage in automatic calibration, where it is necessary to perform grid labeling without prior knowledge of camera or projector geometry.

There are five steps in the direct construction of labels

1. Build a list of intersection adjacencies together with the directional sense of each adjacency.

2. Detect marker locations.

3. Construct the reduced graph.

4. Compute the longest paths.

5. Assign labels along each of the grid coordinate axis.

For more information and details of the grid labeling procedure, see [32].

3.5 Triangulation

Triangulation is the method by which the three dimensional location of points on the human body can be determined as the intersection of two lines: one line which passes through the camera lens center and the image of a grid intersection in the camera image plane, and a second line which passes through the projector lens center and the matching grid intersection on the slide.

Because of the quantization of the digital image and the numerical inaccuracies in determining the camera calibration and projector calibration matrices, those two lines usually come close to intersecting but do not actually meet. The usual approach to solve this problem is to find the *least squares intersection* of the two lines, which can be defined as the point in space which minimizes the sum of the squares of the distances to the two lines [1] and is given by

$$\begin{bmatrix} t_0 - ut_8 & t_1 - ut_9 & t_2 - ut_{10} \\ t_4 - ut_8 & t_5 - ut_9 & t_6 - ut_{10} \\ l_0 - pl_8 & l_1 - pl_9 & l_2 - pl_{10} \\ l_4 - ql_8 & l_5 - ql_9 & l_6 - ql_{10} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ut_{11} - t_3 \\ vt_{11} - t_7 \\ pl_{11} - l_3 \\ ql_{11} - l_7 \end{bmatrix} \tag{3.4}$$

where t_i , $0 \leq i \leq 11$ are the elements of camera calibration matrix, l_i, $0 \leq i \leq 11$ are the elements of projector calibration matrix, (u, v) are the image plane coordinates of a detected grid intersection and (p, q) is the grid label of the matching grid intersection on the slide. The unknowns are the object centered coordinates (x, y, z) of the point.

This procedure gives a set of points (x, y, z) on the human body which are used to reconstruct the surface.

Chapter 4

Segmenting the Burn Area

The problem at hand is how to identify which portion of the image is real burn wound area. In this chapter we will discuss several segmentation methods to solve the problem.

This is the most difficult step to automate in the entire system since the boundary may be irregular and the contrast will not be constant in an image and may vary from patient to patient. In this chapter we shall present 3 methods:

1 Edge extraction followed by edge following.

2 Multi-level thresholding.

3 Manual contouring.

for segmenting the burn area. The first two of these are automatic procedures, and if they fail then the burn area can be segmented by choice 3, manual contouring.

4.1 Edge Extraction Followed by Edge Following

4.1.1 Edge Extraction

Because of the importance of edges and contours which delineate image regions and encode shape information, few problems in computer vision have been more intensively

studied than that of edge detection. Even so, there are probably only modest differences in performance between the most sophisticated modern techniques and the edge operator originally described fifteen to twenty year ago.

Martelli [38] considered the edge-detection essentially as a tracking problem and used graphical theoretic techniques for its solution. Modestino and Fries [40] proposed a bidimensional recursive filtering scheme using statistical modeling of the image for edge detection. Basseville [6] had used a sequential algorithm for edge detection which uses a line-by-line detector of edge elements.

The early edge operators generally embodied a specific edge model; they were designed to locate step edges, or measure local intensity gradients. These operators are generally of a fixed size and do not address the issues of edge width and resolution [24].

The "zero-crossing operator" introduced by Marr and his co-workers [37] has one important feature: it explicitly deals with scale of resolution by smoothing, as well as the image-intensity surface; it smooths out the intensity variation below the spatial frequency of the edge it finds. John Canny [12] proposed a many zero-crossing scale implementation. The Canny edge operator is in wide use including medical applications. For example, Eichel implemented the Canny operator to identify coronary artery edges from cineangiograms [18].

Most of the existing approaches are based on the design of a detector that identifies likely edge pixels. Nalwa and Binford [42] seek to identify edgels – short linear edge elements – rather than individual edge pixels. Haralick [26] models the intensity of surface patches analytically, then deduces the presence of edges from the coefficients of the model. A competing approach is regularization: For example, Torre and Poggio [61] argue that the numerical differentiation of images required for edge detection is an

S S S

S S S

S S S

Figure 4.1: A 3 × 3 constant region

ill-posed problem that must be regularized by a filtering operation before differentiation.

The algorithm that we used poses the problem of edge extraction as a statistical classifier problem [34].

First, we assume the image is locally modeled as consisting of constant regions and two-valued regions as defined below.

Definition 1 *An $n \times n$ region is called a constant region if all the pixels inside the region have the same value, as in Figure 4.1 for a 3×3 region.*

Definition 2 *An $n \times n$ region is called a two-valued region if every pixel inside the region can have one of two possible values, either the higher pixel value S_h or lower pixel value S_l, as in Figure 4.2.*

This is not saying that the image consists of only two levels. All we are assuming is that over a small region of the image, the pixel values are either approximately constant or can have one of two possible values (a low value of low dispersion or a high value of low dispersion). The latter case belongs to an edge because most edges in the real

$$\begin{array}{ccc} S_l & S_l & S_l \\ S_l & S_h & S_h \\ S_l & S_h & S_h \end{array}$$

Figure 4.2: A 3 × 3 two-valued region

world are sharp edges where the intensity function is composed of a few step changes over a small number of pixels. An edge in the image is thus a collection of two-valued regions.

The algorithm of edge extraction that we are using consists of three major steps.

Step 1.

The program starts from moving a 3 × 3 window over the image in a raster scan fashion. At each instant, we labeled each pixel i inside the window as T_i, $i = 0, \ldots, 8$.

We define S_h and S_l as follow:

$$\begin{aligned} S_h &= \text{one below the maximum value of } T_i \\ &= \text{the eight order statistic of } T_i \\ S_l &= \text{one above the minimum value of } T_i \\ &= \text{the second order statistic of } T_i \end{aligned}$$

and assume that the 3 × 3 region being tested is a two-valued region. Then the range R_9 of the nine pixels in a two-valued region is

$$R_9 = S_h - S_l \tag{4.1}$$

If the 3×3 region is close to the edge, R_9 given by $S_h - S_l$ is likely to be large; and if the 3×3 is over a constant region, R_9 is likely to be small. So if $R_9 > \tau_1$ and *the center of the window* $< (S_h + S_l)/2$, the central pixel possibly belongs to an edge and we go to step 2. If the condition above fails, the central pixel is assumed to be part of a constant region. Then we move the window to the adjacent pixel in a raster scan fashion and start from step 1 all over again.

In the above definition, S_h and S_l could be approximated by the maximum and the minimum order statistics. However, the maximum and the minimum order statistics are prone to wide margins of error. Furthermore if we use a sorting algorithm to order the set of data T_i from small value to a large value, it is no more difficult to get the eight and the second order statistic of the set of data T_i.

At an edge, the width of the edge is two pixel thick; one belongs to the edge contour of the lower pixel value and the other belongs to the edge contour of the higher pixel value. The condition of "the pixel value of the center of the window $< (S_h + S_l)/2$" enables the algorithm to trace only the contour of the lower pixel value at any edge in the image.

In our program, we treat τ_1 as one of the input arguments. The choice of τ_1 determines the strength of the edges to be detected. For higher value of τ_1, only the sharper edges will be detected. To detect the relatively weaker edges, τ_1 has to be made smaller.

Step 2

Because the criteria of determining whether the test window belongs to a constant region or a two-valued region are not strict, more pixels are detected as candidates for

$$\begin{array}{ccccc} T_0 & T_1 & T_2 & T_3 & T_4 \\ T_5 & T_6 & T_7 & T_8 & T_9 \\ T_{10} & T_{11} & T_{12} & T_{13} & T_{14} \\ T_{15} & T_{16} & T_{17} & T_{18} & T_{19} \\ T_{20} & T_{21} & T_{22} & T_{23} & T_{24} \end{array}$$

Figure 4.3: A 5 × 5 window.

two-valued regions than there possibly are. In step 2, what we are going to do is to isolate those pixels which truly belong to two-valued regions among all possible pixels detected by step 1. We first increase the window size to 5 × 5 centered around the same pixel as the 3 × 3 window of step 1. Consider the 5 × 5 window shown in Figure 4.3.

We assume that every pixel inside a two-valued region has a Bernoulli distribution, and the pixels are independent of one another, i.e.:

$$T_i = \begin{cases} S_h & \textit{with probability } \rho \\ S_l & \textit{with probability } (1-\rho) \end{cases} \tag{4.2}$$

That is because the window size of its 5 × 5 region is considerably large and the pixel value of each individual pixel has no correlation to its neighbors.

We can also assume that ρ is 0.5. Over an infinite collection of two-valued regions, on an average S_h and S_l will appear an equal number of times. The expectation of T_i is:

$$\begin{aligned} E(T_i) &= \textstyle\sum_i \rho T_i \\ &= \frac{S_h}{2} + \frac{S_l}{2} \\ &= \frac{S_h+S_l}{2} \end{aligned} \tag{4.3}$$

and the variance of T_i is :

$$\begin{aligned} \sigma^2(T_i) &= \textstyle\sum_i (x_i - \eta)^2 \\ &= E(x^2) - E^2(x) \\ &= \frac{S_h^2}{2} + \frac{S_l^2}{2} - (\frac{S_h+S_l}{s})^2 \\ &= (\frac{S_h-S_l}{2})^2 \end{aligned} \tag{4.4}$$

The problem now is to classify the pixels inside the 5×5 region into two distinct classes when the pixels have the distribution given by equation 4.2.

Denoting $\overline{T}_{25} = \sum_{i=0}^{24} T_i/25$. It can be readily shown:

$$E(\overline{T}_{25}) = \frac{\sum_{i=0}^{24} E(T_i)}{25} = \frac{S_h + S_l}{2} \tag{4.5}$$

$$\sigma^2(\overline{T}_{25}) = \frac{1}{(25)^2} \sum_{i=0}^{24} \sigma^2(T_i) = \frac{1}{25}(\frac{S_h - S_l}{2})^2 = (\frac{S_h - S_l}{10})^2 \tag{4.6}$$

By the central limit theorem which states that if the random variables x_i are independent, then under general condition, the density $f(x)$ of their sum $x = x_1 + \ldots + x_n$ properly normalized, tends to a normal curve as $n \rightarrow \infty$. In other words, if n is sufficiently large, then

$$f(x) \cong \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\eta)^2/2\sigma^2} = N(\eta\ ,\ \sigma) \tag{4.7}$$

Thus the equations 4.5 and 4.6 can be rewritten as

$$\overline{T}_{25} \cong N((S_h + S_l)/2, (S_h - S_l)/10) \tag{4.8}$$

Consider the statistic

$$s_i = \overline{T}_{25} - T_i \tag{4.9}$$

We shall use the statistic s_i to decide whether the pixel T_i has value S_h or S_l. In order to do so, we define two hypotheses. Under the first hypothesis H_0, called the null hypothesis, the pixel T_i has the value S_l. Under the second hypothesis H_1, called the alternative hypothesis, the pixel T_i has the value S_h.

So under H_0,

$$s_i = \overline{T}_{25} - S_l \tag{4.10}$$

has an approximate Gaussian distribution given by

$$N(\frac{24}{50}(S_h - S_l), \frac{S_h - S_l}{2\sqrt{26}}) \tag{4.11}$$

Under H_1,

$$s_i = \overline{T}_{25} - S_h \tag{4.12}$$

has an approximate Gaussian distribution given by

$$N(\frac{24}{50}(S_l - S_h), \frac{S_h - S_l}{2\sqrt{26}}) \tag{4.13}$$

Now we define

$$\begin{aligned} D &= E(s_i)|_{H_0} - E(s_i)|_{H_1} \\ &= \tfrac{24}{50}(S_h - S_l) - \tfrac{24}{50}(S_l - S_h) \\ &= \tfrac{48}{50}(S_h - S_l) \end{aligned} \tag{4.14}$$

and

$$D/\sigma = \frac{\frac{48}{50}(S_h - S_l)}{\frac{S_h - S_l}{2\sqrt{26}}} \cong 9.8 \tag{4.15}$$

This implies that the mean values under the hypotheses H_0 and H_1 are separated by approximately 10 times the standard deviation. In [34], Kundu and Mitra found the upper tail of s_i under H_1 at a distance 7.5σ from its mean or -2.5σ from the mean of s_i under H_0 is almost negligible. So they used this observation as decision rule instead of using zero which will identify a constant region as a two-valued region incorrectly because of the variance over a region even when it is a small region.

So under H_0:

$$\begin{aligned} E(s_i)|_{H_0} - 2.5\sigma &= \tfrac{24}{50}(S_h - S_l) - 2.5 \times \tfrac{(S_h - S_l)}{2\sqrt{26}} \\ &= \tfrac{11.5}{50}(S_h - S_l) \\ &> \tfrac{S_h - S_l}{5} \\ &= \tfrac{\Theta_{max-3} - \Theta_{min+3}}{5} \end{aligned} \tag{4.16}$$

where

S_h = 22nd order statistic of $T_0, \ldots, T_{24}$

$= \Theta_{max-3}$

S_l = 4th order statistic of $T_0, \ldots, T_{24}$

$= \Theta_{min+3}$

Let $S_h = m \times S_l$ where $m > 1$. Then from 4.16

$$\begin{aligned} E(s_i)|_{H_0} - 2.5\sigma &> \frac{m-1}{5m} S_h \\ &= \frac{\Theta_{max-3}}{\tau_2} \end{aligned} \tag{4.17}$$

where we replaced $5m(m-1)^{-1}$ by τ_2 which is another argument of the program. If $s_i > \Theta_{max-3}/\tau_2$, T_i is to have the value S_l, and vice versa. So at this step, every pixel T_i is tested using the statistic s_i to determine whether it has the value S_l or the value S_h.

Step 3

In this step, the program will decide whether or not the central pixel belongs to a two-valued region. In the program, the variable "count" gives the number of pixels having the value S_l inside the 5 × 5 window. According to the prior knowledge, we have 25 elements (trials) for each 5 × 5 window and the probability for each element is 0.5. Therefore, the distribution of the variable "count" is binomial with mean 25×0.5 and the variance $25 \times 0.5 \times 0.5$. We define a new variable "Stat" such that

$$Stat = \frac{count - 12.5}{\sqrt{25 \times 0.5 \times 0.5}} = \frac{count - 12.5}{2.5} \tag{4.18}$$

The distribution of *Stat* is nearly a Gaussian distribution with zero mean and unity standard deviation. Since $\pm 2.3\sigma$ limit contains 99 percent of the probability in the Gaussian distribution [44]. $p\{|Stat| = \frac{x-\eta}{\sigma} = \frac{count-12.5}{2.5} \leq 2.3\} \cong 0.98928$

In the program, we replaced 2.3 by a general argument τ_3. So if the absolute value of *Stat* is less than or equal to τ_3 , the central pixel belongs to a two-valued region. The corresponding pixel in the edge extraction output image is assigned the value 255. Otherwise, the central pixel does not belong to two-valued region and the corresponding pixel in the edge extraction output image is assigned the value 0. If the value of *count*

in step 2 is always less than or equal to 25, assign the pixel value *count* $\times$ 10 to the central pixel of its corresponding pixel in another output image called potential edge image. Thus, any point in the potential edge image which has the higher pixel value has a higher probability of becoming an edge point.

After determining if the central pixel belongs to the two-valued region or not, and assigning the proper pixel value to its corresponding pixel in output image, the program will move the window to the adjacent pixel in a raster scan fashion and start step 1 all over again. The flow chart of this program is shown in Figure 4.4.

The threshold τ_1 controls the complexity of the algorithm by limiting the number of pixels to be considered in step 2 and step 3, τ_2 controls the distribution of the sharp edges and the weak edges, and τ_3 controls the correct identification of true edge elements and controls the thickness of the detected edges. To detect weak edges, τ_2 should be increased. A larger value of τ_3 will make the detected edges thicker. Generally, the value of τ_3 lies between 1 and 3.

4.1.2 Edge Following

For fairly clear images, conventional edge extraction algorithms are generally able to obtain the closed contours of the objects in the image. However, when an image is contaminated by noise, it becomes difficult to extract perfectly closed contours, and the contours extracted are most likely broken. When such a situation occurs, as in our example, it becomes very difficult for the system to extract reliable features, such as burn wound area, from those imperfect contours for successful recognition or interpretation.

In our case, we have already detected the partial region of the burn wound image, but nothing is known about the boundary shape. Ballard and Brown [5] suggested that the boundary is recovered by one of the simplest edge-following operations: "blob

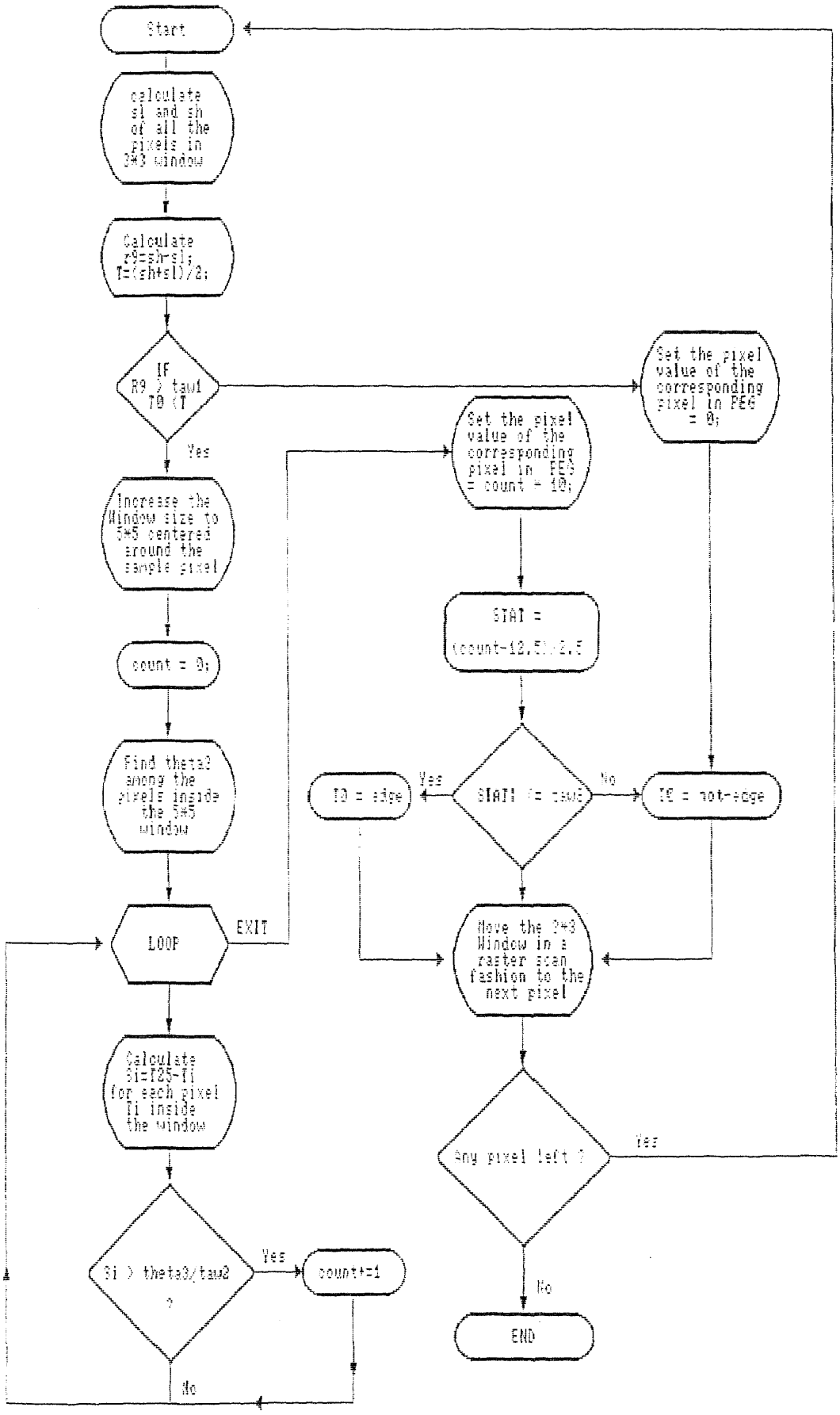

Figure 4.4: The flow chart of Edge Extraction Algorithm

finding" in images. Rosenfeld and Kak [56] implemented this by first finding a four-connected background pixel from a known boundary pixel, then the next pixel is the first pixel encountered when the eight neighbors are examined in a counter clockwise order from the background pixel.

Our algorithm for extracting closed contours from edge pixels is very closely related to Chen's forward/backward contour tracing [13]. The basic algorithm consists of the following two stages:

1. Starting point determination.

2. Forward and backward tracing.

Starting Point Determination

The main purpose of this stage is to derive directly from the image all the relevant parameters that will be used as the starting point criteria in the edge extraction procedure.

Recall that, in the edge extraction procedure we have two output images, one is the edge-point image and the other is the potential-edge-point graph. From now on we will call the edge point image derived from edge extraction procedure the candidate-edge-point image, because all of the edge points shown on that image are not yet desired contour points. Our goal is to have all the contour points belong to a certain closed path and all the points in the path are edge points detected from the edge extraction procedure. Not all the edge points on edge-point image are the final edge points: they are only candidate edge points.

The program begins with a scan process and scans the candidate-edge-point image from left to right and top to bottom until it encounters a starting point which satisfies all the following conditions which we called starting point criteria:

1. It is a candidate edge point.

2. At least one of its eight neighbors is also a candidate edge point.

This definition limits the starting point (SP) picked in the scan process to be a candidate boundary point. Then, after the position of SP is recorded and is assigned to be the root or the head of a binary tree which contains all the boundary points (contour points) and named "contour-path", the scan process is suspended and the trace process is initiated.

The trace process has two main functions. One is to search for the next boundary point. The other is to check and record the termination conditions for the boundary being traced.

Forward and Backward Tracing

This process begins with SP. The first step in the tracing procedure is to look for the next boundary point by determining which one of the SP's eight neighbors has the highest potential edge value corresponding to the pixel value of the potential-edge-point image. If the direction between SP and the neighbor with highest potential value is 0, 1, 2, or 3, then the tracing procedure to find the next boundary points will be in the clockwise manner, otherwise it will be in counter clockwise manner for the forward tracing. The definition of direction between the center point and its eight neighbors are shown in Figure 4.5.

Whenever the direction and the next boundary point is found, the boundary point chosen is assigned to be current point P and is also placed into the boundary tree "contour-path".

The data structure of the binary tree "contour-path" is defined as : the head of this tree contains the information of SP, all the following boundary points detected in

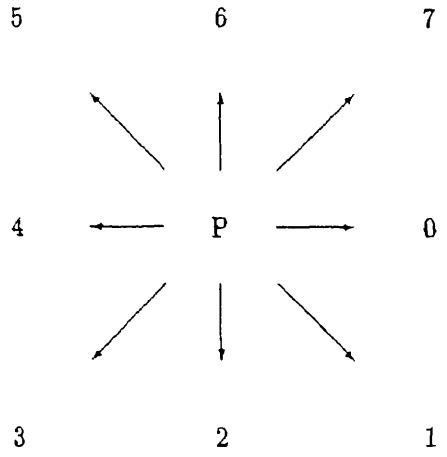

Figure 4.5: The directions of eight neighbors

the clockwise direction are assigned to the left direction of this tree and the boundary points detected in the counter clockwise direction are assigned to the right direction of this tree.

Each boundary point in turn becomes the current point for the trace process. In looking for the next boundary point of current point P, only three out of eight of its neighbors are examined. This is based on the facts that successive boundary points are adjacent to one another. These three neighbors of P are called the *search area* for next boundary point and are determined by which one of them has the highest potential value corresponding to the pixel value of the potential-edge-point image. The eight possible search areas are shown in Figure. 4.6.

When ties occur, the middle point in the search area is selected. If there are two points in the search area that satisfy the criteria, then we pick the first point that we

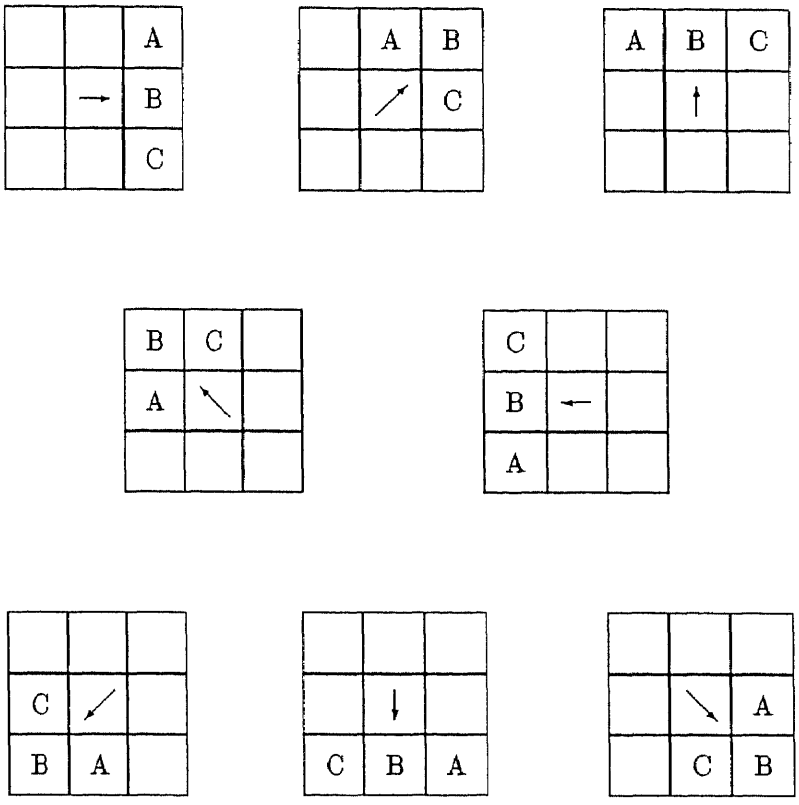

Figure 4.6: Search Area

find. The trace process continues in this pattern until one of the following termination conditions occurs:

1. The next boundary point is the starting point.
2. The next boundary point is an image border point, i.e., the contour path has hit the image border.
3. The next boundary point is already in the contour path.
4. The potential value of the next boundary point corresponding to its potential-edge-image is zero and the next boundary point is not a border point.

Condition 1 implies that a closed boundary has been successfully traced out. In this case, the boundary will be flagged as closed. Then the scan process will be resumed, starting with a point next to the original starting point in raster scan manner. Condition 2 implies that the object is partially visible. Condition 3 implies that the boundary intersects itself or other boundary at some point. Condition 4 implies a dead end or gap of the boundary encountered.

If Condition 4 occurs, then we need to do more work to determine if there is a real gap or if a dead end has been encountered in that region. To do so, we increase the search area as shown in Figure 4.7. We determine which one of the pixels in such enlarged search area has the highest potential edge value from the potential-edge-point image. If there is a point in the enlarged search area that satisfies the above criteria, then such point becomes the next boundary point. We not only assign the current point P to the contour path but also the point on the gap by linear interpolation between the current point P and the next boundary point, and continue tracing boundary points by determining the potential edge value of its three out of eight neighbors.

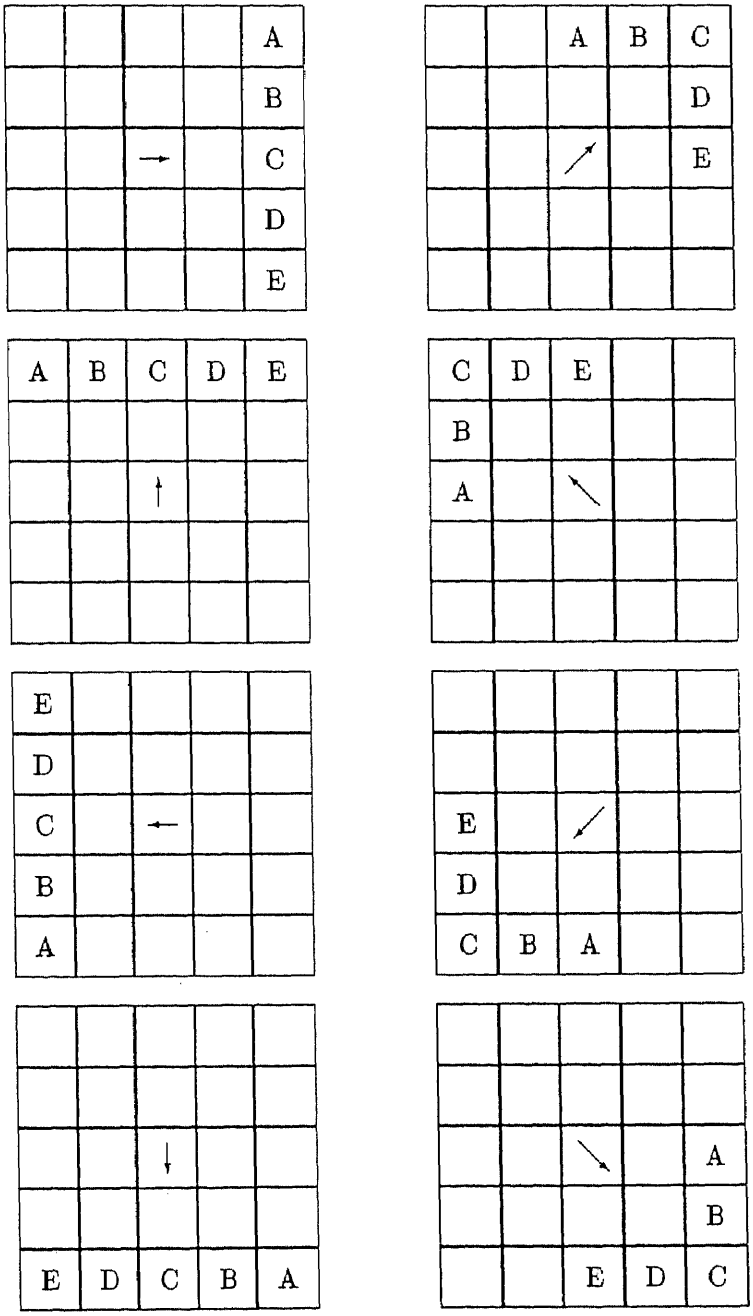

Figure 4.7: Large search area

If there are no points in the enlarged search area that satisfy the above criteria, then we say that the termination condition 4 does really occur and there is a dead end.

If termination condition 2, 3 or 4 occurs, then the trace process begins to traverse the other portion of the boundary. The tracing begins with the original starting point SP, but traverses the boundary in the backward manner which is the opposite direction to the forward tracing. In other words, if the forward tracing procedure is in a clockwise manner, then the backward tracing procedure will be in a counterclockwise manner. This boundary tracing procedure continues until one of the above four termination conditions occurs.

In addition, if both the forward and backward tracing procedures terminate at condition 2, then the object is considered to be partially visible and we say such contour path is closed and efficient. If either termination condition 3 or 4 occurs in either the forward or backward trace process, we have to determine if the contour path is efficient or not by counting the length of the contour path. If it is larger than a given threshold, then we keep this path as boundary, otherwise discard the whole path.

After the backward trace process, the scan process is resumed to find the starting point of another boundary, if any. In this manner, the scan process and the trace process alternate each other until the lower right corner of the image is reached in the scan process.

4.2 Multi-level thresholding

In this section we consider another way to handle the segmentation problem. The idea of this approach came from clustering: grouping similar pixels into regions. There are basically three kinds of "objects" in a burn image: background, normal or healthy skin

and burn wound skin. The regions in a burn area image can be more carefully classified using five kinds of "objects": background, normal skin, first degree burn wound skin, second degree burn wound skin, and third degree burn wound skin. So for this problem, we can use the idea of clustering the gray levels of the image's histogram into many clusters separating objects (normal skin and burn wound skin) from their background.

Our idea is based on observation and is very closely related to nonhierarchical clustering methods which choose an initial partition of the data and then alter cluster memberships so as to obtain a better partition. There are various algorithms which have been proposed and they differ as to what constitutes a "better partition" and the methods used for achieving improvements.

We begin by choosing an intial partition of the data into groups or using a set of seed points around which clusters may be found [2]. The following methods are representative examples of how such seed points can be generated and we assume that a set of k seed points can be used as cluster nuclei around which the set of m data can be grouped.

1. Choose the first k data units in the data set. If the initial configuration does not influence the ultimate outcome in any important way, then this method is the cheapest and simplest.

2. Label the data from 1 to m and choose those labeled m/k, $2m/k$, ..., $(k-1)m/k$, *and* m This method is almost as simple as method 1 but tries to compensate for a natural tendency to arrange the data units in the order of collection or some other nonrandom sequence.

3. Subjectively choose any k data units from the data set.

4. Label the data units from 1 to m and choose the data units corresponding to k different random numbers in the range 1 to m.

5. Generate k synthetic points as vectors of coordinates where each coordinate is a random number from the range of the associated variable. Unless the data set "fills" the space, some of these seed points may be quite distant from any of the data units.

6. Take any desired partition of the data units into k mutually exclusive groups and compute the group centroid as seed points.

7. An intuitively appealing goal is to choose seed points which span the data sets, that is, most data units are relatively close to a seed point but the seed points are well separated from each other.

8. Ball and Hall [4] suggested taking the overall mean vector of the data set as the first seed point; select sequence seed points by examining the data units in their input sequence and accept any data unit which is at least some specified distance, say d, from all previously chosen seed points; continue choosing points until k seed points are accumulated or the data set is exhausted. This method is sufficiently simple that two or three values of the threshold distance d could be tried if the first value gave too few seed points or examined too little of the data set.

Any one of these methods can be used to generate the initial partition of the data. Next, we discuss how to cluster the data using an initial partition.

In our case, we are dealing with multithresholding, that is we want to separate the pixels in the image data into several clusters. We can use MacQueen's idea [36] of using the term "k-means" to denote the process of assigning each datum to a cluster

(of k clusters) with the nearest centroid (mean). The key implication in this process is that the cluster centroid is computed on the basis of the cluster's current membership. MacQueen's algorithm for sorting m data units into k clusters is composed of the following steps:

1. Take the first k data units in the data set as clusters of one member each.

2. Assign each of the remaining $m - k$ data units to the clusters with the nearest centroid. After each assignment, recompute the centroid of the cluster.

3. After all data units have been assigned in step 2, take the existing cluster centroids as fixed seed points and make one more pass through the data set assigning each data unit to the nearest seed point.

Hartigan and Wong [27] present an algorithm for k-means clustering which produces a global minimum only for the two-means case. Pollard [49] discusses the convergence of the k-mean clustering. However, our approach is much more closely related to ISODATA which is the most elaborate of the nearest centroid clustering method and was first developed at the Stanford Research Institute.

Ball and Hall [3] present a complete step-by-step description of the original method and illustrate it in great detail with a two-dimensional example. Velasco [63] points out that the ISODATA can be used to requantize images into a specified number of gray levels.

The basic ISODATA algorithm [15] is a procedure for classifying a set of sample vectors $\{\boldsymbol{x} = \bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n\}$ into c distinct classes.

Basic ISODATA

1. Choose some initial values for means $\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_c$.

Loop: 2. Classify the n samples by assigning them to the class of the closest mean.

3. Recompute the means as the average of the samples in their class.

4. If any mean has changed value, go to Loop; otherwise, stop.

In our case, we have an image in which each point has a "gray level" in the interval [0,G] and where G may go up to 255. Each sample is a point of the image. Instead of calculating the distribution of gray levels of a whole image of pixels , we only take the gray levels of the real edge points into account. In other words, we only collect the gray levels of those points which are the edge points in the burn image corresponding to the output image of edge extraction procedure. We have adopted this approach to reduce the effect of noise in the image. The result of the basic ISODATA is shown in Figure 4.8 which can be compared to Figure 7.6. In this manner all of the histogrammed points are edge points, eliminating the effect of noise. This can be easily seen from Figure 4.9.

In the upper left is the raw image, in the upper right is the histogram of the raw image. The lower left is the edge extraction output and the lower right is the gray level histogram of the edge points. It can be easily seen that we can discern more clusters from the histogram of the edge points. So in our program, we have to input two images which are the raw image of burn patient and the edge point image from the output of edge extraction procedure.

In the modified ISODATA algorithm, we let $[G_{low}, G_{high}]$ denote the smallest interval containing all nonzero histogram values.

Edge Point ISODATA

1 Choose some initial values for the means $u_1, u_2, \ldots, u_c$ such that $G_{low} \leq u_1 < u_2 <$

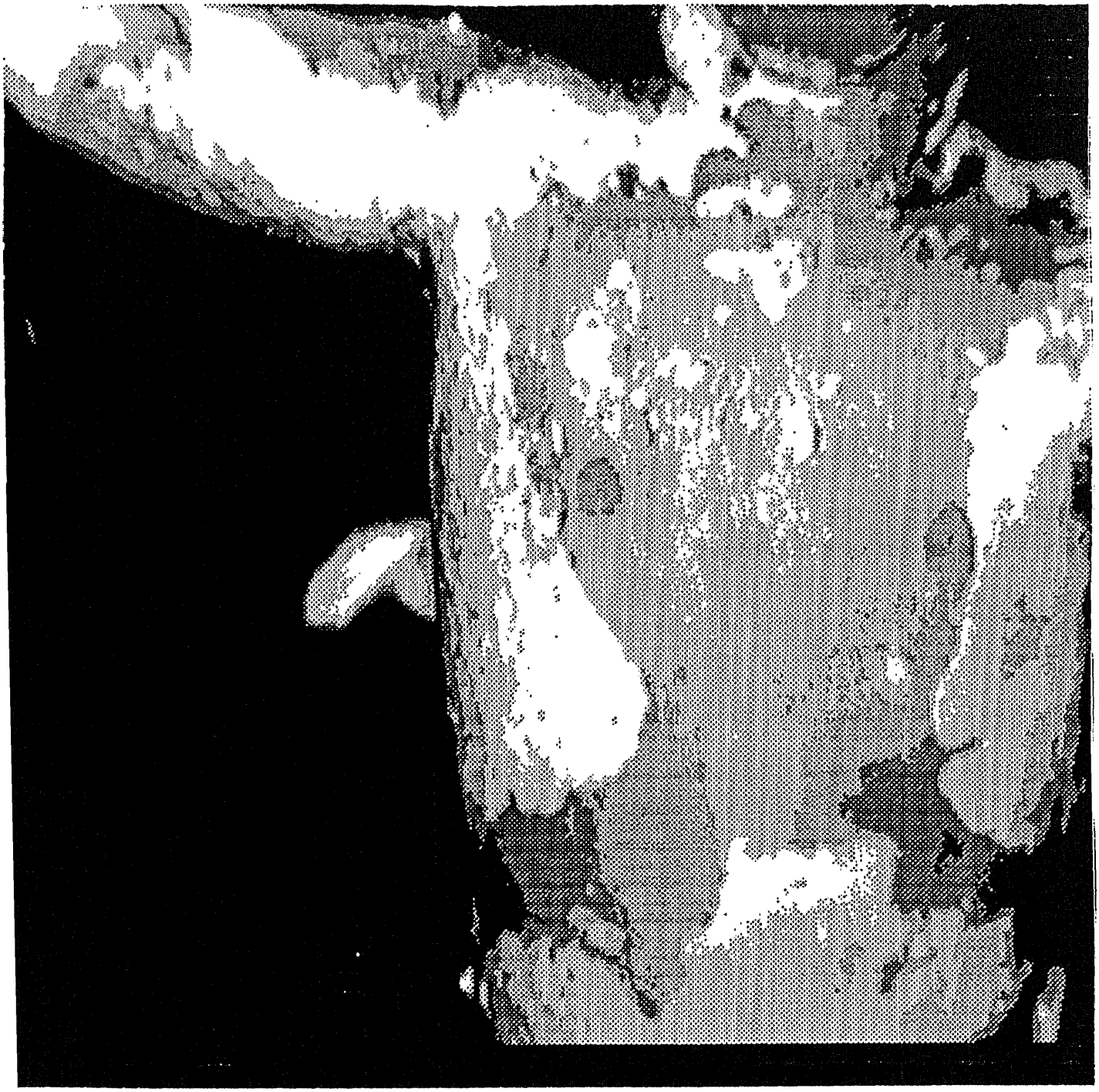

Figure 4.8: The result of basic ISODATA

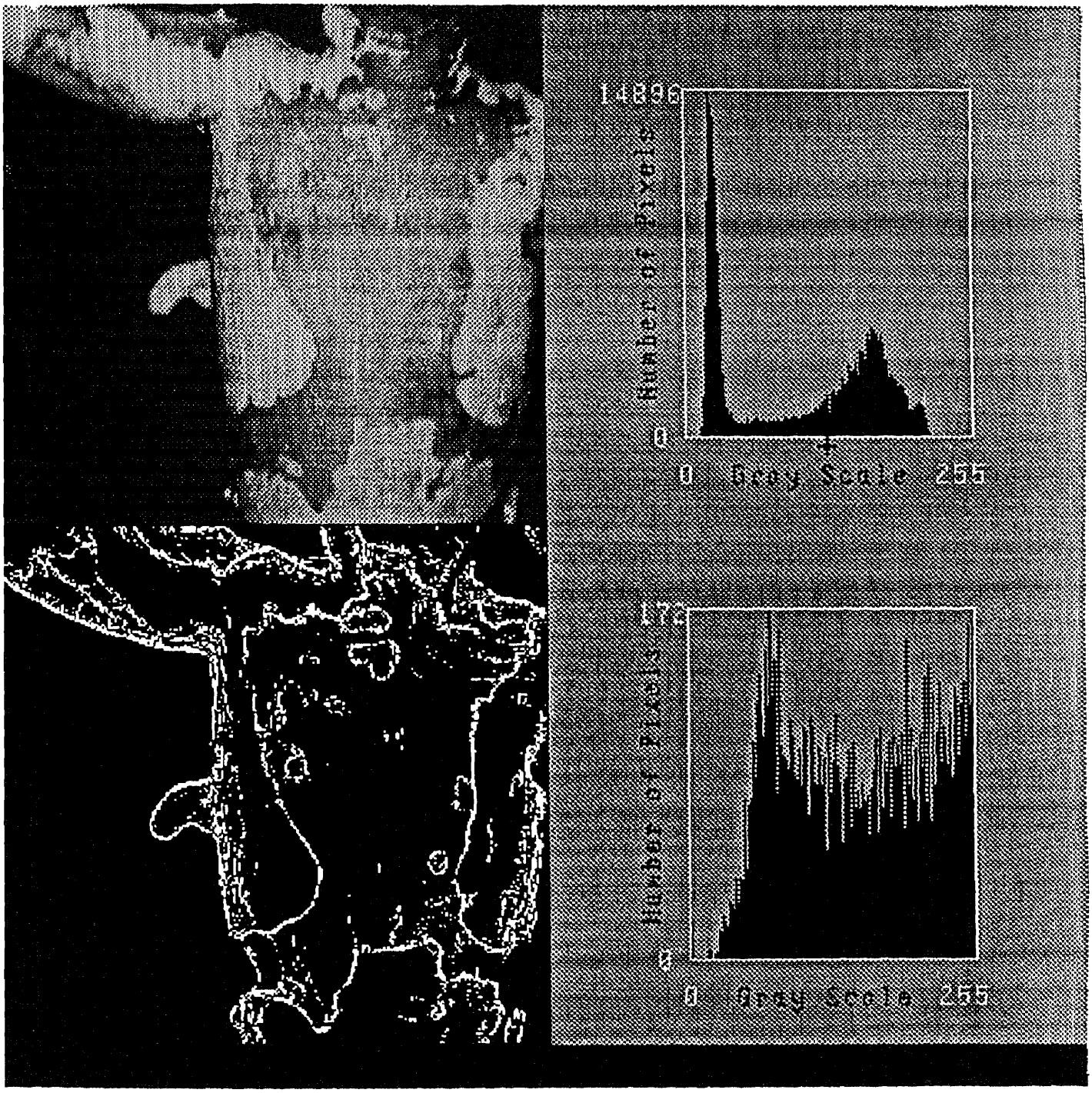

Figure 4.9: Display the histogram

$\ldots < u_c < G_{high}$

Loop: 2 Calculate thresholds $T_1, T_2, \ldots, T_{c-1}$ by the formula

$$T_i = \lfloor (u_i + u_{i+1})/2 \rfloor, \qquad 1 \leq i \leq c \tag{4.19}$$

where $\lfloor x \rfloor$ is the largest integer not greater than x. Assign to class i , $1 \leq i < c$, all gray levels in the interval $I_i = [T_{i-1} + 1, T_i]$ And $T_0 = G_{low} - 1$, $T_c = G_{high}$.

3 Recompute the means : for every i make $\hat{u_i}$ the nearest gray level to

$$(\sum_{j \in I_i} j \cdot h(j) / \sum_{j \in I_i} h(j)), \qquad 1 \leq i \leq c \tag{4.20}$$

4 If any means has changed value, go to Loop; otherwise go to the next step.

5 Requantize the raw image based upon the threshold values.

We can choose our initial threshold values which can split the set of data units into desire groups and then get the initial values for the means by two different ways:

1. divided the interval $[G_{low}, G_{high}]$ into the number of classes groups constantly. In other words, the initial values for the means were chosen so that they were evenly distributed in the interval $[G_{low}, G_{high}]$.

2. divided the interval $[G_{low}, G_{high}]$ by its overall mean value. Because the overall mean value is always in the interval $[G_{low}, G_{high}]$, we can separate the different number of classes by:

   A If there are two classes image then the initial threshold value is assigned to be the overall mean value.

B Three classes then the two thresholds are

$$T_0 = \lfloor (mean + G_{low})/2 \rfloor; \tag{4.21}$$

$$T_1 = \lfloor (mean + G_{high})/2 \rfloor; \tag{4.22}$$

C Four classes. Then the three threshold values are:

$$T_0 = \lfloor (mean + G_{low})/2 \rfloor; \tag{4.23}$$

$$T_1 = mean \tag{4.24}$$

$$T_2 = \lfloor (mean + G_{high})/2 \rfloor; \tag{4.25}$$

D Five classes. Then the four threshold values are:

$$T_0 = \lfloor (mean + G_{low})/3 \rfloor; \tag{4.26}$$

$$T_1 = \lfloor (mean + G_{low}) \times 2/3 \rfloor; \tag{4.27}$$

$$T_2 = \lfloor (mean + G_{high})/3 \rfloor; \tag{4.28}$$

$$T_3 = \lfloor (mean + G_{high}) \times 2/3 \rfloor; \tag{4.29}$$

Velasco [63] proves that the algorithm terminates in a finite number of steps, but does not have to always arrive at the same threshold, irrespective of the initial value of the means. This does not necessarily hold for the edge point algorithm.

For display purposes we use threshold values $T_1, T_2, \ldots, T_{c-1}$, we requantize the raw image by assigning its corresponding new gray levels as $255 \times i/(c-1)$ if the gray level of the raw image lies between $[T_i, T_{i+1}]$, $0 \leq i \leq c-1$ where $T_0 = 0$ and $T_c = 255$.

Chapter 5

Surface Representation

By using our structured light system and triangulation, we can fine the location (x_i, y_i, z_i) of a set of points on the surface of the human body. In this chapter, we will show how to reconstruct the surface from the set of points (x_i, y_i, z_i). We shall begin by choosing a representation of space curves. Each of the points (x_i, y_i, z_i) will be parameterized by its grid labels assigned in the grid labeling procedure, thus treating the data as the points $[x_i(u, v), y_i(u, v), z_i(u, v)]$ where the grid labels (u_i, v_i) are the independent variables. Furthermore, each of these points lies on several curves, i.e., a network of stripes. We shall represent this network by *cubic splines*. The surface will be represented by *bicubic surface patches*. In this chapter we shall outline both representations.

5.1 Cubic Spline

In general, a spline is a piecewise polynomial of degree K with continuity of derivatives of order $K - 1$ at the common joints between segments. Thus, a cubic spline has second-order continuity at the joints. Piecewise splines of low degree polynomials are usually more useful for forming a curve through a series of points. The use of low degree polynomial reduces the computational requirements and reduces numerical instabilities that arise with higher order curves. These instabilities can cause an undesirable varia-

tion when several points must be joined in a common curve. However, since low degree polynomials cannot span an arbitrary series of points, adjacent polynomial segment are needed. The cubic spline is advantageous since it is the lowest degree space curve which allows a point of inflection and has the ability to twist through space.

The equation for a single parametric cubic spline segment, in terms of a parameter t, is given by

$$P(t) = \sum_{i=0}^{3} B_i t^i; \qquad t_1 \leq t \leq t_2 \tag{5.1}$$

where

$$P(t) = [x(t) \quad y(t) \quad z(t)]$$

$P(t)$ is the position vector on the space curve. It has three components $x(t), y(t)$ and $z(t)$ which are considered the cartesian coordinates of the position vector. The coefficients B_i are determined by specifying four boundary conditions for the spline segment.

In expanded form 5.1 may be written as

$$P(t) = B_0 + B_1 t + B_2 t^2 + B_3 t^3 \tag{5.2}$$

Let a given pair of points through which a curve segment passes be the vectors P_1 and P_2. The corresponding tangent vectors at these given points are indicated by P_1', P_2', the derivatives with respect to the parameter t. Within the cubic segment the parameter t varies between two end-point values t_1 and t_2. To simplify the calculations, we can assign $t_1 = 0$, fixing the one point.

The single segment between P_1 and P_2 consists of the two end points and the tangent vector at each end point. These conditions are given by

$$P(0) = P_1 \tag{5.3}$$

$$P(t_2) = P_2 \tag{5.4}$$

$$\frac{dP}{dt}|_{t=0} = P_1^{'} \tag{5.5}$$

$$\frac{dP}{dt}|_{t=t_2} = P_2^{'} \tag{5.6}$$

From 5.2 and 5.3 we see that

$$P(0) = B_0 = P_1 \tag{5.7}$$

$$P(t_2) = B_0 + B_1 t_2 + B_2 t_2^2 + B_2 t_2^3 \tag{5.8}$$

$$\frac{dP}{dt}|_{t=0} = B_1 = P_1^{'} \tag{5.9}$$

$$\frac{dP}{dt}|_{t=t_2} = B_1 + 2B_2 t + 3B_3 t^2 \tag{5.10}$$

Solving for B_2 and B_3 yields

$$B_2 = \frac{3(P_2 - P_1)}{t_2^2} - \frac{2P_1^{'}}{t_2^2} - \frac{P_2^{'}}{t_2^2} \tag{5.11}$$

$$B_3 = \frac{2(P_1 - P_2)}{t_2^3} + \frac{P_1^{'}}{t_2^2} + \frac{P_2^{'}}{t_2^2} \tag{5.12}$$

along with $B_0 = P_1$ and $B_1 = P_1^{'}$. These values of B_0, B_1, B_2, and B_3 fix the curve for the cubic segment. The shape of the cubic spline segment depends on the end-point position and tangent vectors.

Substituting 5.7, 5.8, 5.9, and 5.10 into 5.2 yields

$$P(t) = P_1 + P_1^{'}t + [\frac{3(P_2 - P_1)}{t_2^2} - \frac{2P_1^{'}}{t_2} - \frac{P_2^{'}}{t_2}]t^2 + [\frac{2(P_1 - P_2)}{t_2^3} + \frac{P_1^{'}}{t_2^2} + \frac{P_2^{'}}{t_2^2}]t^3 \tag{5.13}$$

For two adjacent cubic segments, the second derivative must be continuous at the internal joints. From the generalized equations for two adjacent segments

$$P_K(t) = P_K + P_K' t + [\frac{3(P_{K+1} - P_K)}{t_2^2} - \frac{2P_K'}{t_2} - \frac{P_{K+1}'}{t_2}]t^2 + [\frac{2(P_K - P_{K+1})}{t_2^3} + \frac{P_K'}{t_2^2} + \frac{P_{K+1}'}{t_2^2}]t^3 \quad (5.14)$$

and

$$P_{K+1}(t) = P_{K+1} + P_{K+1}' t + [\frac{3(P_{K+2} - P_{K+1})}{t_3^2} - \frac{2P_{K+1}'}{t_3} - \frac{P_{K+2}'}{t_3}]t^2 + [\frac{2(P_{K+1} - P_{K+2})}{t_3^3} + \frac{P_{K+1}'}{t_3^2} + \frac{P_{K+2}'}{t_3^2}]t^3 \quad (5.15)$$

Here we assume that the parameter variation is $0 \leq t \leq t_2$ for the first segment and $t_2 \leq t \leq t_3$ for the second segment.

From 5.1 we have

$$P'' = \sum_{i=0}^{3}(i)(i-1)B_i t^{i-2};\ t_1 \leq t \leq t_2 \quad (5.16)$$

At the end of the first cubic spline segment, where $t = t_2$

$$P'' = 6B_3 t_2 + 2B_2$$

and at the beginning of the second spline segment, where $t = t_2$,

$$P'' = 2B_2$$

Equating these two results and using 5.11 and 5.12 yields

$$6t_2[\frac{2(P_1-P_2)}{t_2^3} + \frac{P_1'}{t_2^2}+\frac{P_2'}{t_2^2}] + 2[\frac{3(P_2-P_1)}{t_2^2} - \frac{2P_1'}{t_2} - \frac{P_2'}{t_2}]$$
$$= 2[\frac{3(P_3-P_2)}{t_3^2} - \frac{2P_2'}{t_3} - \frac{P_3'}{t_3}] \qquad (5.17)$$

Multiplying by t_2t_3 and collecting terms gives

$$t_3P_1' + 2(t_2+t_3)P_2' + t_2P_3' = \frac{3}{t_2t_3}[t_2^2(P_3-P_2) + t_3^2(P_2-P_1)] \qquad (5.18)$$

which can be solved for P_2', the unknown tangent vector at the internal joint.

For n data points the results given above can be generalized to yield $n-1$ cubic spline segments with position, slope, and curvature continuity at all the internal joints.

The corresponding matrix equation for all segments of the piecewise cubic is therefore

$$\begin{bmatrix} t_3 & 2(t_2+t_3) & t_2 & 0 & 0 & \dots \\ 0 & t_4 & 2(t_3+t_4) & t_3 & 0 & \dots \\ 0 & 0 & t_5 & 2(t_4+t_5) & t_4 & \dots \\ & \vdots & & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ P_n' \end{bmatrix}$$
$$= \begin{bmatrix} \frac{3}{t_2t_3}[t_2^2(P_3-P_2)+t_3^2(P_2-P_1)] \\ \frac{3}{t_3t_4}[t_3^2(P_4-P_3)+t_4^2(P_3-P_2)] \\ \vdots \\ \frac{3}{t_{n-1}t_n}[t_{n-1}^2(P_n-P_{n-1})+t_n^2(P_{n-1}-P_{n-2})] \end{bmatrix} \qquad (5.19)$$

The expansion of 5.19 gives $n-2$ equations in n unknown tangent vectors. When the two end tangent vectors P_1' and P_n' are specified, the system of equations is determined. Thus, to generate a curve, the position vectors P_i, $1 \le i \le n$, along with P_1' and P_n', are specified. Then 5.19 is used to calculate the intermediate tangent vectors $P_2', P_3', \dots, P_{n-1}'$. The tangent vectors are then used to calculate the coefficients B_i given by the generalized form of 5.7, 5.9, 5.11, and 5.12 for each curve segment,

$$B_0 = P_K \tag{5.20}$$

$$B_1 = P_K^{'} \tag{5.21}$$

$$B_2 = \frac{3(P_{K+1} - P_K)}{t_{K+1}^3} - \frac{2P_K{'}}{t_{K+1}} - \frac{P_{K+1}^{'}}{t_{K+1}} \tag{5.22}$$

$$B_3 = \frac{2(P_K - P_{K+1})}{t_{K+1}^3} + \frac{P_K^{'}}{t_{K+1}^2} + \frac{P_{K+1}^{'}}{t_{K+1}^2} \tag{5.23}$$

Now consider the boundary conditions. Many choices exist for specifying the end boundary conditions for a piecewise cubic curve. Several different choices may be desirable if only a few data points are known or if physical constraints require accurate control of the curve shape at the ends. The most direct solution is obtained by specifying the two end tangent vectors $P_1^{'}$ and $P_n^{'}$ of the total piecewise spline. This boundary is called *the clamped end condition* and *the encastered spline.* For this application, we are using the condition called either *relaxed* or *natural condition* which considered the mathematical end condition that $d^2P/dt^2 = 0$. Setting 5.16 equal to zero for the first span ($K = 1$), with $t = 0$, and using 5.11 leads to

$$P_1^{'} + \frac{1}{2}P_2^{'} = \frac{3}{2}\frac{(P_2 - P_1)}{t_2} \tag{5.24}$$

If the same end condition is used for the last span ($K = N - 1$), with $t = t_n$, then from 5.22 and 5.23 one obtains

$$2P_{n-1}^{'} + 4P_n^{'} = \frac{6}{t_n}(P_n - P_{n-1}) \tag{5.25}$$

Finally, each cubic segment is generated by use of 5.1 , with $0 \leq t \leq t_{max}$. Before the curve can be generated, the maximum parameter value t_{max} for each segment, t_2,t_3,...,t_n, must be chosen. This choice will affect the curve smoothness.

Continuity of second derivatives at the interval joints does not in itself produce a smooth spline in the sense of minimum curvature along the curve. To obtain a minimum, and hence maximum smoothness, the coefficients B_2 and B_3 must be minimized for each segment by choosing the correct values of the parameter range within each segment.

One approach is to set the maximum parameter values equal to the chord lengths between successive data points. A second approach is to normalize the variation by choosing $t_{max} = 1.0$ for each cubic segment. Each choice of t_{max} will produce different coefficient values and, hence, different curves through the given data points. As the magnitude of the tangent vectors is changed, the slope of the cubic segments between data points is changed. On the other hand, the direction of the tangent vectors controls the shape of the cubic segments at their end points.

For simplicity we normalized parameter ranges for all cubic segments of a spline curve. This is because for this approach the additional computational effort of calculating the chord lengths between each successive data points is *not* required and the normalized formation is much easier in creating cubic boundary curves for three-dimensional surface patch which we will discuss later. When this normalized parameter approach is used, we have $0 \leq t \leq 1$ for all spans and equation 5.19 can be rearranged in the form:

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 \\ & \vdots & & \vdots & & & \vdots & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} P_2' \\ P_3' \\ P_4' \\ P_5' \\ \vdots \\ P_{n-1}' \end{bmatrix} = \begin{bmatrix} 3(P_3 - P_1) - P_1' \\ 3(P_4 - P_2) \\ 3(P_5 - P_3) \\ 3(P_6 - P_4) \\ \vdots \\ 3(P_n - P_{n-2}) - P_{n-2}' \end{bmatrix} \tag{5.26}$$

The four coefficients for each parametric cubic equation can be expressed in matrix form as :

$$\begin{bmatrix} B_3 \\ B_2 \\ B_1 \\ B_0 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_k \\ P_{k+1} \\ P_k' \\ P_{k+1}' \end{bmatrix} 1 \leq k \leq n-1 \qquad (5.27)$$

We compute the cubic spline representations of each space surve as follows. First each point in the set of range data is represented by the vector $[x(u,v), y(u,v), z(u,v)]$ on a surface. Thus we generate space curves from all the points with the same u values by cubic splines. All those space curves are parallel in u direction. Similarly, generate other groups of space curves which are parallel in v direction. We first assign all the points with the same u value u_i to a structured one-dimensional array. Each element of such array consists of the x, y and z values of each point. Thus we rearrange the array in decreasing order of its x value by *the quick sort algorithm.* We continue processing all the points in this manner until there is no other value u_i in the u domain. Since we need at least three points to form a cubic spline space curve, we skip the array whose element is less than three. After processing all the curves in u direction, we repeat the whole procedure but in the v direction.

5.2 Bicubic Surface Patch

There are various analytical and numerical techniques for representing a three-dimensional surface. Before proceeding with the details of the representation of curved three-dimensional surfaces, it is necessary to make several decisions. First we assume a surface representation based on a vector-valued parametric concept. There are several reason for this. This type of representation is axis-independent: it avoids infinite slope values with respect to some arbitrary axis system, it allows the unambiguous representation of multivalued surfaces or space functions, it facilitates the representation

of space curves in homogeneous coordinates, and it is compatible with the use of the three-dimensional homogeneous coordinate transformations. We assume the surface of human body can be representation in a piecewise fashion, i.e, a surface is composed of a number of patches which are joined together in some fashion at the boundaries. One of the most useful patch descriptions uses parametric cubics for the edge curves $P(u,0)$, $P(u,1)$ and $P(0,v)$, $P(1,v)$ and the blending functions. A patch means a curve bounded collection of points whose coordinates are given by continuous, two parameter, single valued mathematical functions of the form $x = x(u,v), \quad y = y(u,v), \quad z = z(u,v)$. The parametric variables u, v are constrained to the intervals $u, v \in [0,1]$.

Fixing the value of one of the parametric variables results in a curve on the patch in terms of the other variable which remains free. By continuing this process first for one variable and then the other for any number of arbitrary values in the allowed interval, we form a parametric net of two one-parameter families of curves on the patch so that just one curve of each family passes through each point $p(u,v)$. Again, the direction of any curve is the sense in which the free parameter increases. Associated with every patch is a set of boundary conditions, see Figure 5.1.

For an ordinary patch, there are always four and only four corner points and boundary curves. This follows from the possible combinations of the two limits of the two parametric variables. We obtain these by allowing one of the variables to remain free fixing the other to its limiting values. This procedure results in four and only four possible combinations yielding the functions of the four parametric boundary curves $P_{u,0}$, $P_{u,1}$, $P_{0,v}$, and $P_{1,v}$.

The algebraic form of a bicubic patch is given by

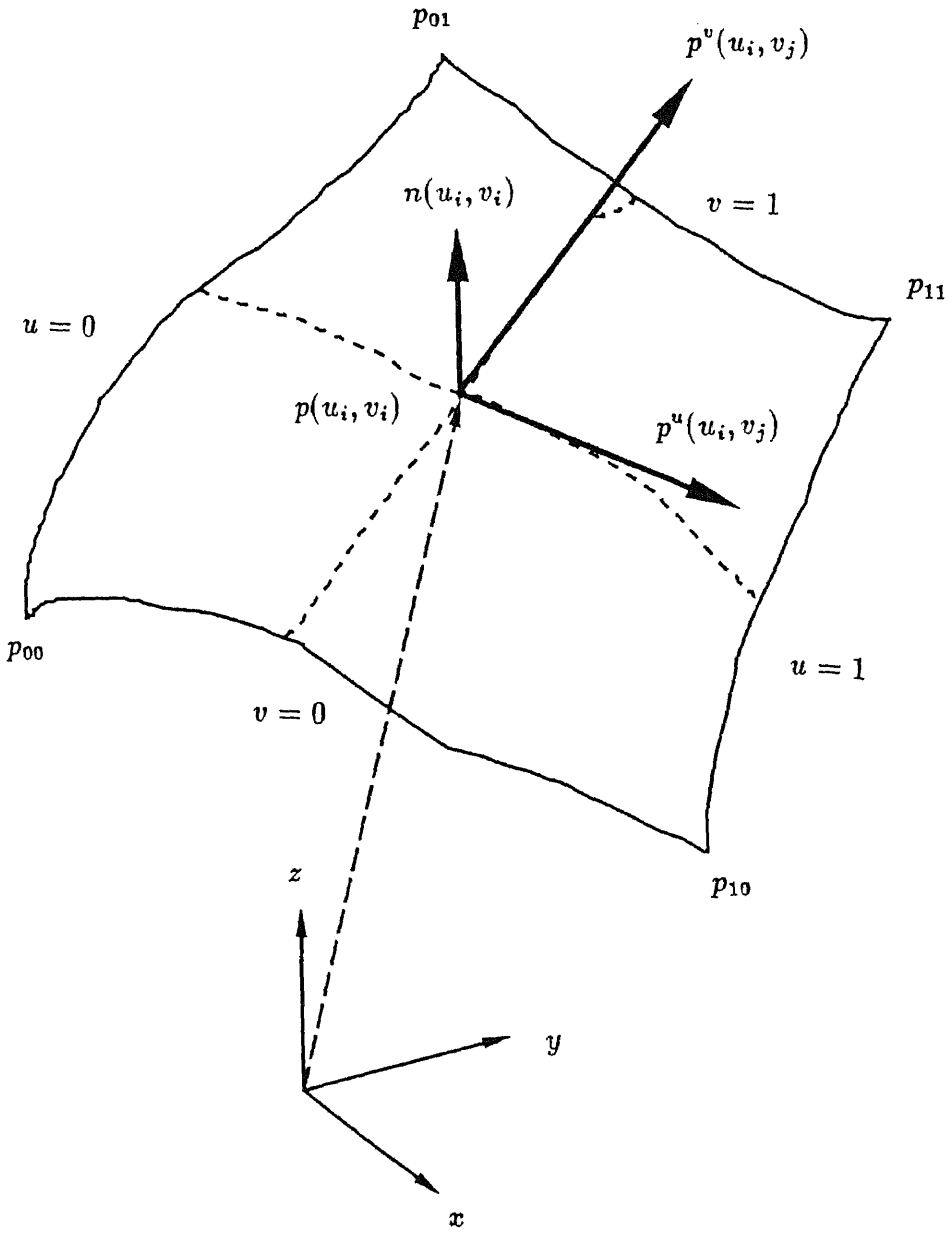

Figure 5.1: A parametric surface patch

$$P(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}u^i v^j \tag{5.28}$$

The restriction on the parametric variables is

$$u, v \in [0,1]$$

The a_{ij} vectors are called the algebraic coefficients of the surface. The reason for the term *bicubic* is obvious, since the polynomial $P(u,v)$ is cubic in u and v.

Expanding 5.28 and arranging the terms in descending order

$$\begin{array}{rcccccccc} P(u,v) & = & a_{33}u^3v^3 & + & a_{32}u^3v^2 & + & a_{31}u^3v & + & a_{30}u^3 \\ & + & a_{23}u^2v^3 & + & a_{22}u^2v^2 & + & a_{21}u^2v & + & a_{20}u^2 \\ & + & a_{13}uv^3 & + & a_{12}uv^2 & + & a_{11}uv & + & a_{10}u \\ & + & a_{03}v^3 & + & a_{02}v^2 & + & a_{01}v & + & a_{00} \end{array} \tag{5.29}$$

This 16-term polynomial in u and v defines the set of all points lying on the surface. It is the algebraic form of a bicubic patch. Since each of the vector coefficients a has three independent components, there is a total of 48 algebraic coefficients, or 48 degrees of freedom. Thus, each vector component is simply

$$x(u,v) = a_{33x}u^3v^3 + a_{32x}u^3v^2 + a_{31x}u^3v + \ldots + a_{00x} \tag{5.30}$$

There are similar expressions for $y(u,v)$ and $z(u,v)$.

The algebraic form in matrix notation is

$$P = UAV^T \tag{5.31}$$

where

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

and

$$A = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p(0,0) & p(0,1) & p_{01}(0,0) & p_{01}(0,1) \\ p(1,0) & p(1,1) & p_{01}(1,0) & p_{01}(1,1) \\ p_{10}(0,0) & p_{10}(0,1) & p_{11}(0,0) & p_{11}(0,1) \\ p_{10}(1,0) & p_{10}(1,1) & p_{11}(1,0) & p_{11}(1,1) \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

As we found with curves, the algebraic coefficients of a patch determine its shape and position in space. However, patches of the same size and shape have a different set of coefficients if they occupy different positions in space. Change any one of the 48 coefficients, and a completely different patch results.

Observe that a bicubic patch is bounded by four curves and each boundary curve is obviously a parametric curve. Each of the curves is named as P_{0v}, P_{1v}, P_{u0} and P_{u1} (see Figure 5.1). There are also four unique corner points p_{00}, p_{10}, p_{01} and p_{11}. The 16 coefficients for each of the four cubic boundary curves are four corner points, eight tangent vectors at the corners, and four twist vectors at the corners [55] [41].

The 4×4 p-matrix may be considered as a boundary condition matrix. It contains only information about corner points, corner tangent vectors, and corner twist vectors. Examining of the P-matrix shows that the term can be classified as

$$P = \begin{bmatrix} Corner & v-tangent \\ coordinates & vectors \\ \dots & \vdots \quad \dots \\ u-tangent & Twist \\ vectors & vectors \end{bmatrix}$$

And we can derive the coefficients of P-matrix from equations 5.7, 5.8, 5.9, and 5.10 for four boundary curves. In other words we begin by filling the first two rows with the geometric coefficients of curves P_{0v} and P_{1v}

Row1: $P_{0v} \rightarrow p_{00} \quad p_{01} \quad p^v_{00} \quad p^v_{01}$

Row2: $P_{1v} \rightarrow p_{10} \quad p_{11} \quad p^v_{10} \quad p^v_{11}$

Next, fill the first two columns with the geometric coefficients of the curves P_{u0} and P_{u1}.

Column1: $P_{u0} \rightarrow p_{00} \quad p_{10} \quad p^u_{00} \quad p^u_{10}$

Column2: $P_{u1} \rightarrow p_{01} \quad p_{11} \quad p^u_{01} \quad p^u_{11}$

Computing and understanding the significance of the parametric derivatives of the bicubic patch function is similar to these same processes for the parametric curve. The difference is that the bicubic function is expressed in two independent variables instead of just the one for curves. This means that we must deal with partial derivatives.

Now let us define the twist vectors as

$$P^{u_1 v_2}_{u_3 v_4} = \frac{\partial^2 p(u_1, v_2)}{\partial u_3 \partial v_4}$$

One interesting result is that a change to any of the scalar multiplies of the twist vectors does not change any of the four boundary curves. We can fix the four corner points and boundary curves and still affect the interior shape of a patch through operations on the twist vector. We can define a patch with all twist vectors equal to zero.The matrix of geometric coefficients then becomes

$$P = \begin{bmatrix} p_{00} & p_{01} & p^v_{00} & p^v_{01} \\ p_{10} & p_{11} & p^v_{10} & p^v_{11} \\ p^u_{00} & p^u_{01} & 0 & 0 \\ p^u_{10} & p^u_{11} & 0 & 0 \end{bmatrix} \tag{5.32}$$

This type of patch is called an F-patch. Only first-order, C^1, continuity is possible across the boundaries of adjacent F-patches because these patches are constrained to

have zero cross-derivatives at their corners. In particular, F-patches have proven to be useful for axisymmetric surfaces , such as vases, cups, bottles, aircraft bodies, etc. And in our system, we use the F-patch concepts to deal with surface representation.

By now we know that we need four and only four points and boundary curves to form a single surface patch. Unfortunately, if we look at the intersection graph carefully, the intersection points that we detected are not evenly spread on the objects (human body). This is due to the fact that the projection of the pattern is not uniform on the surface of the body, causing us to have to do extra work to find the four boundary curves and points. Otherwise, the patches on the surface will be irregular. Timmer [41] suggests us to use the following strategy to avoid problems with meshes that are not rectangular.

> Let n_0 denote the number of data points required to define the most complex cross-section curve and m the number of cross-sections. A rectangular mesh requires $m \times n$ mesh points; however, instead of a rectangular mesh format, we now allow the model input to consist of the exact number of points necessary for each cross section individually. Thus, a linear cross-section curve could be described using only two points. Using this approach, the total number of points required to reduce the surface is
>
> $$N = \sum_{i=1}^{m} n_i \tag{5.33}$$
>
> Obviously, $N \leq mn_0$. Using this reduced input data, we interpolate each cross-section curve, using a conventional spline or other form of representation. Next, we compute a set of n_0 points for each curves, and we continue as outlined for a rectangular mesh.

In his method, the patch will be a little bit larger than the original patch. Because we are are trying to get the surface area of human body, Timmer's technique will cause the size of a single surface patch to be larger and there will be fewer of them. This may effect the accuracy of calculating the surface area. In order to avoid reducing the data set, we use another approach.

First we store each space point into a structured two-dimensional array indexed by its uv-coordinate value, say $p(u_i, v_i)$. Each element of the array consists of its 3D location values and the u and v values are increased by 1 in the array. Due to the irregular spacing of grid points not every element in the array will have 3D values. We distinguish these invalid points by assigning it a vary large positive number which cannot be generated in our system. If the three neighboring points of each point in the array are all valid (say $p(i,j)$, $p(i+1,j)$, $p(i,j+1)$ and $p(i+1,j+1)$ are valid points), then those four point can form a smallest surface patch. Otherwise, if some of its successive points are invalid then we examine another three successive points such as $p(i+2,j)$, $p(i,j+2)$ and $p(i+2,j+2)$. If they are all valid then we choose these four points as corner points to form a surface patch. Using this surface patch,we can interpolate the missing points. If this interpolation still fails, we give up to do further trial. This is because we only miss some points around the marker on projector slide and where u or v values vary by 2. For the other part of the image, our system rarely misses detecting two successive intersection points. After we examine all the space points, there are some new points generated by surface interpolation. Since the cubic spline coefficients are affected by all points, we have to recompute the parameters of the space curves for those artificial points using the cubic spline estimation procedure.

The side effect of this procedure is it takes more computation time. But after this

procedure, we have a surface which is composed of the smallest possible surface patches and we lose no real points.

The following is the basic surface patch estimation procedure without surface interpolation. The procedure is:

1. Visit every intersection point in u and v order

2. If $p(i,j)$ exists and is not the end point for either a u or v path then

3. Check $p(i+1,j)$ and $p(i,j+1)$. If they both exist then

   A Check whether $p(i+1,j+1)$ exists or not. If it exists, then choose these four points $p(i,j), p(i+1,j), p(i,j+1)$ and $p(i+1,j+1)$ as corner points to compute the surface patch.

   Then go to visit next point and start from step 2 again.

   B If $p(i+1,j+1)$ does not exist then check $p(i+2,j), p(i+2,j+1), p(i,j+2), p(i+1,j+2)$ and $p(i+2,j+2)$. If they all exist, then treat $p(i,j), p(i+2,j), p(i,j+2)$ and $p(i+2,j+2)$ as corner points and compute the surface patch. If one of them does not exist, then go to the next point and start from step 2.

4. If one of the points $p(i+1,j)$ and $p(i,j+1)$ does not exist, then go to the next point and start from step 2.

5. If $p(i,j)$ does not exist, then go to the next point and start from step 2.

6. Continue processing until every point in the space has been visited.

Chapter 6

Estimating Surface Area

After we reconstruct the surface of human body using bicubic surface patches, the geometric properties of the surface of human body can be estimated. We can estimate the intrinsic properties such as the normal vectors, the tangent plane, and curvature. We can also estimate global properties such as area because the bicubic surface patches are well adapted to computing geometric properties, since we do not have to compute explicit points. To compute surface area, we use an elementary property of vectors shown in Figure 6.1 and given by

$$\begin{aligned} dA &= |n|dudv \\ &= f_1(u,v)dudv \end{aligned} \tag{6.1}$$

where $n(u,v)$ is a vector function defining the patch normal and dA is the scalar element of area. Therefore,

$$A = \int_0^1 \int_0^1 f_1(u,v)dudv \tag{6.2}$$

and using Gaussian quadrature, we approximate the double integral by

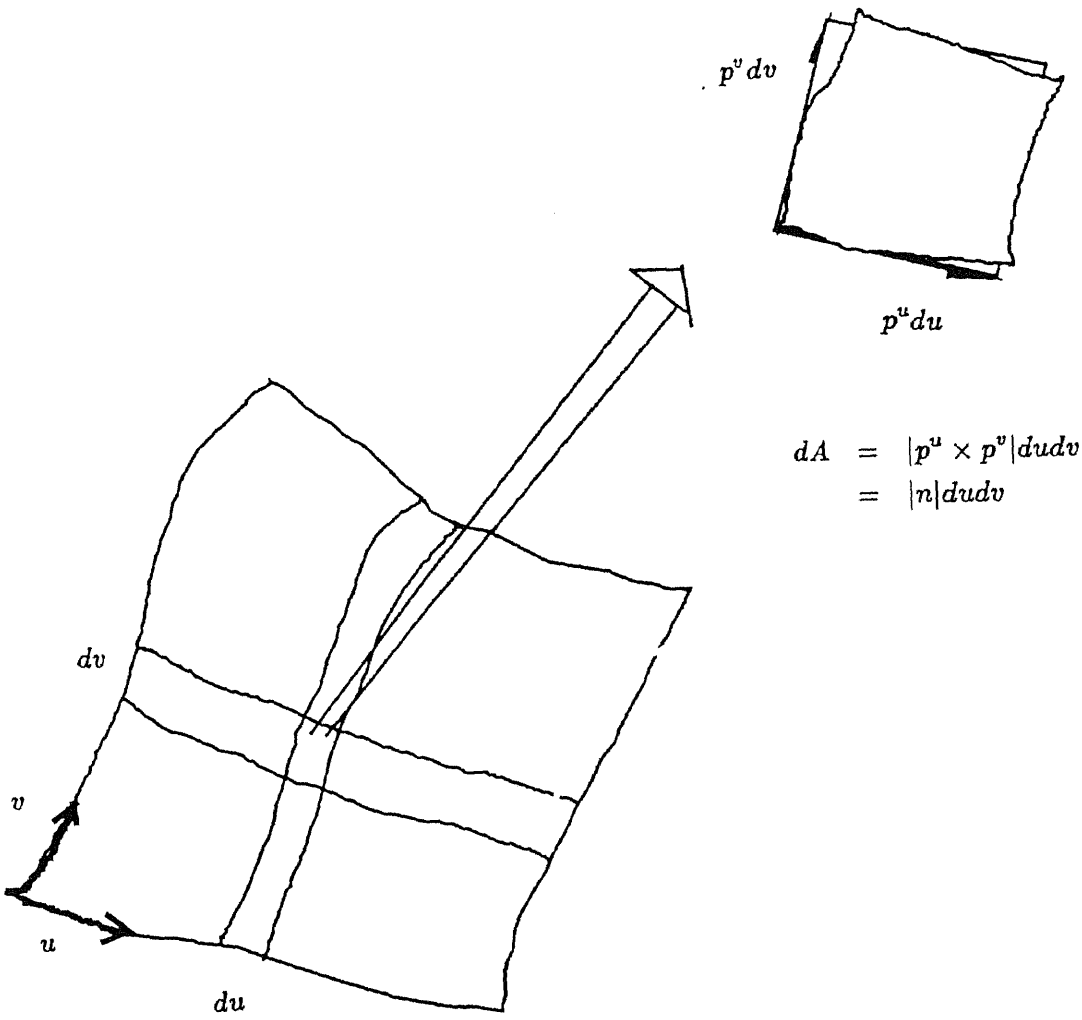

Figure 6.1: Surface Area

$$A \approx \sum_{i=0}^{n} \sum_{j=0}^{n} g_i h_j f_1(u_j, v_i) \tag{6.3}$$

where g_i and h_j are the weight values associated with a specific n-point formula.

From Equation 6.3, we can compute the surface area for a single surface patch. The total surface area is given by the quadrature sum over all of the areas of the individual surface patches of the human body captured by our imaging system.

Our immediate goal is to compute the surface area of the burn. Using the segmented image of the burn patient which identifies the burn wound portion that that we are actually interested in, we can superimpose the burn area boundary and the detected intersection points by projecting the points to the camera image and find out which points and which surface patches lie inside the burn area. Then we calculate the surface area of the patches using equation 6.3. Unfortunately, the shape of burn is not always rectangular. Thus, for the contour of the burn, we need to form a new surface patch using its original four corner points and the shape of burn in that small patch. There arise 12 cases according to how many corner points of that patch lie on the burn wound portion. So in this chapter we will discuss

- Forming the normal to the patch.
- Gaussian Quadrature.
- Reforming the surface patch for irregular shapes.

6.1 Forming the normal to the patch

At any point $p(u, v)$ on a bicubic patch, we can construct a normal vector perpendicular to the patch. We easily find a unit normal vector $n(u, v)$ by computing the vector product of the tangent vectors p^u and p^v at the point.

$$n(u,v) = \frac{p^u \times p^v}{|p^u \times p^v|} \tag{6.4}$$

The order in which we take the vector product determines the direction of $n(u,v)$. Notice that we can interpret $n(u,v)$ itself as a patch, the **normals patch.**

Figure 6.2 shows a convention for assigning identifying numbers to the corner points and boundary curves. Here, we see if the fingers of the right hand curl around the patch in the direction of ascending curve or corner-point numbering, then the thumb points in the direction of the positive or outward surface normal as defined by equation 6.3. This convention gives a consistent algebraic sign when summing the area of several surface patches.

Normally, we want the normal to point outward from the surface of a solid model. Thus, the unit normal is indispensible in almost all phases of geometric modeling, and the normal direction is required.

Equation 6.4 for the normal vector can be expressed in a more convenient form, which will take advantage of existing algorithms for the bicubic surface patch. We rewrite this equation as

$$\begin{aligned} k_n n &= [x^u \; y^u \; z^u] \times [x^v \; y^v \; z^v] \\ &= [(y^u z^v - y^v z^u) \quad (z^u x^v - z^v x^u) \quad (x^u y^v - x^v y^u)] \end{aligned} \tag{6.5}$$

where $k_n = |p^u \times p^v|$

The x component of n is

$$n_x = \frac{y^u z^v - y^v z^u}{k_n} \tag{6.6}$$

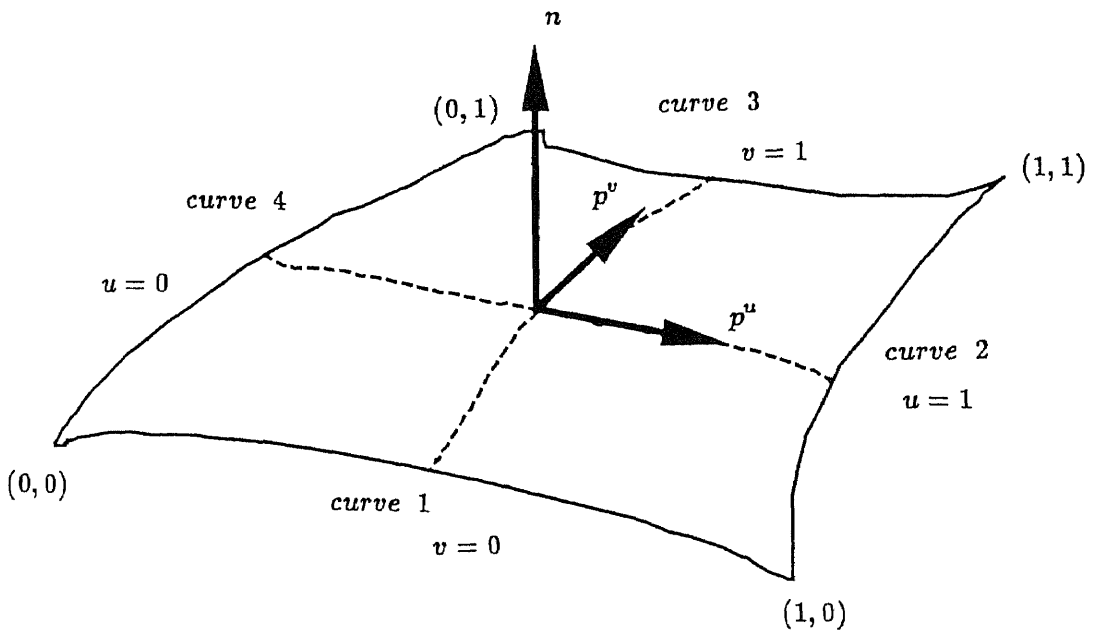

Figure 6.2: Normal vector to a surface

By the bicubic patch we can form the tangent vectors and the twist vectors as

$$p^u = UM^uBM^TV^T \tag{6.7}$$

$$p^v = UMBM^{vT}V^T \tag{6.8}$$

$$p^{uv} = UM^uBM^{vT}V^T \tag{6.9}$$

where

$$M^u = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M^{vT} = \begin{bmatrix} 0 & 6 & -6 & 0 \\ 0 & -6 & 6 & 0 \\ 0 & 3 & -4 & 1 \\ 0 & 3 & -2 & 0 \end{bmatrix}$$

$$M = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$M^T = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

We can rewrite each of the terms y^u, z^u, y^v , and z^v in their matrix form.

$$n_x = \frac{1}{k_n}[(UM^uB_yM^TV^T)(UMB_zM^{vT}V^T) - (UMB_yM^{vT}V^T)(UM^uB_zM^TV^T)] \tag{6.10}$$

and similarly for n_y and n_z

$$n_y = \frac{1}{k_n}[(UM^u B_z M^T V^T)(UMB_x M^{vT} V^T) - (UMB_z M^{vT} V^T)(UM^u B_x M^T V^T)] \tag{6.11}$$

$$n_z = \frac{1}{k_n}[(UM^u B_x M^T V^T)(UMB_y M^{vT} V^T) - (UMB_x M^{vT} V^T)(UM^u B_y M^T V^T)] \tag{6.12}$$

We can then approximate the normal at any point on the surface with a bicubic expression, so that n is

$$n = UMB_n M^T V^T \tag{6.13}$$

where

$$\begin{aligned} B_{nx} &= \begin{bmatrix} n_x & n_x^v \\ n_x^u & n_x^{uv} \end{bmatrix} \\ B_{ny} &= \begin{bmatrix} n_y & n_y^v \\ n_y^u & n_y^{uv} \end{bmatrix} \\ B_{nz} &= \begin{bmatrix} n_z & n_z^v \\ n_z^u & n_z^{uv} \end{bmatrix} \end{aligned}$$

are the components of the 4×4 matrix B_n.

We have already derived n_x,n_y, and n_z. We find the three remaining components of the B_{nx} by differentiating n_x

$$\begin{aligned} n_x^u &= \frac{1}{k_n}[(UM^{uu} B_y M^T V^T)(UMB^z M^{vT} V^T) \\ &\quad +(UM^u B_y M^T V^T)(UM^u B_z M^{vT} V^T) \end{aligned}$$

$$
\begin{aligned}
& -(UM^{u}B_{y}M^{vT}V^{T})(UM^{u}B_{z}M^{T}V^{T}) \\
& -(UMB_{y}M^{vT}V^{T})(UM^{uu}B_{z}M^{T}V^{T})] \\
n_{x}^{v} \;=\; & \frac{1}{k_{n}}[(UM^{u}B_{y}M^{vT}V^{T})(UMB_{z}M^{vT}V^{T}) \\
& +(UM^{u}B_{y}M^{T}V^{T})(UMB_{z}M^{vvT}V^{T}) \\
& -(UMB_{y}M^{vvT}V^{T})(UM^{u}B_{z}M^{T}V^{T}) \\
& -(UMB_{y}M^{vT}V^{T})(UM^{u}B_{z}M^{vT}V^{T})] \\
n_{x}^{uv} \;=\; & \frac{1}{k_{n}}[(UM^{uu}B_{y}M^{vT}V^{T})(UMB_{z}M^{vT}V^{T}) \\
& +(UM^{uu}B_{y}M^{T}V^{T})(UMB_{z}M^{vvT}V^{T}) \\
& +(UM^{u}B_{y}M^{vT}V^{T})(UM^{u}B_{z}M^{vT}V^{T}) \\
& +(UM^{u}B_{y}M^{T}V^{T})(UM^{u}B_{z}M^{vvT}V^{T}) \\
& -(UM^{u}B_{y}M^{vvT}V^{T})(UM^{u}B_{z}M^{T}V^{T}) \\
& -(UM^{u}B_{y}M^{vT}V^{T})(UM^{u}B_{z}M^{vT}V^{T}) \\
& -(UMB_{y}M^{vvT}V^{T})(UM^{uu}B_{z}M^{T}V^{T}) \\
& -(UMB_{y}M^{vT}V^{T})(UM^{uu}B_{z}M^{vT}V^{T})] \qquad (6.14)
\end{aligned}
$$

where

$$
M^{uu} \;=\; \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 12 & -12 & 6 & 6 \\ -6 & 6 & -4 & -2 \end{bmatrix}
$$

$$
M^{vvT} \;=\; \begin{bmatrix} 0 & 0 & 12 & -6 \\ 0 & 0 & -12 & 6 \\ 0 & 0 & 6 & -4 \\ 0 & 0 & 6 & -2 \end{bmatrix}
$$

We can determine the 16 elements of B_{nx} and similarly B_{ny} and B_{nz}, by evaluating each of these expressions at the four corner points $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ of the

surface patch.

6.2 Gaussian Quadrature

Because of the limitations of digital computers, there is no way to calculate the integral of equation 6.2 directly, forcing us to resort to numerical integration. The numerical method that we chose for the integration is called *Gaussian Quadrature* which gives us the freedom to choose not only the weighting coefficients, but also the location of the abscissae at which the function is to be evaluated. The abscissae do not have to be equally spaced. It turns out that we can derive a Gaussian quadrature formula whose order is, essentially, twice that of the Newton-Cotes formula with the same number of function evaluations. We are not saying that high order is the same as high accuracy. High order translates to high accuracy only when the integrand is very smooth, in the sense of being "well-approximated by a polynomial." But, indeed, there is one additional feature of Gaussian quadrature formulae allowing us to arrange the choice of weights and abscissae to make the integral exact for a class of integrands.

The source code that we used is derived from the book "Numerical Recipes in C" [53]. This source code has three departures from the general methods for arbitrary weight functions.

1. Instead of bracketing the roots by their *interleaving* property, it uses a clever approximation to jump directly to the neighborhood of the desired root, where it converges by Newton's method.

2. It uses a special formula that holds for the Gauss-Legendre case

$$w_i = \frac{2}{(1 - x_i^2)[p_N'(x_i)]^2} \tag{6.15}$$

where $p_N(x)$ is a set of polynomials that are mutually orthogonal over the specified weight function $w(x)$.

3. The routine scales the range of integration from (x_1, x_2) to $(-1, 1)$ and provides abscissae x_i and weights w_i for the Gaussian formula. It is called the **Gauss-Legendre n-point quadrature formula**

$$\int_{x_1}^{x_2} f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i) \tag{6.16}$$

The program returns the abscissae and weights of the Gauss-Legendre n-point quadrature formula in two arrays and we set the lower and upper limits (x_1, x_2) of integration to be 0 and 1 to fix the interval of bicubic surface patch.

Because the double summation in equation 6.3 is separable, we can treat g_i, v_i and h_j,u_j as two independent groups and use the procedure twice to obtain the values of g_i and v_i and the values of h_j and u_j, separately.

6.3 Reforming the Surface Patch for Irregular Shapes

Due to the irregular shape of the burn area, we need to do extra work to correct those surface patch whose four corner points do not all lie inside the burn area. First, we determine how many corner points lie inside the burn area. If there are four, then we can use equation 6.3 to directly calculate the surface area. If there are no corners in the burn area then we discard the patch. For the remaining cases, there are four instances to consider.

The three remaining situations are shown in Figure 6.3, 6.4, and 6.5 which are either one, two, or three corner points inside the burn area. If one of these cases occurs, we need to reform four boundary curves for such a patch. For each individual patch

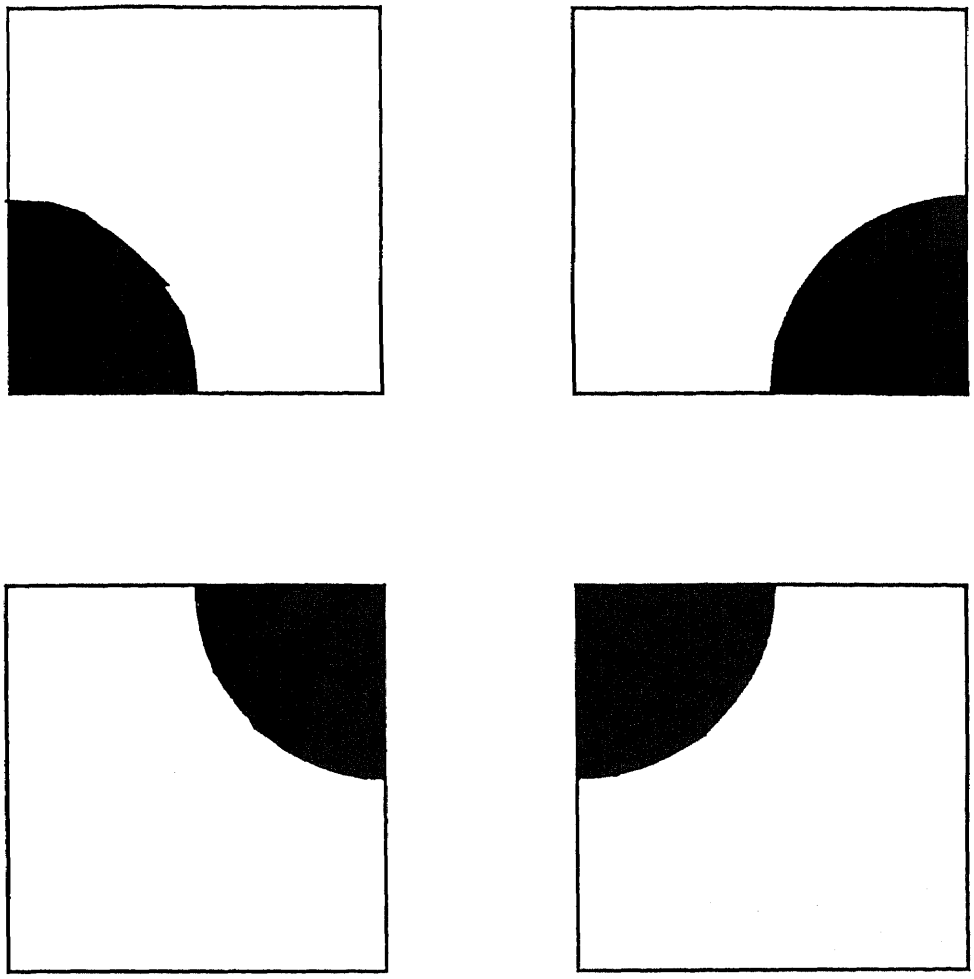

Figure 6.3: Case 1. One out of four points lie on the burn area

there will be one, and only one, of the cases shown in Figures 6.3, 6.4 or 6.5 if it has one to three corner points inside the burn area. Here we assume that the boundary of the burn area is smooth, unlike that in Figure 6.6 which we will discuss later.

In order to reform four boundary curves for those unperfect cases, we need to know the exact positions of all those points which really lie on the edge of the burn in that surface patch. Then, we reorder those points and reform the boundary curves. The simplest way to find those points is by first using a bicubic surface patch. Because we already know the geometric coefficients for a bicubic patch, we can determine the

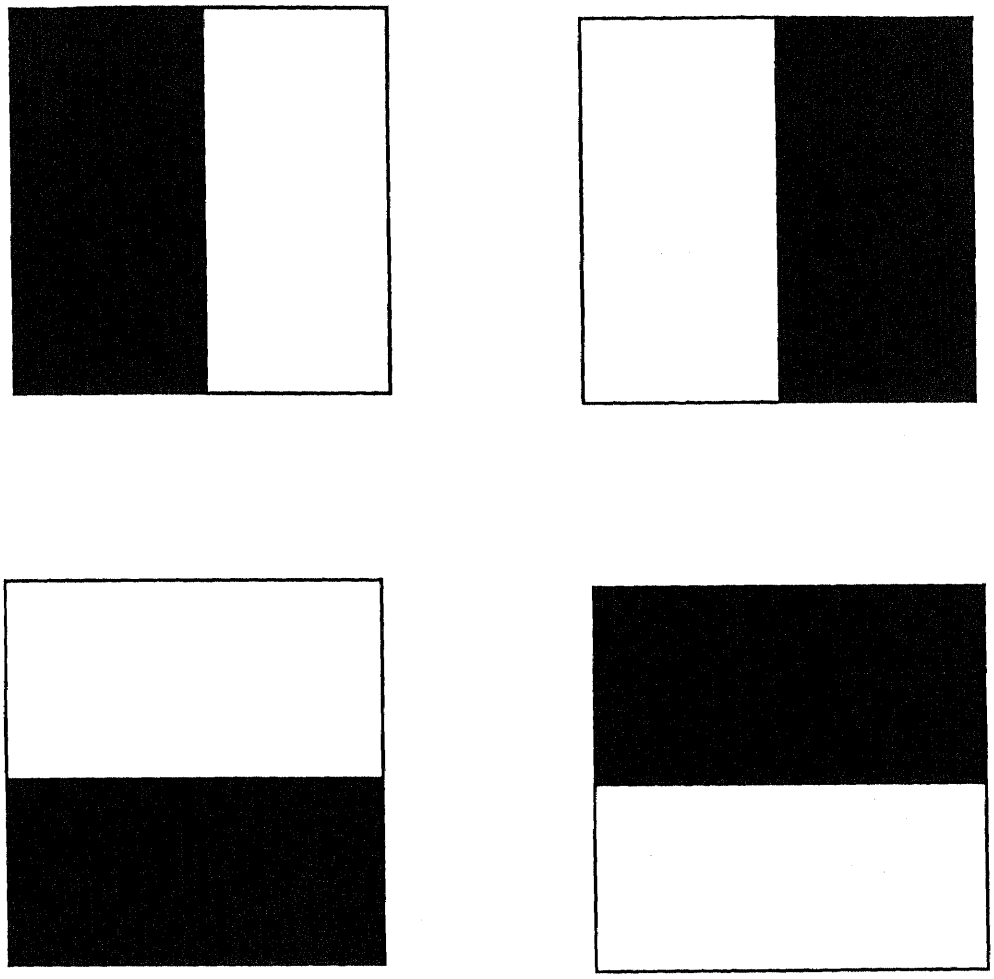

Figure 6.4: Case 2. Two out of four points lie on the burn area

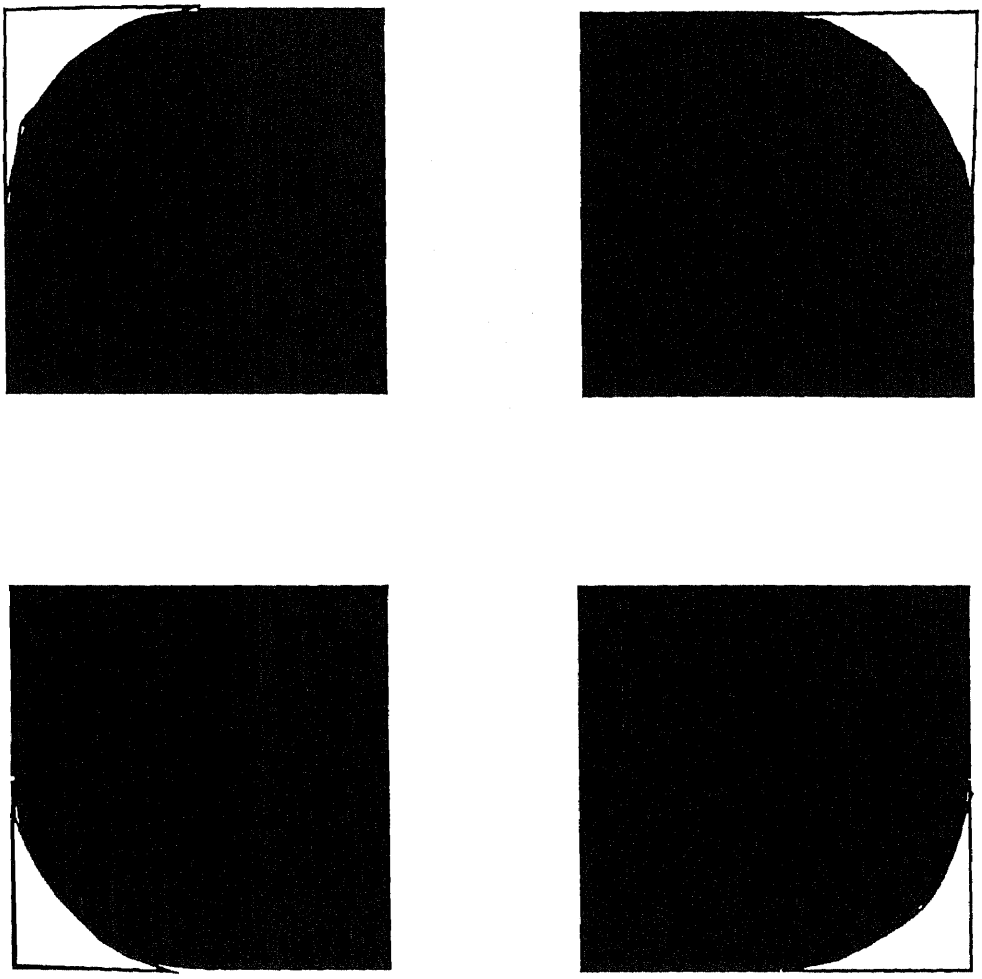

Figure 6.5: Case 3. Three out of four points lie on the burn area

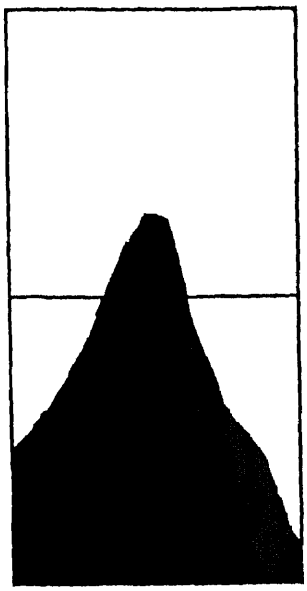

Figure 6.6: Special case

coordinates of a point $p(u, v)$ on a bicubic surface by using the formula

$$p(u, v) = UMBM^T V^T \tag{6.17}$$

which is derived from solving the intersection of the curves p_{iu} and p_{uj} in Figure 6.7.

We neglect the derivation of equation 6.17 here, because we are only interested in how to use this equation. We first fix one variable and allow the other variable to remain free varying from 0 to 1. In this way we can trace a curve on the patch. We record only those points on the bounday of the burn area by checking its state in the segmented image and comparing its state to the state of its previous point and its successor. Then we increase the value of the fixed variable by a certain amount to trace another point on the curve. We record these edge points until the fixed variable reaches its limit. Due to the fact that the burn area will hit the patch at least twice, we continue this process until the fixed variable reaches its upper limit 1 and there are at

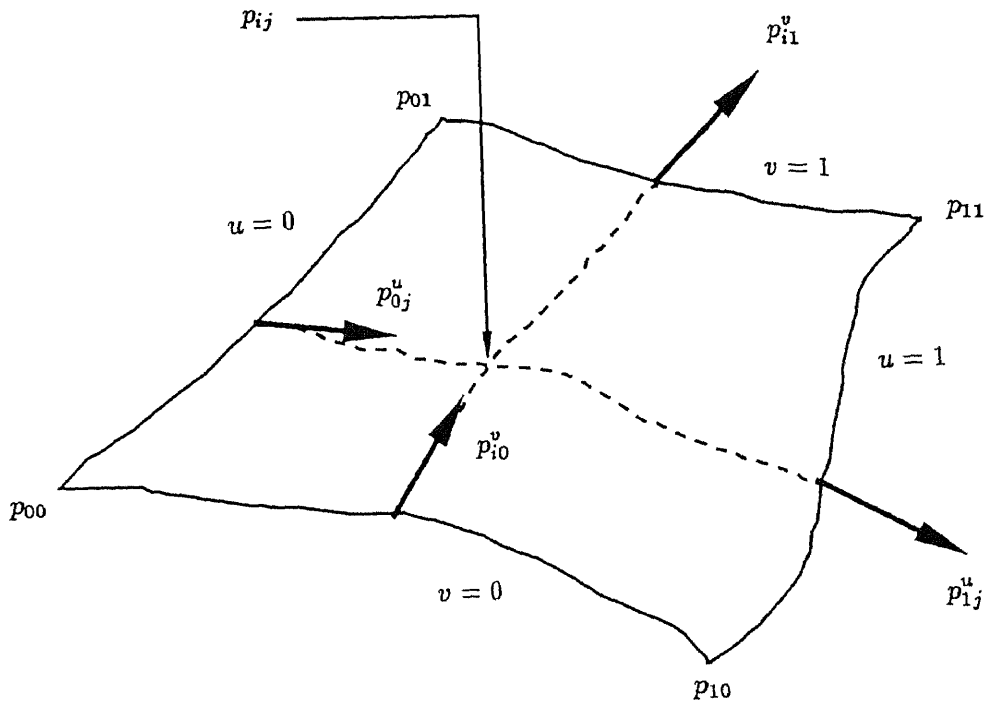

Figure 6.7: Determining a parametric point on bicubic surface

least two intersection points between the burn and the boundary curves of such patch. Otherwise, we decrease the sampling interval and redo the whole procedure all over again. After obtaining the points on the edge of burn on a bicubic surface, we have to rearrange those points to form a space curve. Because there are many ways to rearrange those points and the results will affect the simplicity of generating four new boundary curves and four new corner points, we need to consider how many original corner points lie in the burn area, and their relative position with respect to the burn area will dictate how to proceed. We first determine which point has the maximum or minimum values of u or v values among those points and assign this point as a seed point. Then, the points on the boundary of the burn area are sorted by finding the nearest neighbor of each point. We need to use the concepts of truncating and subdividing of a space curve to regenerate the four boundary curves.

Figure 6.8 illustrates a curve truncated at u_i and u_j, that is, the segments from u_0 to u_i and from u_j to u_1 are eliminated. We can represent the remaining segment as a full parametric space curve, parametrized from $v = 0$ to $v = 1$, by proceeding as follows: compute p_i and p_j using $p = UMB$ and p_i^u and p_j^u using $p^u = UM^uB$. The ratio of parametric interval lengths $(u_j - u_i)/(v_j - v_i)$ becomes $u_j - u_i$, since $v_j - v_i = 1$. If q represents elements of a truncated curve and p the elements of the original, then

$$\begin{aligned} q_0 &= p_i \\ q_1 &= p_j \\ q_0^u &= (u_j - u_i)p_i^u \\ q_1^u &= (u_j - u_i)p_j^u \end{aligned} \tag{6.18}$$

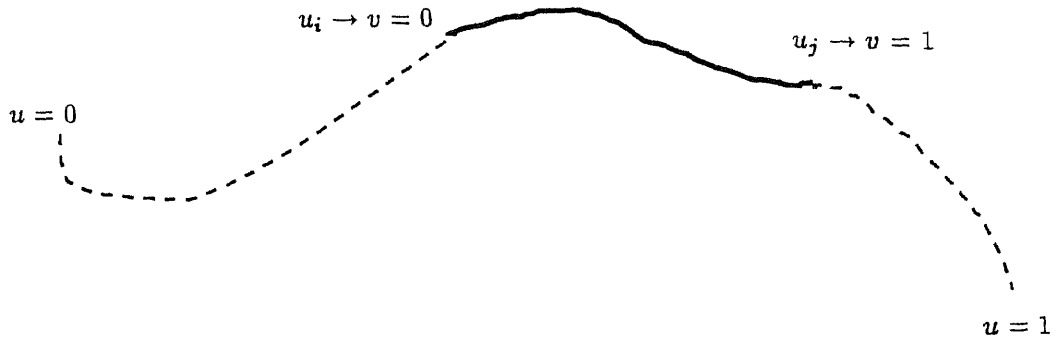

Figure 6.8: Reparametrization of a truncated curve

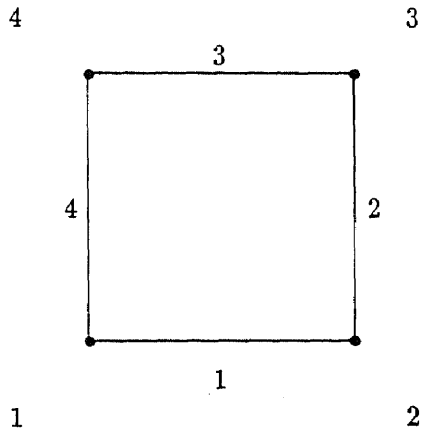

Figure 6.9: The definition of boundary curves and corner points

We define the four boundary curves as shown in Figure 6.9. The procedure of reforming the four new boundary curves based on the observed curve is as follows:

1. One out of four corner points is on the burn area. The four subcases are shown in Figure 6.3.

   - If the upper-left happens:

     A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

     B. Truncate the original boundary curve 1 to become the new boundary curve 1.

     C. By using cubic splines to form the new boundary 2 with the points starting from the first point to the middle point of the ordered edge point set.

     D. By using cubic splines to form the new boundary 3 with the points starting from the last point to the middle point of the ordered edge point set.

     E. Truncate the original boundary curve 4 to become the new boundary curve 4.

     F. The first new corner point remain the same, the second corner point is the first point in the ordered set. the third corner point is the middle point in the ordered set. the forth corner point is the last point in the ordered set.

   - If the upper-right happens:

A. Find the minimum v value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. Truncate the original boundary curve 1 to become the new boundary curve 1.

C. Truncate the original boundary curve 2 to become the new boundary curve 2.

D. By using cubic splines to form the new boundary 3 with the points starting from the middle point to the last point of the ordered edge point set.

E. By using cubic splines to form the new boundary 4 with the points starting from the first point to the middle point of the ordered edge point set.

F. The first corner point is the first point in the ordered set, the second corner point remain the same, the third corner point is the last point in the ordered set, the forth corner point is the middle point in the ordered set.

- If the lower-right happens:

A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. By using cubic splines to form the new boundary 1 with the points starting from the first point to the middle point of the ordered edge point set.

C. By using cubic splines to form the new boundary 2 with the points starting from the middle point to the last point of the ordered edge point

set.

D. Truncate the original boundary curve 3 to become the new boundary curve 3.

E. Truncate the original boundary curve 4 to become the new boundary curve 4.

F. The first corner point is the first point in the ordered set, the second corner point is the middle point in the ordered set, the third corner point is the last point in the ordered set, the forth corner point remain the same.

- If the lower-left happens:

A. Find the maximum v value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. By using cubic splines to form the new boundary 1 with the points starting from the middle point to the last point of the ordered edge point set.

C. Truncate the original boundary curve 2 to become the new boundary curve 2.

D. Truncate the original boundary curve 3 to become the new boundary curve 3.

E. By using cubic splines to form the new boundary 4 with the points starting from the first point to the middle point of the ordered edge point set.

F. The first corner point is the middle point in the ordered set, the second corner point is the last point in the ordered set, the third corner point

remain the same, the forth corner point is the first point in the ordered set.

2. Two out of four corner points are on the burn area. The four subcases are shown in Figure 6.4.

   - If the upper-left happens:

     A. Find the minimum v value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

     B. Truncate the original boundary curve 1 to become the new boundary curve 1.

     C. Use the whole edge points to form a new boundary curve 2 by using the cubic spline method.

     D. Truncate the original boundary curve 3 to become the new boundary curve 3.

     E. Keep the boundary curve 4 unchanged.

     F. The first corner point and the second corner point uncharge, the third corner point become the last point in the set, the forth corner point become the first point in the set.

   - If the upper-right happens:

     A. Find the minimum v value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

     B. Truncate the original boundary curve 1 to become the new boundary curve 1.

     C. Keep the boundary curve 2 unchanged.

D. Truncate the original boundary curve 3 to become the new boundary curve 3.

E. Use the whole edge points to form a new boundary curve 4 by using the cubic spline method.

F. The second corner point and the third corner point uncharge, the forth corner point become the last point in the set, the first corner point become the first point in the set.

- If the lower-right happens:

  A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

  B. Use the whole edge points to form a new boundary curve 1 by using the cubic spline method.

  C. Truncate the original boundary curve 2 to become the new boundary curve 2.

  D. Keep the boundary curve 3 unchanged.

  E. Truncate the original boundary curve 4 to become the new boundary curve 4.

  F. The third corner point and the forth corner point uncharge, the second corner point become the last point in the set, the first corner point become the first point in the set.

- If the lower-left happens:

  A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

  B. Keep the boundary curve 1 unchanged.

C. Truncate the original boundary curve 2 to become the new boundary curve 2.

D. Use the whole edge points to form a new boundary curve 3 by using the cubic spline method.

E. Truncate the original boundary curve 4 to become the new boundary curve 4.

F. The first corner point and the second corner point uncharge, the third corner point become the last point in the set, the forth corner point become the first point in the set.

3. Three out of four corner points are on the burn area. The four subcases are shown in Figure 6.5.

   - If the upper-left happens:

     A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

     B. Keep the boundary curve 1 unchanged.

     C. Keep the boundary curve 2 unchanged.

     D. Use cubic spline method to generate a space curve with the point beginning with the middle point to the last point of the set and plus the points on the truncated space curve from the original boundary curve 3 to form new boundary curve 3.

     E. Use cubic spline method to generate a space curve with the points on the truncated space curve from the original boundary 4 and the first point to the middle point of the edge point set to become new boundary curve

4.

F. The first, second, and third corner point remain the same, the forth corner point become the middle point in the set.

- If the upper-right happens:

A. Find the maximum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. Keep the boundary curve 1 unchanged.

C. Use cubic spline method to generate a space curve with the points on the truncated space curve from the original boundary 2 and the first point to the middle point of the edge point set to become new boundary curve 2.

D. Use cubic spline method to generate a space curve with the point beginning with the middle point to the last point of the set and plus the points on the truncated space curve from the original boundary curve 3 to form new boundary curve 3.

E. Keep the boundary curve 4 unchanged.

F. The first, second, and forth corner point remain the same, the third corner point become the middle point in the set.

- If the lower-right happens:

A. Find the minimum u value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. Use cubic spline method to generate a space curve with the points on the truncated space curve from the original boundary 1 and the first point

to the middle point of the edge point set to become new boundary curve 1.

C. Use cubic spline method to generate a space curve with the point beginning with the middle point to the last point of the set and plus the points on the truncated space curve from the original boundary curve 2 to form new boundary curve 2.

D. Keep the boundary curve 3 unchanged.

E. Keep the boundary curve 4 unchanged.

F. The first, third, and forth corner point remain the same, the second corner point become the middle point in the set.

- If the lower-left happens:

A. Find the minimum v value among those edge points by the nearest neighborhood finding algorithm to reorder the edge point set.

B. Use cubic spline method to generate a space curve with the point beginning with the middle point to the last point of the set and plus the points on the truncated space curve from the original boundary curve 1 to form new boundary curve 1.

C. Keep the boundary curve 2 unchanged.

D. Keep the boundary curve 3 unchanged.

E. Use cubic spline method to generate a space curve with the point beginning with the middle point to the first point of The set and plus the points on the truncated space curve from the original boundary curve 4 to form new boundary curve 4.

F. The second, third, and forth corner point remain the same, the first

corner point become the middle point in the set.

During the procedure above, we sometimes need to combine two space curves to become one. Theoretically, there are three different types of composite curves, such as C^0, C^1, and C^2 continuity. The simplest kind of continuity a curve can has ensures that there are no gaps or breaks between its beginning and ending point. This is called C^0 continuity. Two parametric space curves jointed at a common end point have at least C^0 continuity at their joint. C^1 continuity between two curve segments requires a common tangent line at their joining point. Obviously, C^1 continuity also implies C^0 continuity. That is C^0 and C^1 continuity imply sharing a common point and tangent line at the junction of two curves. C^2 continuity also requires that the two curves possess equal curvature at their joint. Therefore, second derivatives are used in the calculation. So in order to get a composite curve which has at least C^1 continuity, we use the cubic spline again to generate a new spline curve with all the points on those two segments. This way, we can save the computation time of calculating the second derivative of two space curves and the cubic spline method will give us a space curve with at least C^1 continuity.

As long as we can successfully reform the four new boundary curves and four new corner points, we can use the bicubic surface patch again to obtain the geometric coefficients of the bicubic surface, and calculate the burn wound surface area by equation 6.3. For the special case shown in the Figure 6.6, we can still handle the lower grid because it will fall into one of the cases mentioned before. The area in the upper grid will be difficult because no corner points fall into the burn area even though it contains some parts of the burn. In our system we neglect this case for two reasons. First, the grid size of our projector slide is quite small, and the surface area of the burn in

the upper grid of the special case is negligible. Second, if we really want to calculate such a small area we will have to examine if there is any burn in every single grid. It is a computationally expensive task to determine the coordinates of every points on the bicubic surface and back projecting them onto the segmented image. Rather, we sacrifice this negligible error in the interest of efficient computation.

Chapter 7

Results and Conclusions

7.1 Automatic Calibration

In Chapter 2, we assume our imaging system is well behaved. That means the lenses of the camera and projector behaves as an ideal thin lens. The purpose of this section is to test our automatic calibration procedure and to determine whether the imaging system of ours has ideal thin lenses or not.

Table 7.1 shows the mean and rms errors in computing the objected centered coordinates of intersections on the camera and projector calibration images, respectively. These errors were computed by back projecting points in the image plane onto the calibration box face and back onto the image plane. Those errors are quit small compared to the size of the image (6 by 6 inch) that we captured. Figure 7.1 contains four pictures of the detected grid pattern, back projected intersections, difference of the two, and the sum, clockwise from upper left, showing that the error due to radial distortion is uniformly bounded by one pixel over the image plane. So the lens distortion and camera/light source misalignment do not affect our imaging system seriously. We conclude that the thin lens model is sufficient for our imaging system, although many investigators have found that ignoring lens distortion is unacceptable for 3D measurement

		mean error (inches)	rms error (inches)
camera	x	-0.000002	0.012390
	y	-0.000001	0.008330
projector	x	0.000059	0.019432
	y	0.000032	0.013132
	z	0.000028	0.014351

Table 7.1: The mean and rms error of calibration procedure

[62].

If we have a better designed and structured calibration box which can be positioned accurately and has a face that is exactly perpendicular to the base line, then the error will be reduced to its limit.

7.2 Segmentation

Figure 7.2 is an image of a burned patient. Figure 7.3 is the result of edge extraction from Figure 7.2. We can see that most of the burn edge points are detected, but the contour of the burn is ambiguous. Figure 7.4 represents the original burn patient's image, edge extraction output, the superimposed image of edge points and the original patient's image, and the image of potential edge points, clockwise from the upper left image.

Figure 7.5 shows the outcome of the edge following procedure. From this image we can see most of the contour of the burn.

Figure 7.6 and Figure 7.7 are the results of the multiple isodata procedure corre-

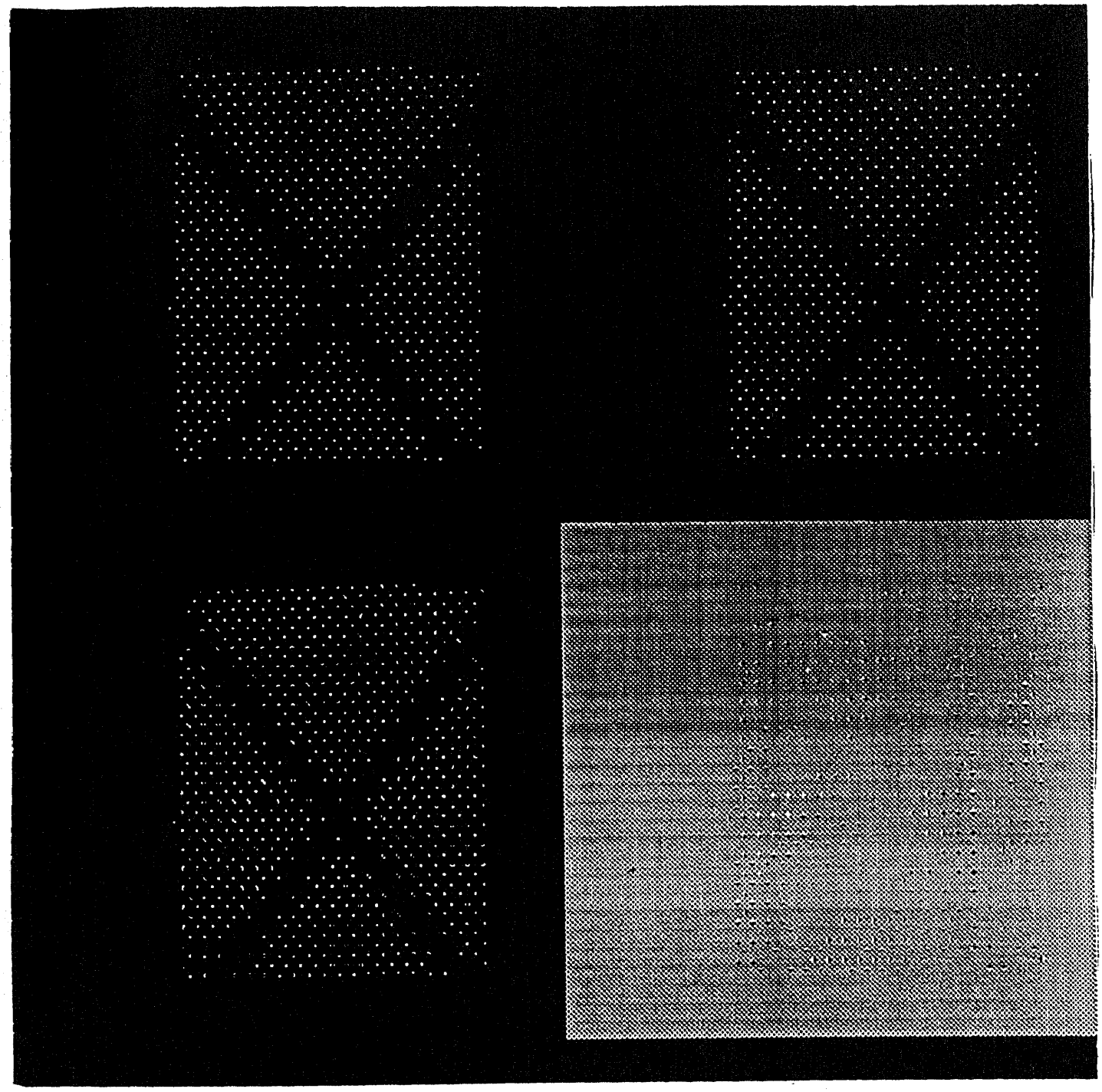

Figure 7.1: The accuracy of the calibration procedure

n	surface area (in^2)	error (%)
1	25.683752	9.162405
2	25.867943	8.510964
3	26.735865	5.441321
4	26.994215	4.527596
5	26.547513	6.107480
6	25.978456	8.120104
7	25.846354	8.587319

Table 7.2: The surface area of disk.

sponding to the original burn patient's image without and with running the reflectance compensation procedure, respectively. In these two images we can distinguish the different degrees of the burn wound from the varieties of the pixel values which are clustered in 5 groups.

7.3 Cubic Spline Interpolation

Numerically, no interpolation procedure perform with one hundred percent accuracy whether or not they have C^0, C^1 or C^2 continuity. Those errors will affect the accuracy of forming space surfaces and calculating the normal vector.

7.4 Surface Area

Table 7.2 is the results of running the whole procedure on a test object which is a disk with known surface area on a flat plane. The surface area of the disk is exactly 28.274364 inches square, because the radius is 3 inches. The n in table is the number of points in Gaussian-Legendre formula. Table 7.3 illustrates the surface area calculated for a square which was pasted on a cylinder. The real surface area of the square is 16 in^2.

n	surface area (in^2)	error (%)
1	13.756767	14.020206
2	14.145632	11.589800
3	14.678452	8.259675
4	14.867459	7.078381
5	14.876576	7.021400
6	14.734145	7.911593
7	14.465784	9.588850

Table 7.3: The surface area of square.

Accoring to these two tables, we can see that the error is normally distributed. We have found that the accuracy is highest when the test areas are smooth, i.e, when the curvature is minimum.

Figures 7.8 to Figure 7.19 show the sequence of steps of one experiment running the entire procedure on a burn patient. This patient is now under the treatment at the Burn Care Unit of St. Barnabas Hospital. Figure 7.8 is the image of part of the burn patient's body illuminated by indoor electric lights. Figure 7.9 is the image with the structured light. Figure 7.10 is the result of running the reflectance compensation procedure on previous image. Figure 7.11 is the output of the bimean thresholding procedure. Figure 7.12 is the result of the classical thinning procedure. Figure 7.13 is the outcome of intersection detection. Figure 7.14 is the result of interpolating the missing points by using bicubic surface patches. Figure 7.15 is the difference between Figure 7.13 and Figure 7.14. All the points in this image are interpolated points. Figure 7.16 is the edge extraction output of Figure 7.8. Figure 7.17 is the output of edge following. Figure 7.18 is the output of multi-level thresholding. As we can see, the results of the segmentation procedures are not very good. We have to use manual contouring with a mouse to segment the burn area. Figure 7.19 is the result of manual

contouring. The surface area of burn in this case is 37.6457 in^2.

We repeated this experiment twice more and got answers of 26.201380 in^2 and 31.913052 in^2. From the Lund and Browder chart for this patient, we estimate the area as 48.5 in^2. Considering the poor calibration of the imaging system and rough estimate using the Lund anf Browder chart, we find that these preliminary resluts are encouraging.

7.5 Conclusions

The overall accuracy of calculating the surface area is within 85% to 95% accuracy.

Although the results that we have so far is promising, we need more sophisticated approaches for solving the segmentation problems. Furthermore, if the patient is hairy, the grid stripes will be broken and the range data will be discontinuous. A more sophisticated grid labeling procedure can partially overcome this problem. Foremost in our agenda is to continue to conduct experiments with real patients to compute the accuracy, reproducibility and resolution of burn area measurements.

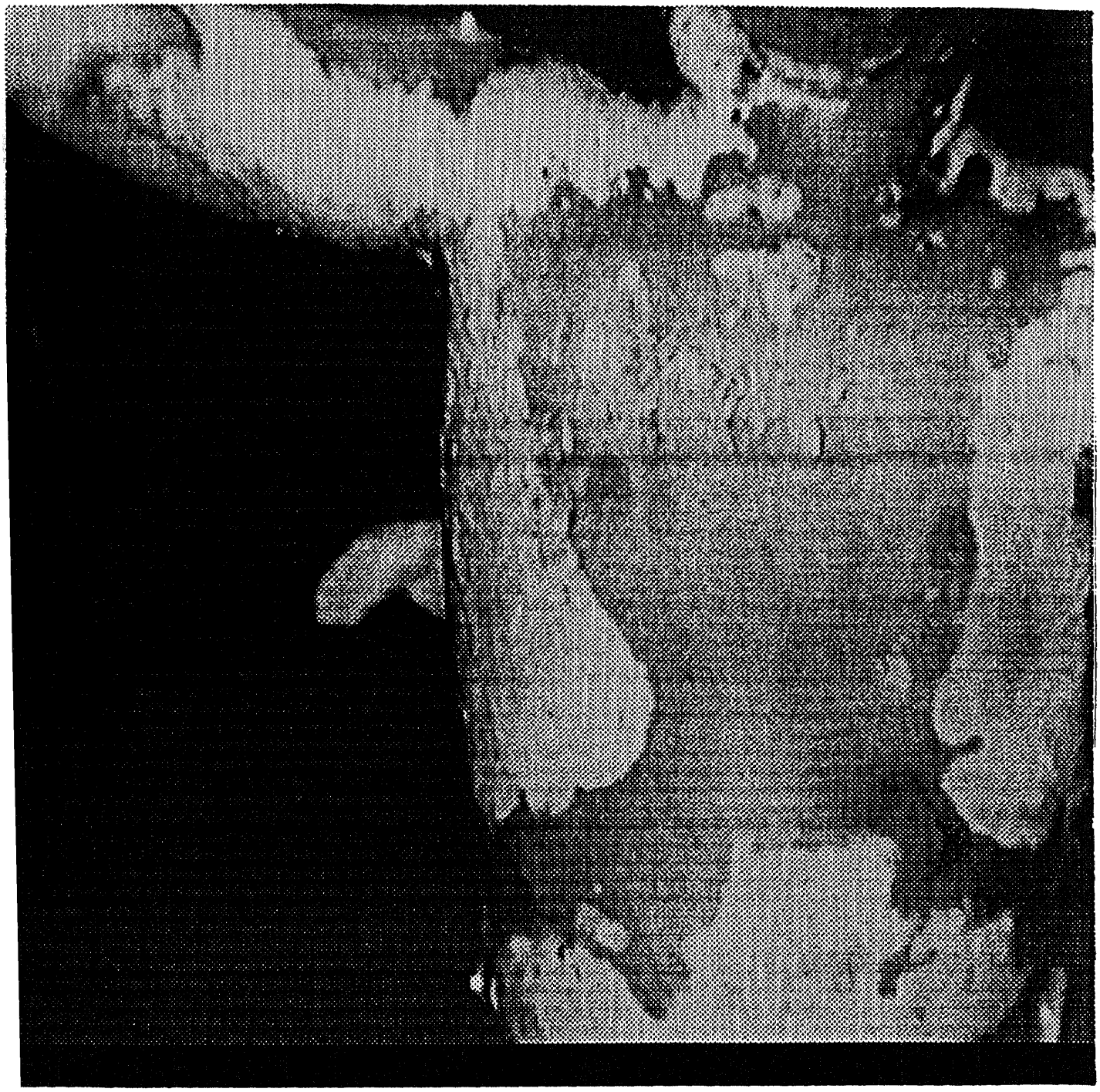

Figure 7.2: The image of a burned wound patient

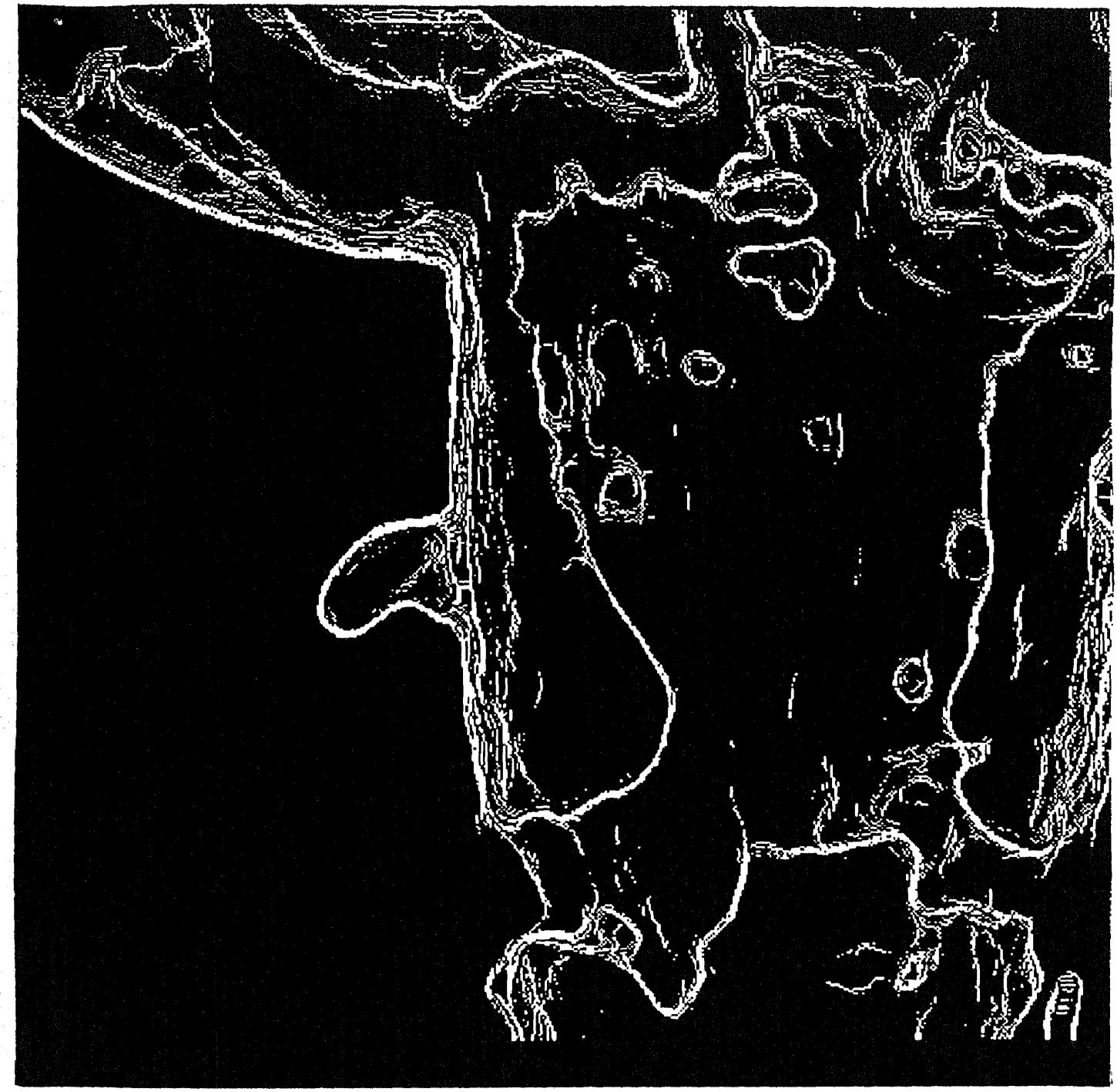

Figure 7.3: The result of edge extraction

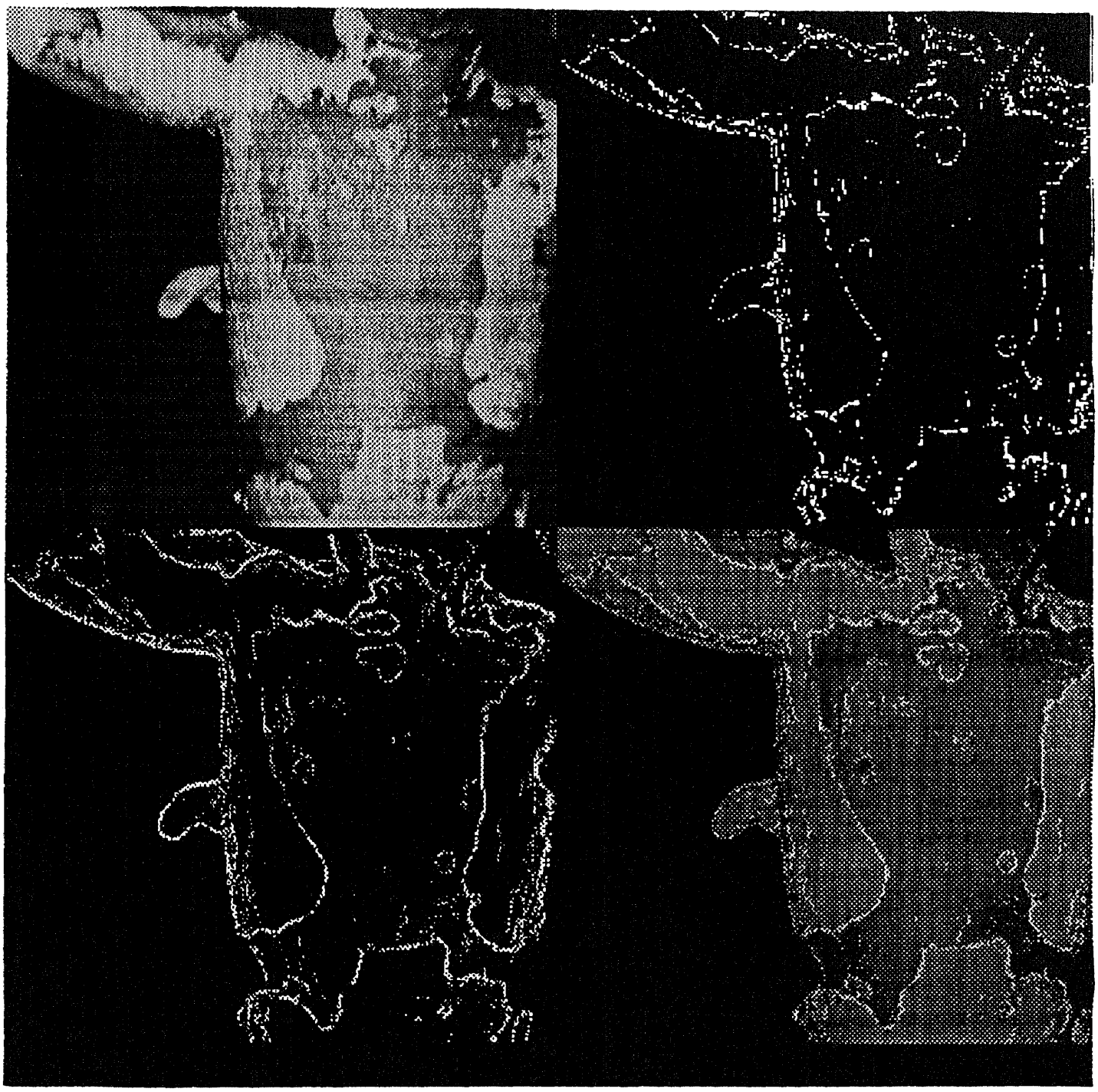

Figure 7.4: The edge image and the potential edge image

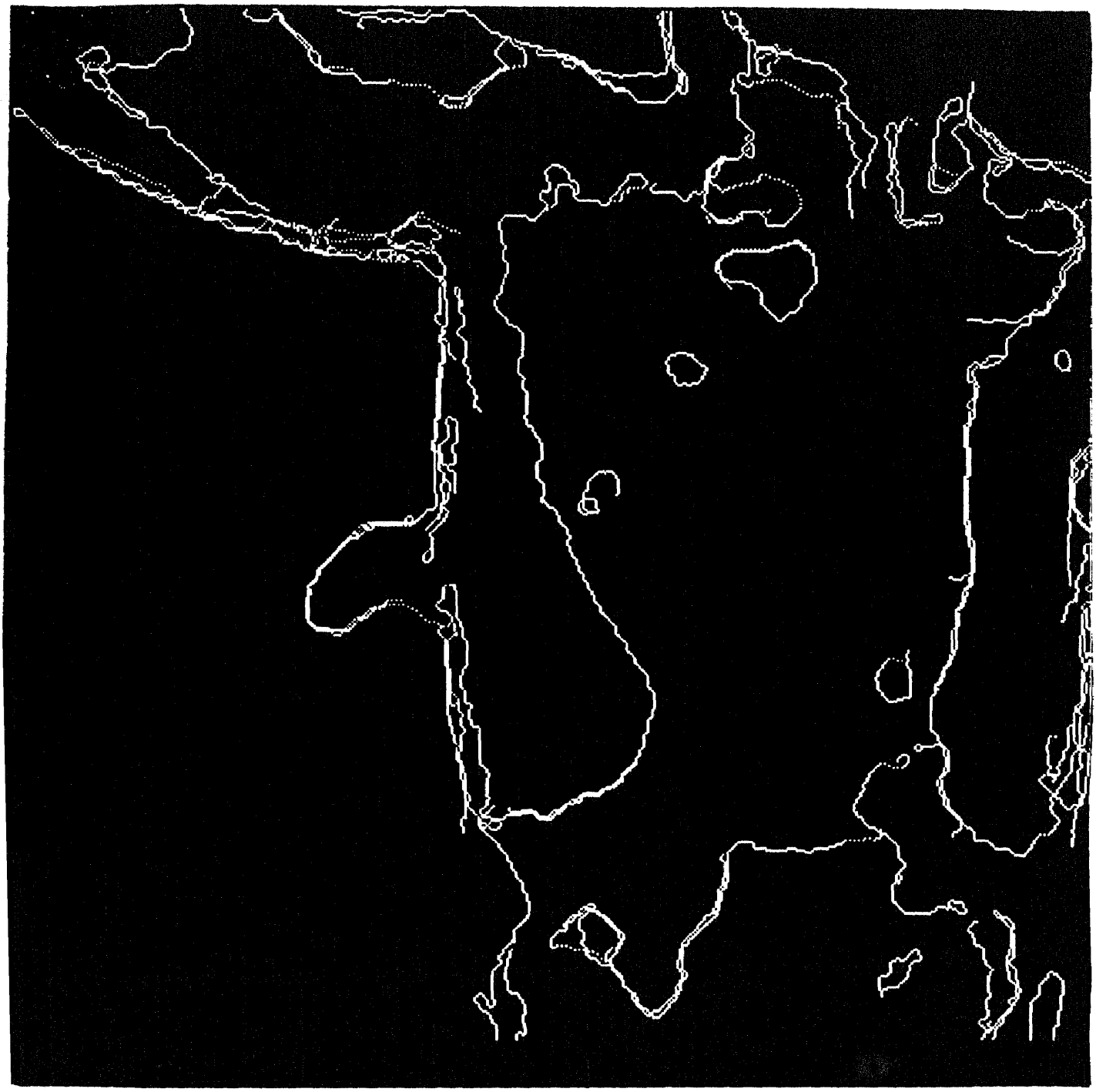

Figure 7.5: The result of edge following

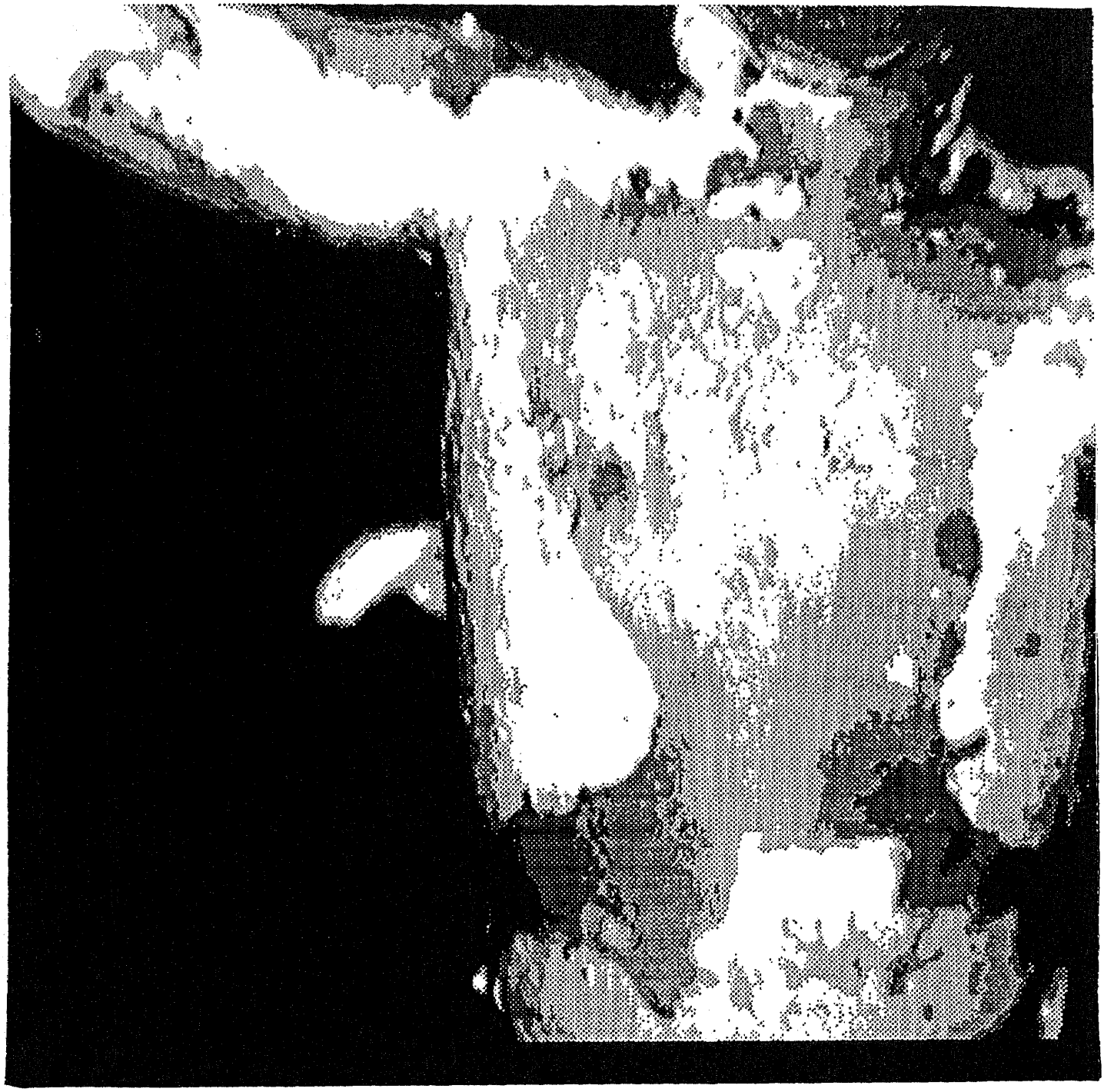

Figure 7.6: The result of multiple isodata of burned patient's image

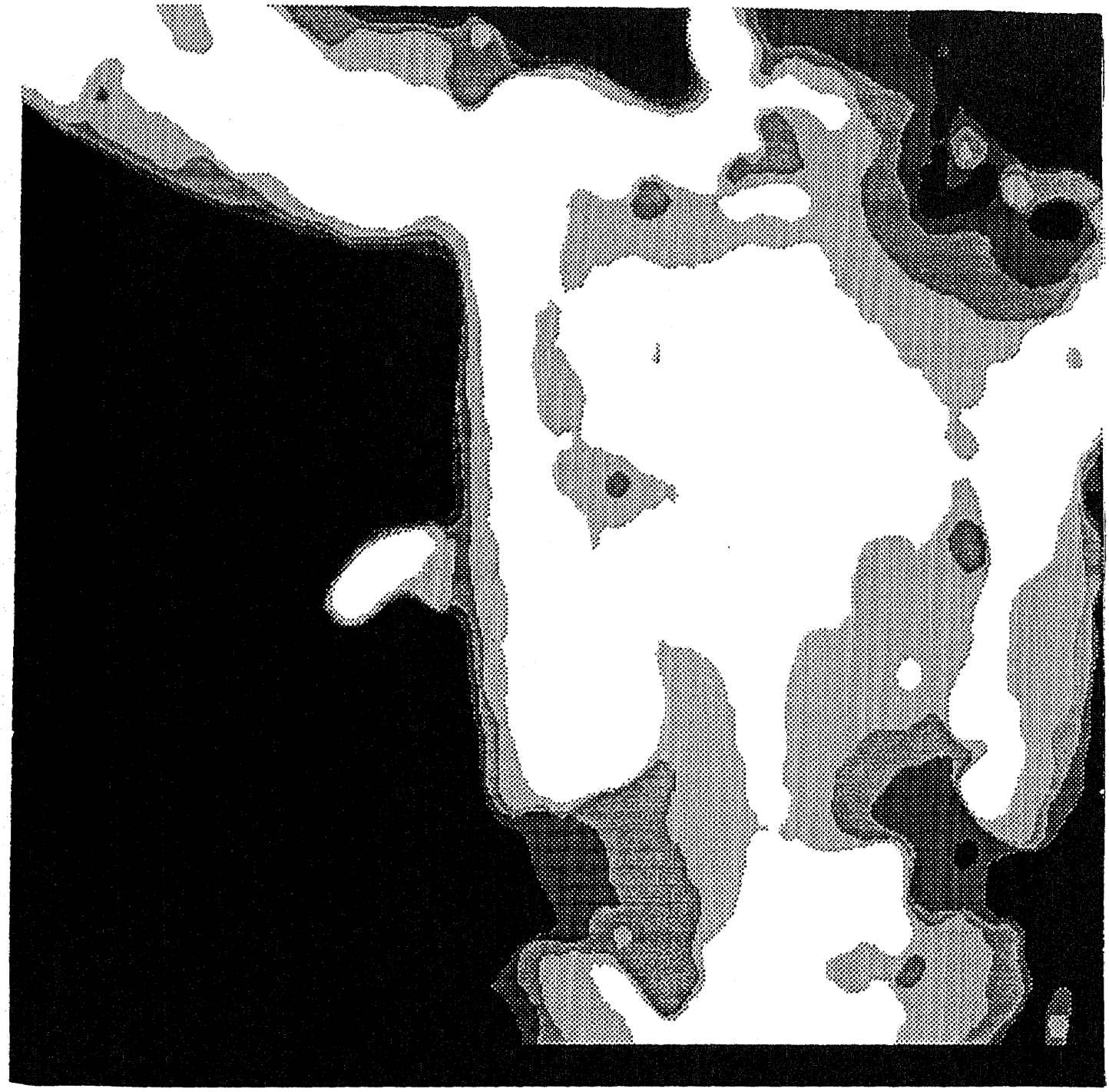

Figure 7.7: The result of multiple isodata from compensated burned patient image

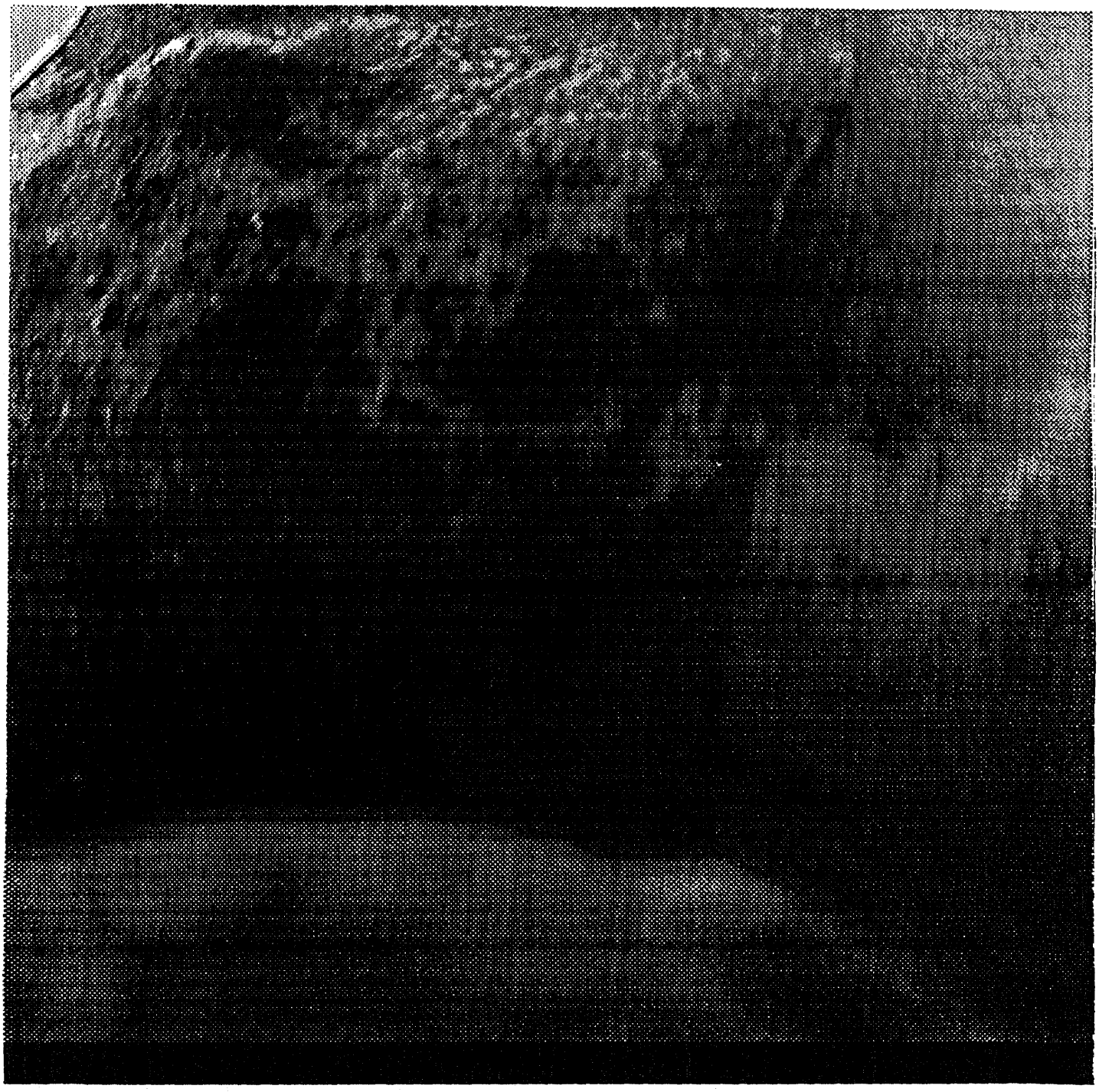

Figure 7.8: The Image of Burn Patient

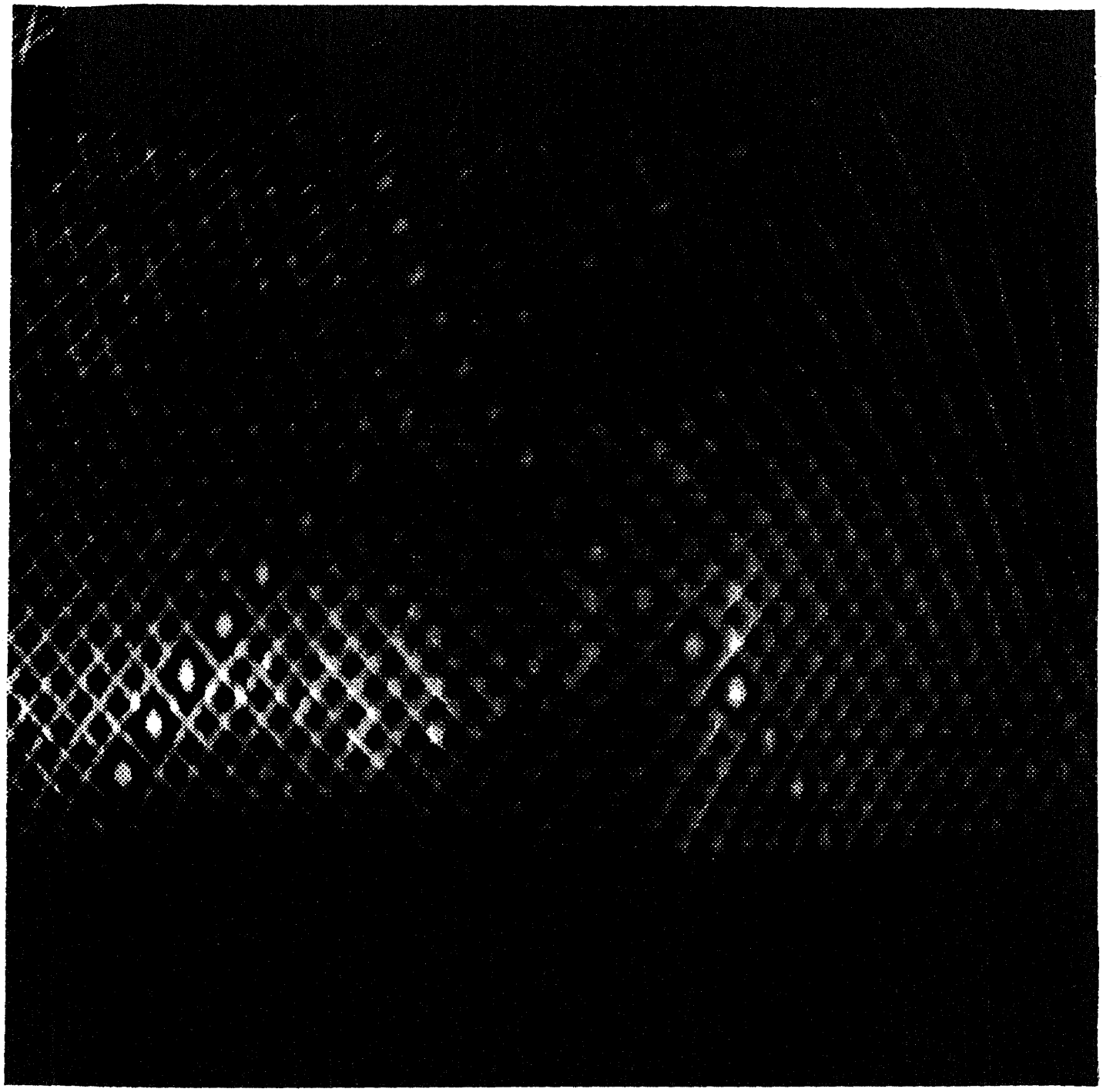

Figure 7.9: The image of Burn Patient with Sturctured light

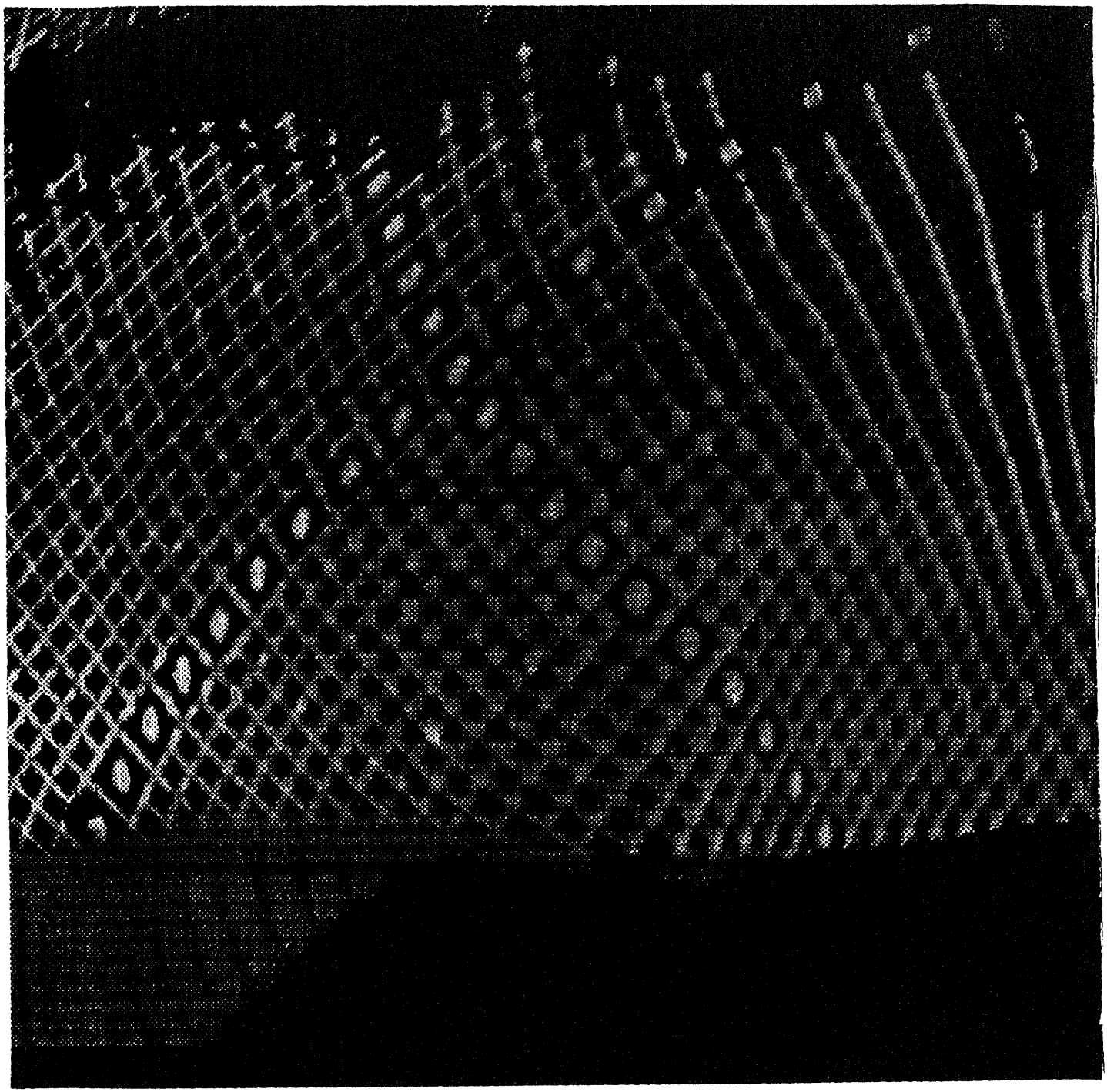

Figure 7.10: The result of reflectance compensation

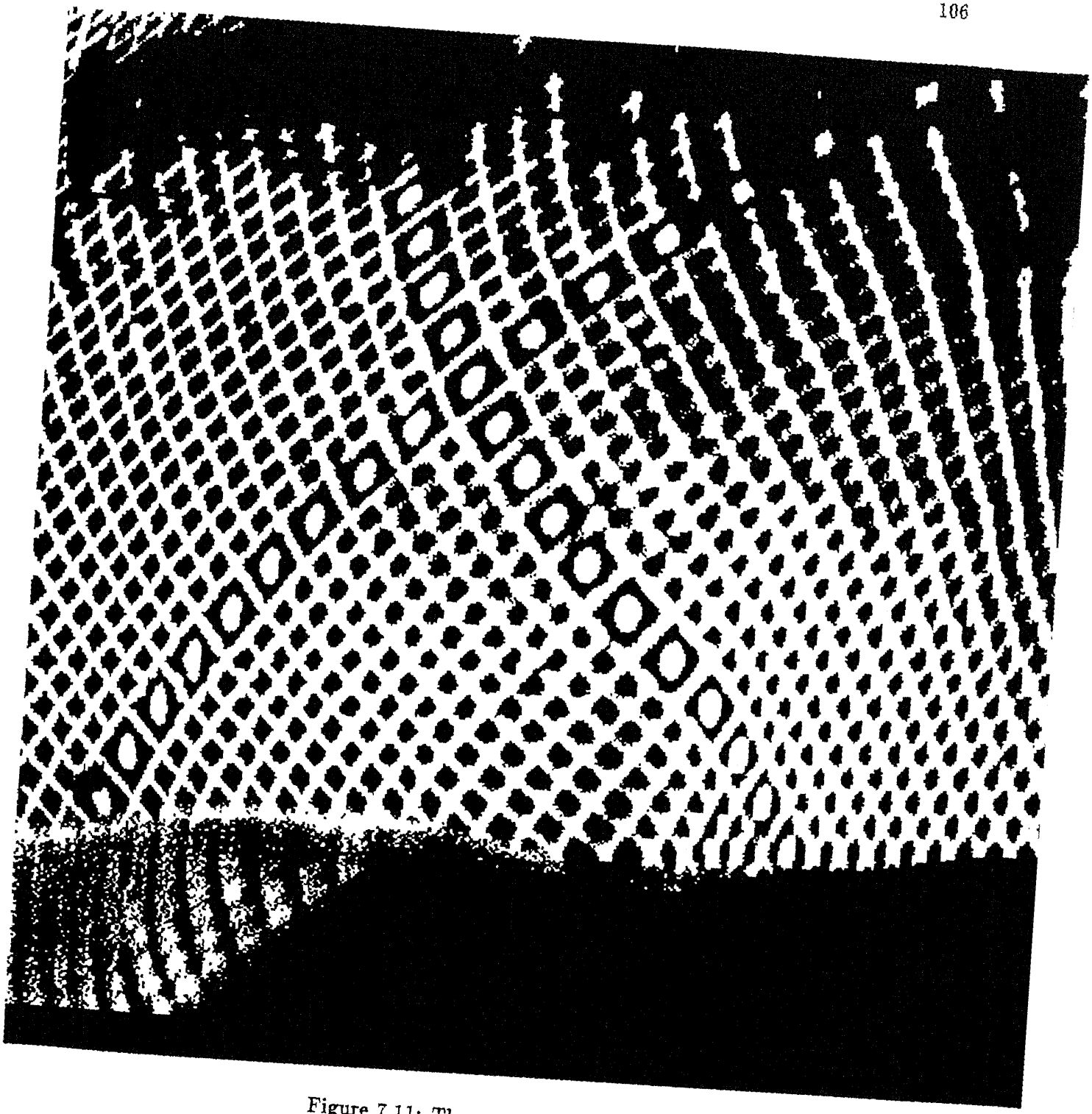

Figure 7.11: The result of bimean

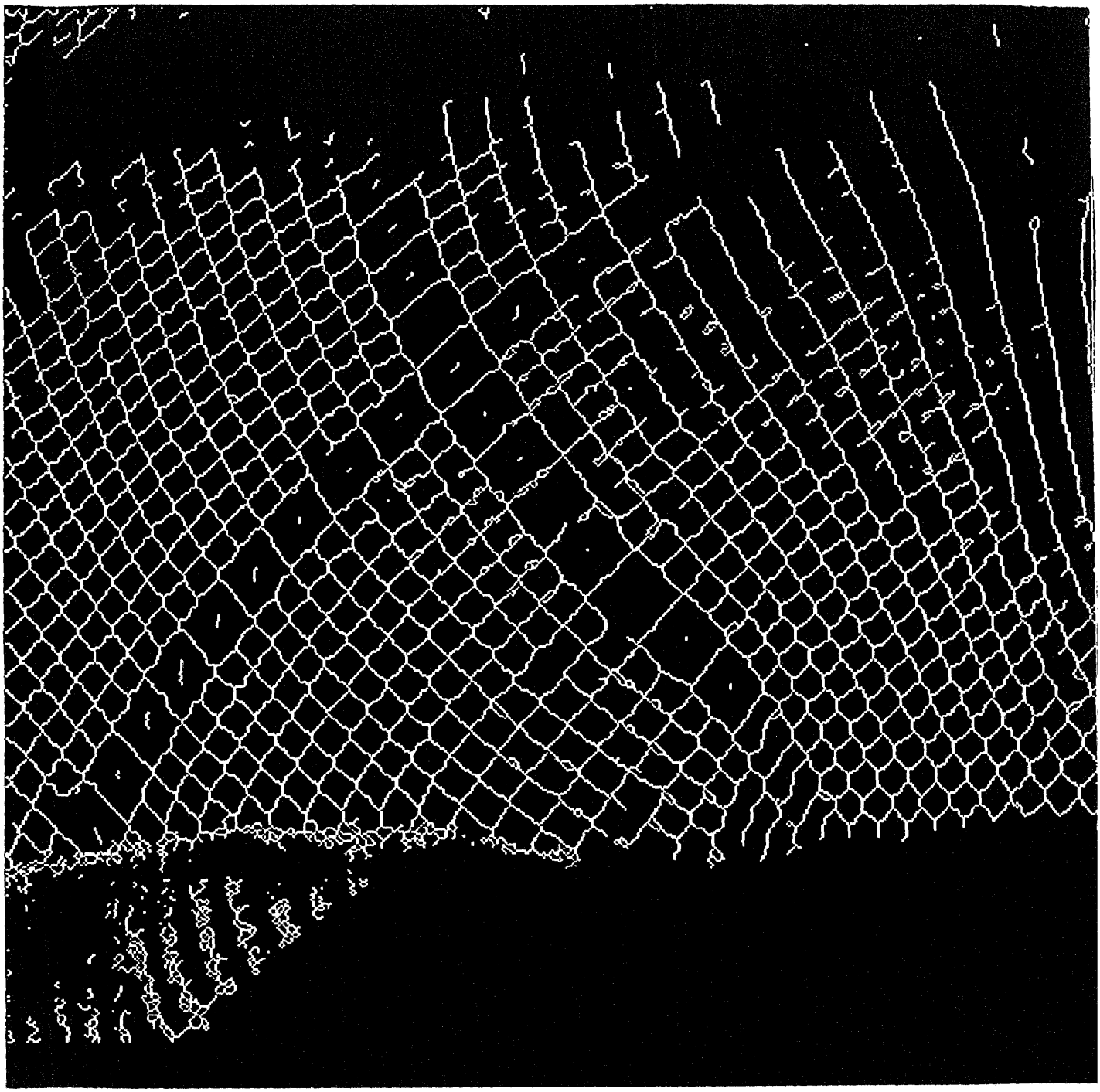

Figure 7.12: The result of classical thinning

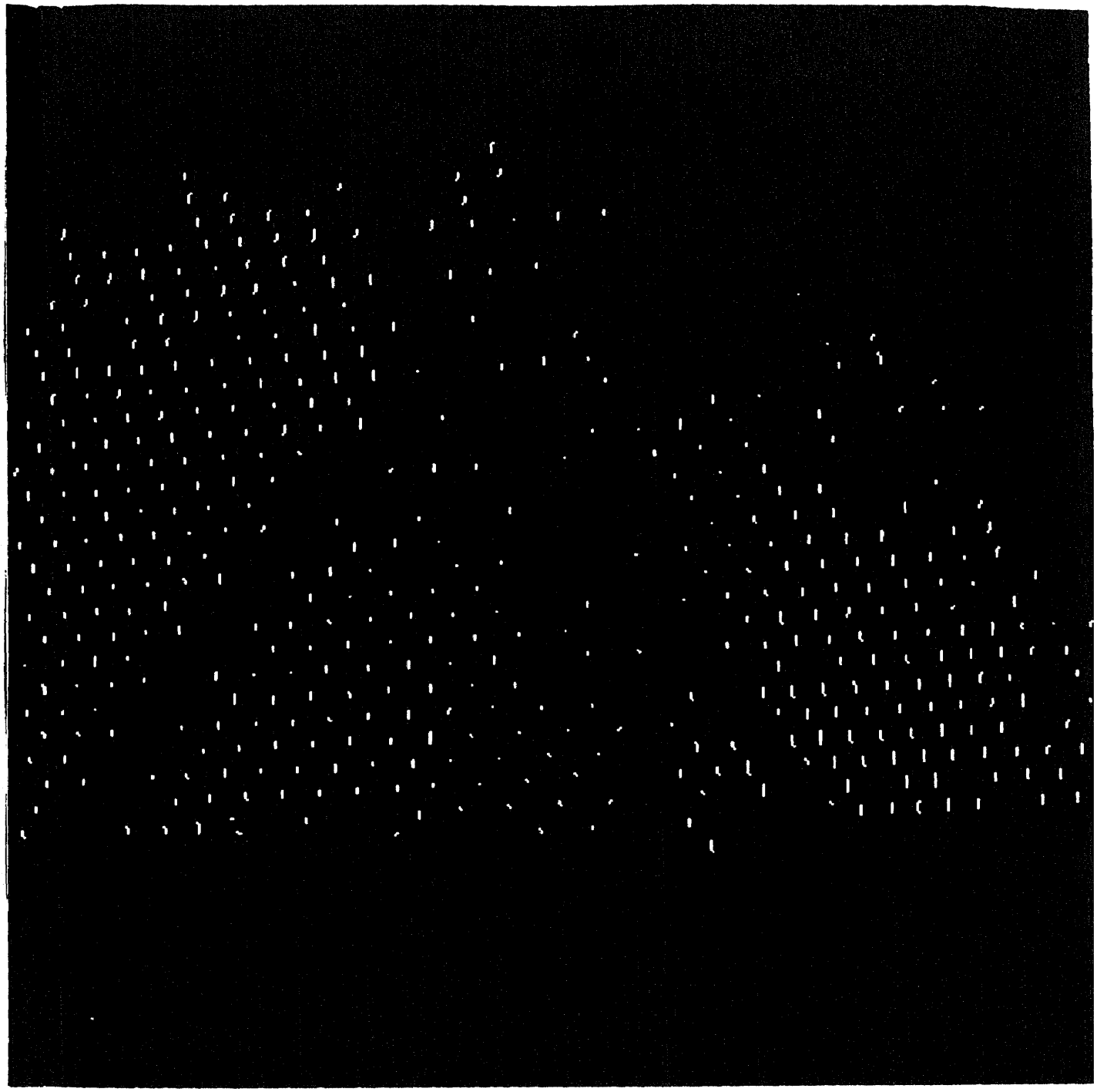

Figure 7.13: The result of intersection detection

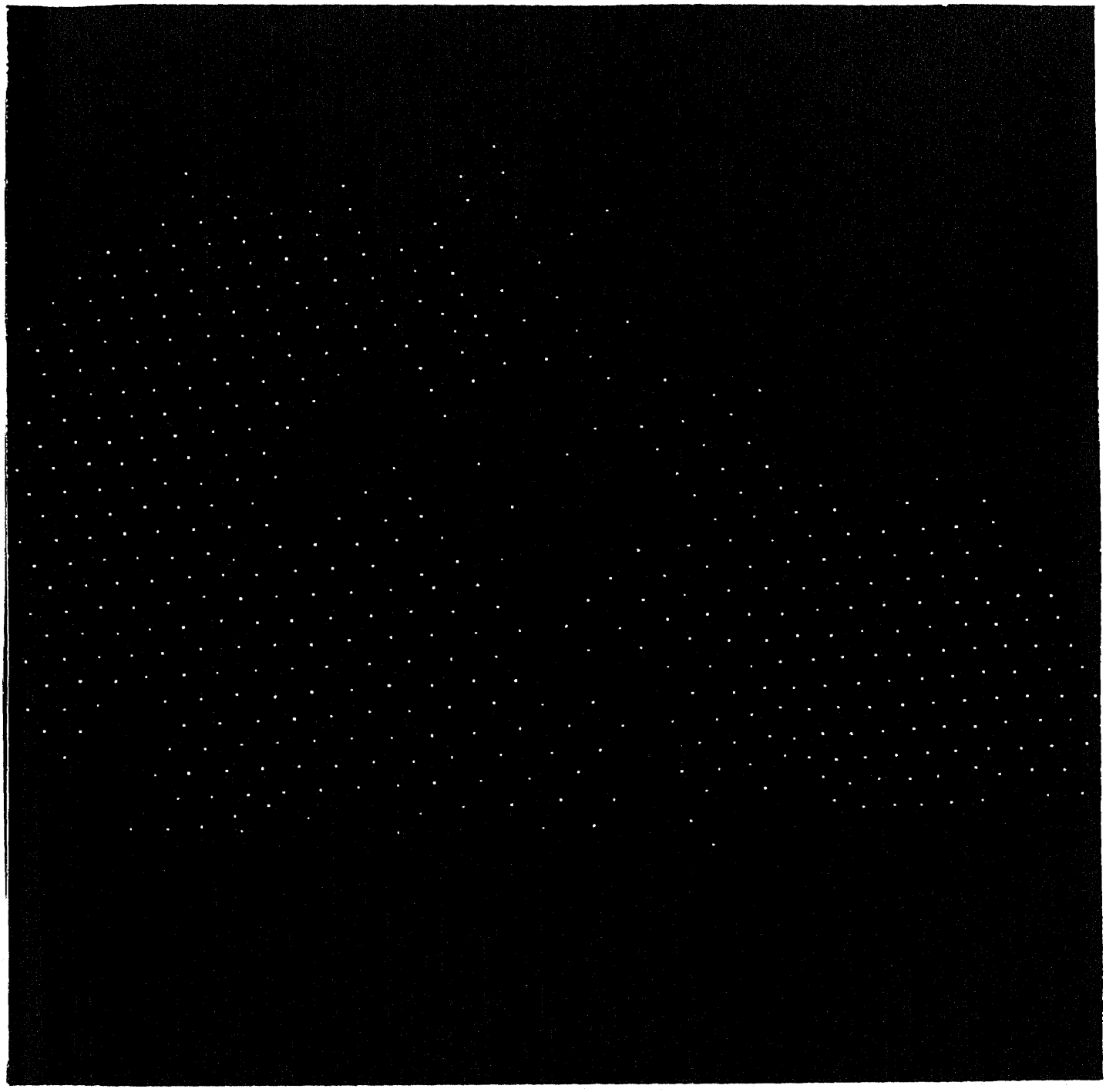

Figure 7.14: The result of surface interpolation

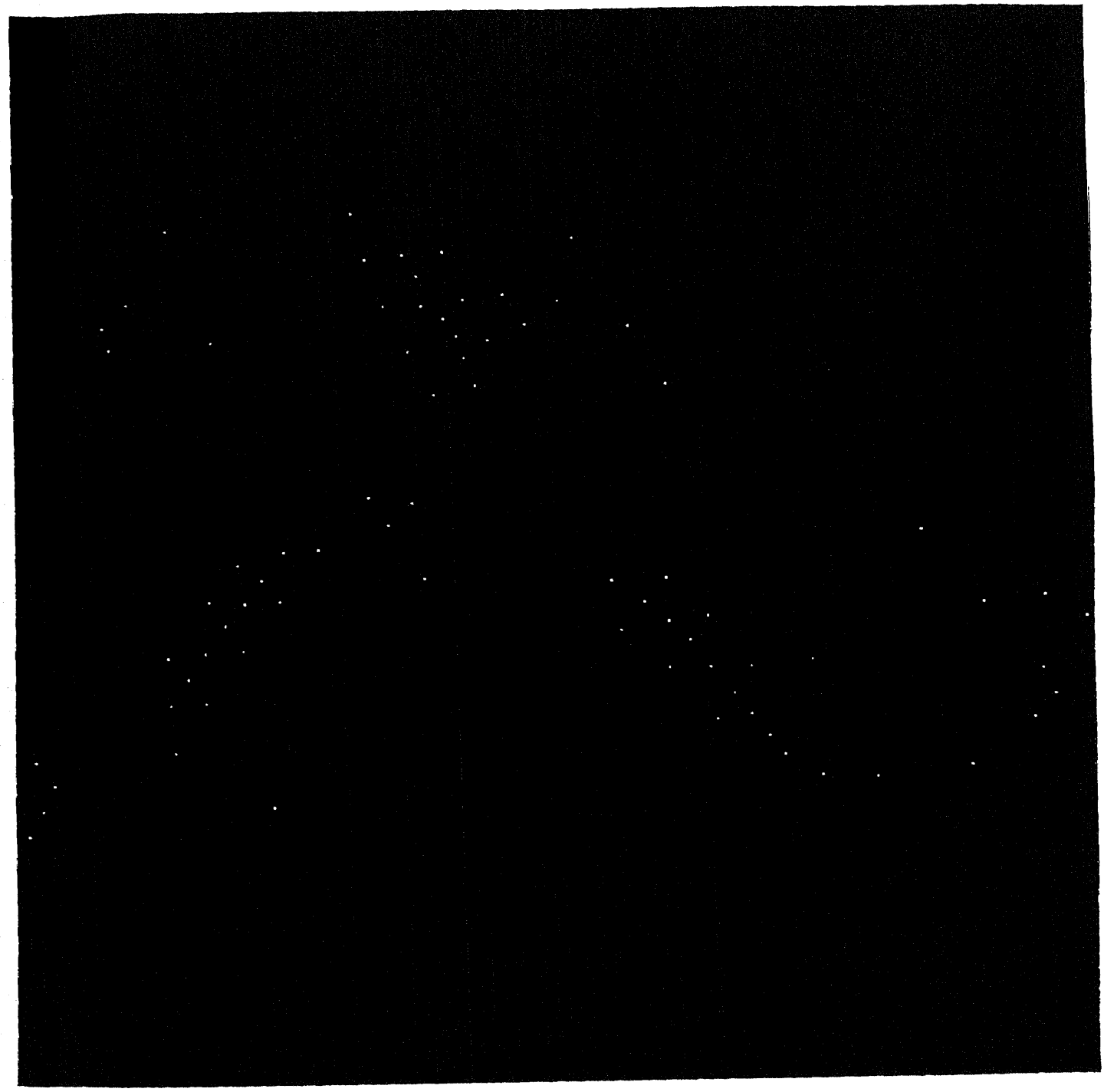

Figure 7.15: The interploated points

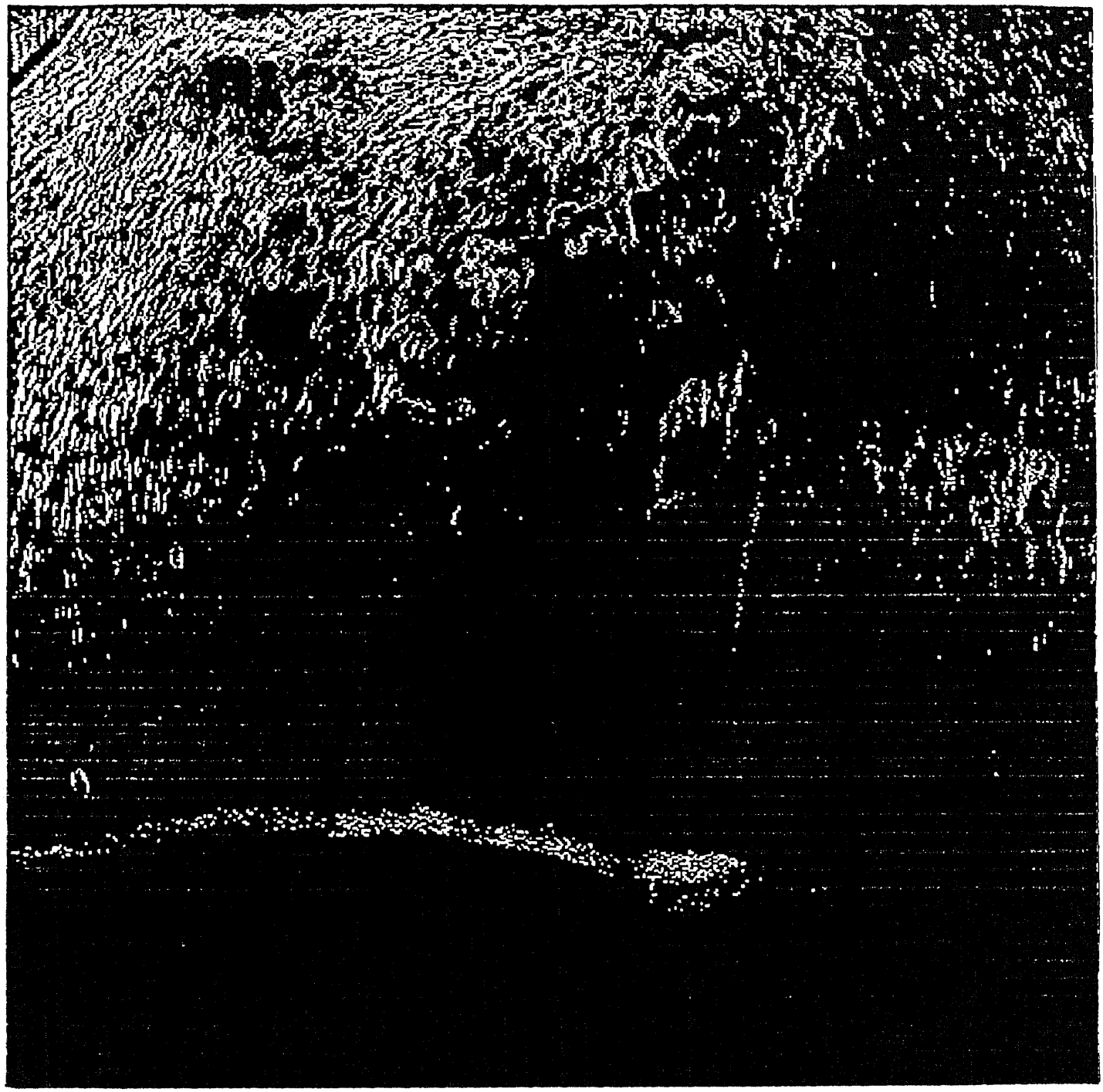

Figure 7.16: The result of edge extraction

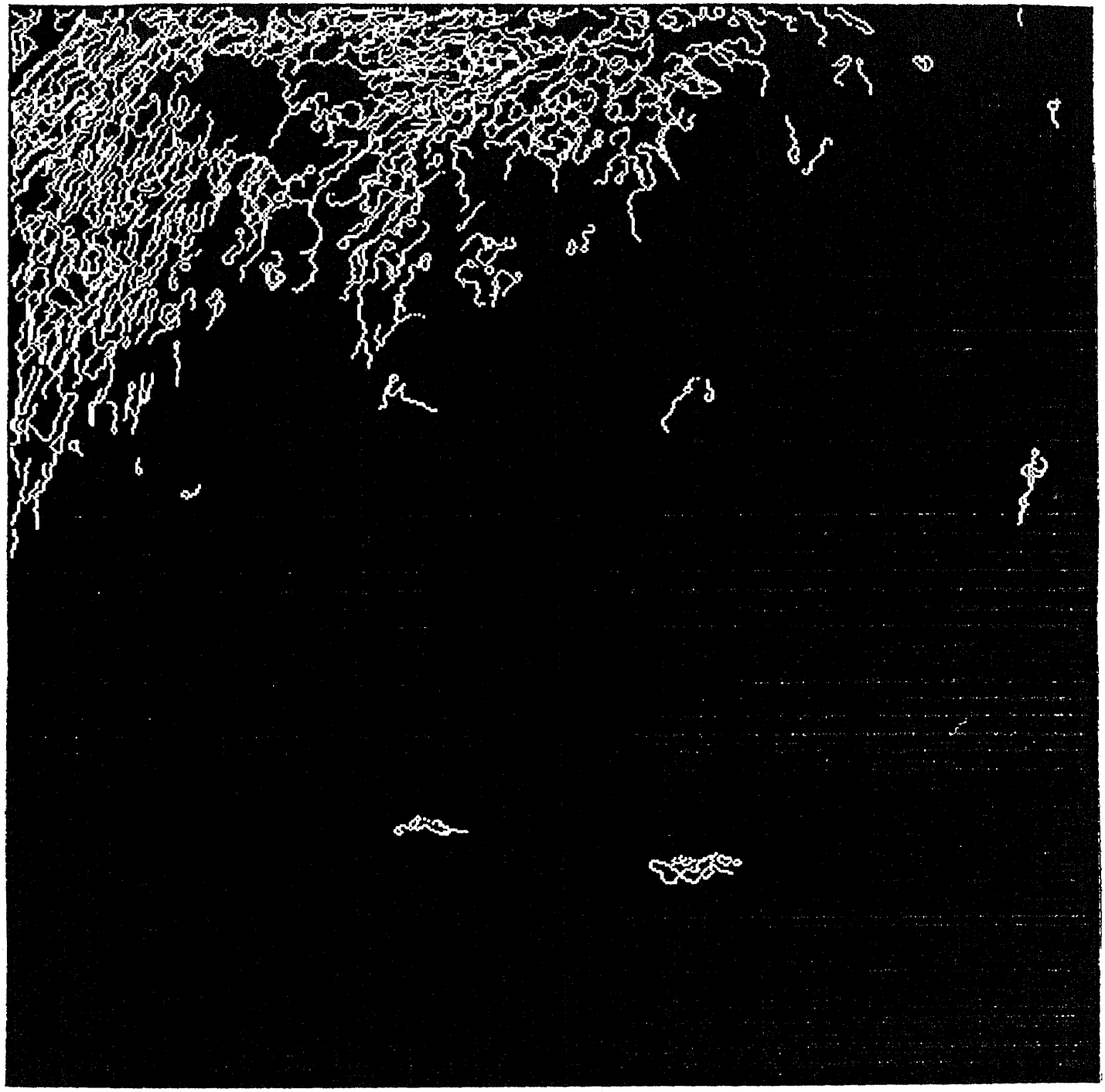

Figure 7.17: The result of edge following

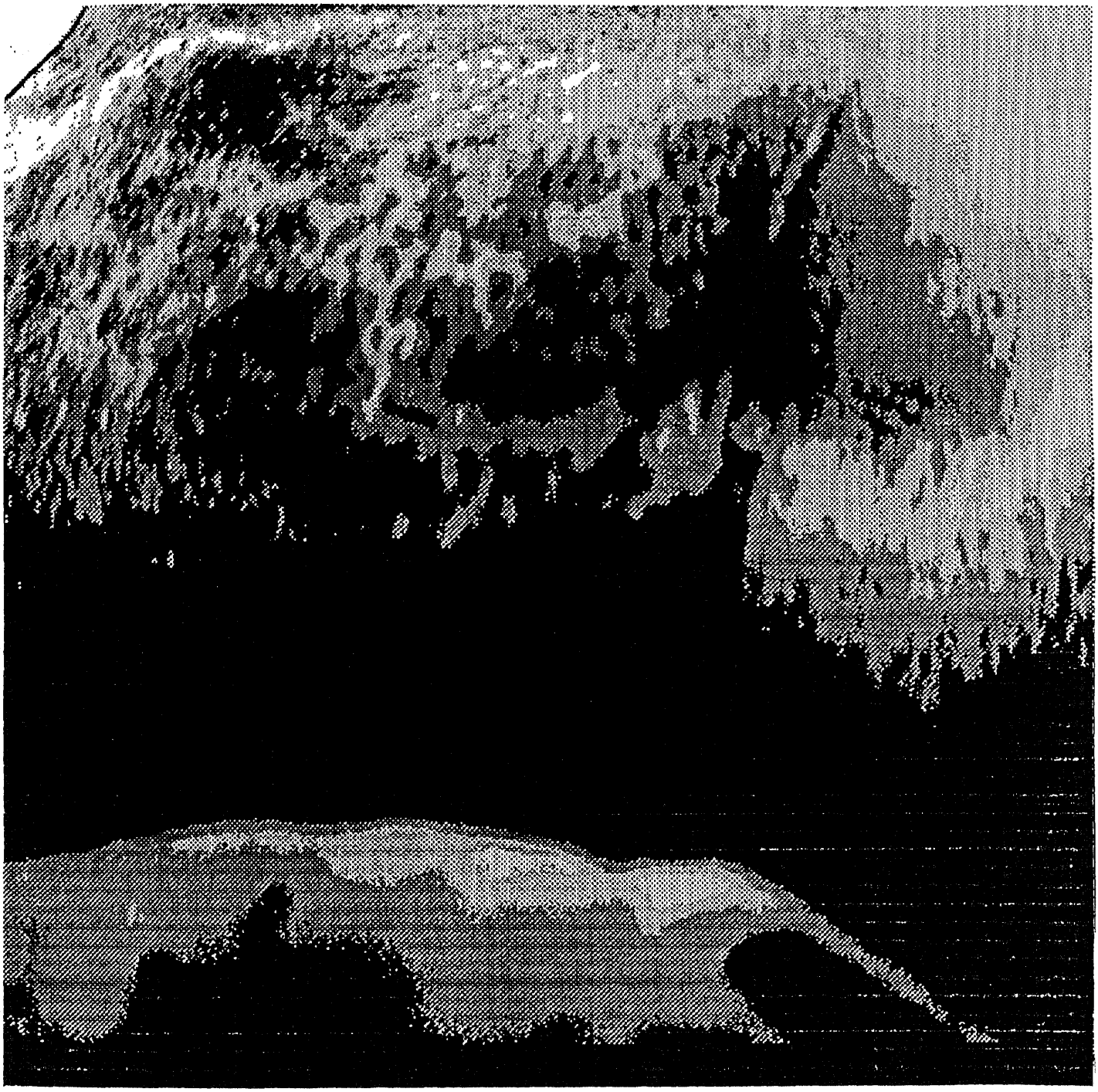

Figure 7.18: The result of multi-level thresholding

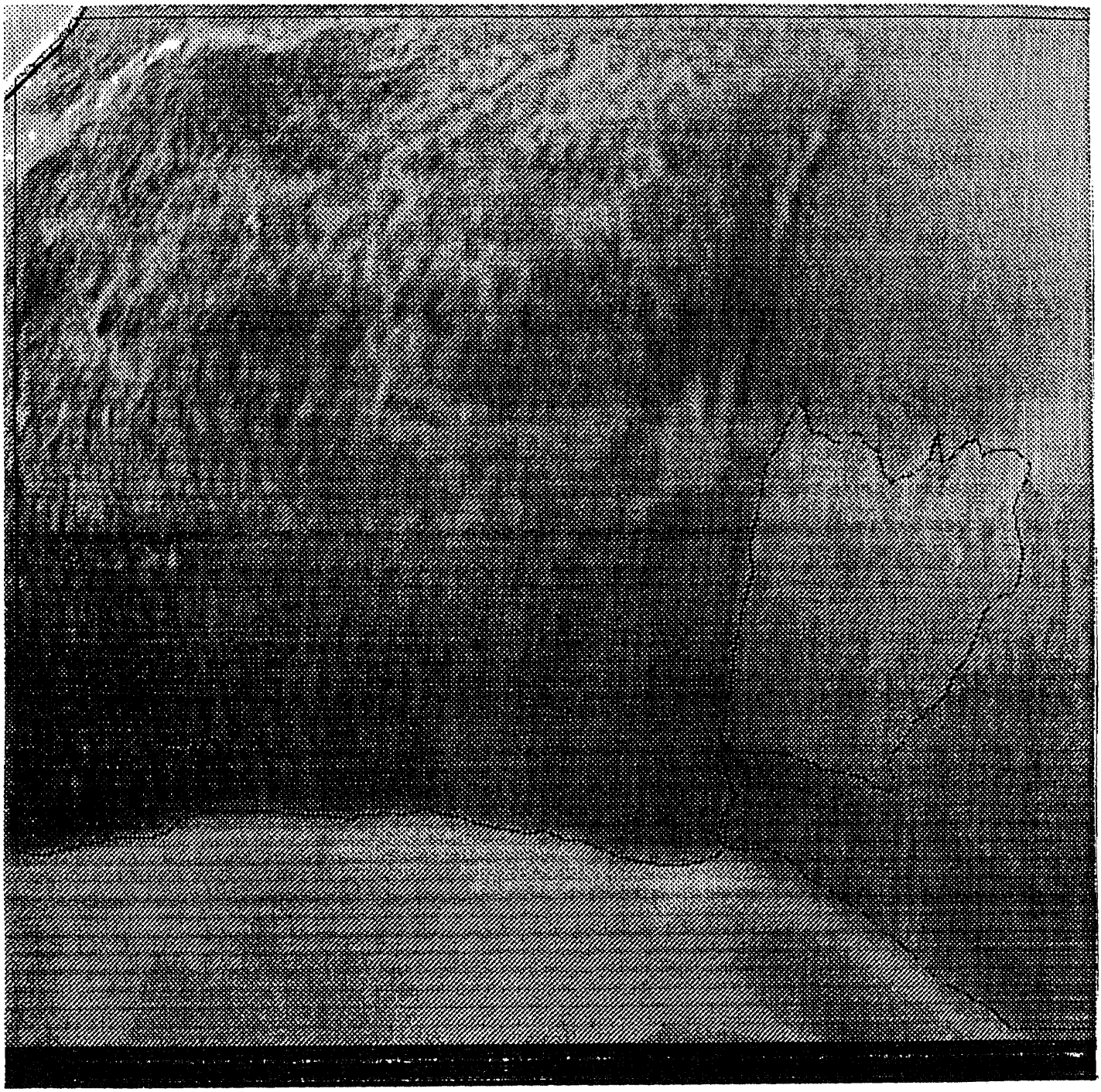

Figure 7.19: The result of manual contouring

Bibliography

[1] Martin D. Altschuler, Kyongtae Bae, Bruce R. Altschuler, Jerome T. Dijak, Louis A. Tamburino, and Barbara Woolford.
Robot Vision by Encoded Light Beams.
Kluwer academic publishers, Takeo Kanade, editor, 1987.

[2] Michael R. Anderberg.
Cluster Analysis for Applications.
Academic Press, New York, 1973.

[3] G.H Ball and D.J. Hall, editors.
A Novel Method of Data Analysis and Pattern Classification., Standford Res. Inst., Menlo Park, CA, 1965.
AD 699616.

[4] G.H Ball and D.J. Hall, editors.
PROMENADE – An on-line Pattern Recognition System., Standford Res. Inst., Menlo Park, CA, 1967.
Report. No RADC-TR-67-310, AD 822174.

[5] Dana H. Ballard and Christopher M. Brown.
Computer Vision.
McGraw Hill, 1986.

[6] M. Basseville, B. Espiau, and J. Gasnier.
Edge detection using sequential methods for change in level– part 1 : a sequential edge detection algorithm.
IEEE Transactions on Acoust. Speech. Signal Processing, 29:24–31, 1981.

[7] S.G. Berkow.
A method for estimating the extensiveness of lesions (burns and scalds) based on surface area proporions.
Arch. Surg, 8:138–148, 1924.

[8] P.J. Besl and R.C. Jain.
Range image understanding.
In *Proc. Computer Vision and Pattern Recognition.*, pages 430–449, San Francisco, CA, June 1985.

[9] M. Born and E. Wolf.
Principles of Optics.
Pergamon, Oxford, 1965.

[10] G.A. Boutry.

Instrumental Optics.
Interscience, New York, 1962.

[11] K.L Boyer and A.C Kak.
Color-encoded structure light for rapid range sensing.
IEEE Trans. PAMI, 1:14–28, Jan 1987.

[12] John F. Canny.
A computationational approach to edge detection.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 8:679–698, 1986.

[13] Bor-Dong Chen and Pepe Siy.
Forward/backward contour tracing with feedback.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 9:438–446, May 1987.

[14] B. Dubois and E.F. Dubios.
The measurement of the surface area of man.
A.M.A. Arch. Intern., Med., 15:868–881, 1915.

[15] R.O. Duda and P.E. Hart.
Pattern Classification and Scenc Analysis.
Wiley, New York, 1973.

[16] Stanley M. Dunn, Richard L. Keizer, and Jongdaw Yu.
Measuring soft tissue contour information with structured light.
In *1988 International Conference of Electronic Imaging*, 1988.

[17] Stanley M. Dunn, Richard L. Keizer, and Jongdaw Yu.
Measuring the area and volume of the human body with structured light.
IEEE Trans. on Systems, Man, and Cybernetics special issue on computer vision, 1989.
To appear.

[18] P.H. Eichel, E.J. Delp, K. Koral, and A.J. Buda.
A method for a fully automatic definition of coronary arterial edges from cineangiogram.
IEEE Trans. on Medical Imaging, 7:313–320, Dec. 1988.

[19] M.A. Fischler and R.C. Bolles.
Random sample consensus: a paradigm for model filling with applications to image analysis and automated cartography.
Comm. ACM, 24:381–395, June 1981.

[20] W. Frobin and E. Hierholzer.
Calibration and model reconstruction in analytical close-range stereophotogrammetry - part 1: mathematical fundamentals.
Photogrammetric Engineering and Remote Sensing, 48:67–72, Jan 1982.

[21] W. Frobin and E. Hierholzer.
Calibration and model reconstruction in analytical close-range stereophotogrammetry - part 2: spacial evaluation procedure for rasterstereograph and moire topography.
Photogrammetric Engineering and Remote Sensing, 48:215–220, Feb 1982.

[22] W. Frobin and E. Hierholzer.
Rasterstereography: a photogrammetric method for measurement of body surface.

Photogrammetric Engineering and Remote Sensing, 47:1717–1724, Dec 1981.

[23] FW. Fuller, E.H. Mansour, P.E. Engler, and B. Shuster.
The use of planimetetry for calculating the surface area of a burn wound.
Journal of Burn Care and Rehabilitation, 6, Jan/Feb 1985.

[24] Rafael C. Gonzalez and Paul Wintz.
Digital Image Processing.
Addison Wesley, 1987.

[25] E.L. Hall.
Measuring curved surfaces for robot vision.
Computer, 15:42–54, Dec. 1982.

[26] R.M. Haralick.
The digital step edge from zero crossings of second directional derivatives.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 6:58–68, 1984.

[27] J.A. Hartigan and M.A. Wong.
A k-means clustering algorithm.
Applied Statistics, 28:100–108, 1980.

[28] Berthold K. P. Horn.
Understanding image intensities.
Artificial Intelligence, 8:201–231, 1977.

[29] R.A Jarvis.
A perspective on range finding techniques for computer vision.
IEEE Trans. PAMI, PAMI-5:122–139, 1983.

[30] F.A. Jenkin and H.E. White.
Fundamentals of Optics.
McGraw-Hill, New York, 1976.

[31] Richard L. Keizer.
Recovering the shape of the human face.
Innovation and Technology in Biology and Medicine, 1988.
To appear.

[32] Richard L. Keizer and Stanley M. Dunn.
Marker grid labeling.
In *IEEE Conference on Computer Vision and Pattern Recognition*, 1989.

[33] G.A. Knayasi, G.F. Crikelair, and B. Cosman.
The rule of nines: its history and accuracy.
Plast. Reconstr. Surg, 41:560–563, 1968.

[34] Amlan Kundu and S.K. Mitra.
A new algorithm for image edge extraction using a statistical classifier approach.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 9:569–577, 1987.

[35] C. Lund and N. Browder.
The estimation of areas of burns.
Surg. Gynes. and Obst., 79:352–358, 1944.

[36] J.B. MacQueen.
Some methods for classification and analysis of multivriate observations.

In *Proc. Symp. Math. Statist. and Probability.*, pages 281–297, Univ. of California Press, Berkeley, 1976.

[37] D. Marr and E. Hildreth.
Theory of edge detection.
In *Proc. Royal Society*, pages 187–217, London, 1980.

[38] A. Martelli.
Edge detection using heuristic search methods.
Computer, Graphics and Image Processing, 1:169–182, 1972.

[39] C.A. McPherson, J.B.K. Tio, F.A. Asdjadi, and E.L. Hall.
Curved surface representation for image recognition.
Conf. Pattern Recognition and Image Processing, 363–369, June 1982.

[40] J.W. Modestino and R. W. Fries.
Edge detection of noisy images using recursive digital filters.
Computer, Graphics and Image Processing, 6:409–433, 1977.

[41] Michael E. Mortenson.
Geometric Modeling.
Wiley, New York, 1985.

[42] Vishvjit S. Nalwa and Thomas O. Binford.
On detecting edges.
IEEE Trans. on Pattern Analysis and Machine Intelligence., 8:699–714, 1986.

[43] L.S. Nichter, C.A. Bryant, and R.F. Edlich.
Efficacy of burned surface area estimates calculated from charts – the need for a computer – based model.
The Journal of Trauma, 6:477–481, 1985.

[44] Michael O'Flynn.
Probabilities, Random variables , and Random Processes.
Harper and Row, New York, 1982.

[45] Theo Pavlidis.
Algorithms for Graphics and Image Processing.
Computer Science Press, Rockville, 1982.

[46] K.S. Pennington and P.M Will.
Grid coding: a novel technique for image processing.
Proc. IEEE, 60:669–680, June 1972.

[47] K.S. Pennington and P.M Will.
Grid coding: a preprocessing technique for robot and machine vision.
Artificial Intelligence, 2:319–329, 1971.

[48] K.S. Pennington and P.M. Will.
A grid-coded technique for recording 3-dimensional scenes illuminated with ambient light.
Optics Commun., 2:167–169, 1970.

[49] D. Pollard.
Strong consistency of k-means clustering.
Annals of Statistics, 9:135–140, 1981.

[50] J.L. Posdamer and M.D. Altschuler.
Surface measurement by space-encoded projected bean system.
Computer Graphic and Image Processing, 18:1–17, 1982.

[51] M. Potmesil.
Generating models of solid objects by matching 3d surface segments.
Proc. 7th Int. Conf. Pattern Recognition, 203–206, July 1984.

[52] M. Potmesil and H. Freeman.
Curved surface representation utilizing data extracted from multiple photographic images.
In *Workshop on the Representation of Three-Dimensional Objects.*, pages H1–H26, Philadelphia, PA, May 1979.

[53] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling.
Numerical Pecipes in C. The Art of Scientific Compution.
Cambridge University Press., Cambridge, 1988.

[54] Sarmishtha Meeta Rath.
The Non-Contacting Area Assessment of Burn Wound Using Computer Image Methods.
Master's thesis, New Jersey Institute of Technology, 1987.

[55] David F. Rogers and J.A. Adams.
Mathematical Elements for Computer Graphics.
Wiley, 1980.

[56] A. Rosenfeld and A.C. Kak.
Digital Picture Processing.
Academic Press, New York, 1976.

[57] Carol E.H. Scott-Conner and Harry F. Conner.
Burn area measurement by computerized planimetry.
The Journal of Trauma, 28:638–641, 1988.

[58] G. Stockman and G. Hu.
Sensing 3d surface patches using a projected grid.
In *IEEE Computer Society Conf. Computer Vision and Pattern Recognition.*, Miami Beach, FL., 1986.

[59] K. Sugihara.
Regular pattern projection for surface measurement.
In *The Second Inter. Symp. on Robotics Research*, MIT, 1985.

[60] J.B.K Tio, C.A. McPherson, and E.L. Hall.
Curved surface for robot vision.
Conf. Pattern Recognition and Image Processing, 370–378, June 1982.

[61] V. Torre and T.A. Poggio.
On edge detection.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 8:147–163, 1986.

[62] Roger Y. Tsai.
A versatile camera calibration technique for high-accuracy 3d vision metrology using off-the-shelf tv cameras and lenses.
IEEE Journal of Robotics and Automation, RA-3:323–344, August 1987.

[63] F.R.D Velasco.
Thresholding using the isodata clustering algorithm.
IEEE Trans. on Systems, Man, Cybernetics, SMC-10:771–774, Nov 1980.

[64] Y.F. Wang, A. Mitiche, and J.K Aggarwal.
Computation of surface orientation and structure of objects using grid coding.
IEEE Trans. PAMI, PAMI-9:129–127, 1987.

[65] Y.F. Wang, A. Mitiche, and J.K Aggarwal.
Inferring local surface orientation with the aid of grid coding.
In *Proc. Third Workshop on Computer Vision: Representation and Control.*, pages 96–104, Bellaire,MI, Oct 1895.

[66] Y.F. Wang, A. Mitiche, and J.K Aggarwal.
On modeling 3d objects using multiple sensory data.
In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1098–1103, Raleigh, NC., 1987.