

BREAST IMAGE REGISTRATION USING A TEXTURAL TRANSFORMATION

BY

RADHIKA SIVARAMAKRISHNA

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba

© September, 1997: Radhika Sivaramakrishna



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-23666-8

**THE UNIVERSITY OF MANITOBA
FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION PAGE**

BREAST IMAGE REGISTRATION USING A TEXTURAL TRANSFORMATION

BY

RADHIKA SIVARAMAKRISHNA

**A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University
of Manitoba in partial fulfillment of the requirements of the degree
of
DOCTOR OF PHILOSOPHY**

Radhika Sivaramakrishna 1997 (c)

**Permission has been granted to the Library of The University of Manitoba to lend or sell
copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis
and to lend or sell copies of the film, and to Dissertations Abstracts International to publish
an abstract of this thesis/practicum.**

**The author reserves other publication rights, and neither this thesis/practicum nor
extensive extracts from it may be printed or otherwise reproduced without the author's
written permission.**

Abstract

Breast cancer could be cured 99.4% if all tumors are detected by the time they reach 0.4 cm. However, since many normal structures in the breast will be of this magnitude, to detect small tumors we need to align two breast images taken some time apart and subtract them. Hence image registration is important for breast imaging. Image registration consists in identifying corresponding points in two images and interpolating between them. The low contrast and noise in breast images permit few control points.

The "starbyte" textural transformation is introduced and used to solve the breast image registration problem. A textural value at a pixel is obtained as a bit string using binary comparisons of average grey values over different sectors in the local neighborhood of a pixel. Starbyte transformations can be done in several ways, converting breast images to texture maps with clearly demarcable regions from which accurate control points can be easily identified. The performance of the starbyte registration algorithm is evaluated for six different types of distortions. A manual and an automatic algorithm for identification of control points have been presented in detail with several examples. Limited studies with noise and simulated tumors have also been carried out. A complete registration algorithm is presented which consists of first starbyte-transforming the original images, extracting control points, and performing unwarping using thin plate splines.

Mammography has limitations because of its two dimensional nature. Abnormalities are missed primarily because of tissue overlap. We show that direct registration

of longitudinal mammograms cannot give us useful results, demonstrating the need for 3D breast imaging. Starbyte registration of pairs of roughly corresponding slices from 3D breast MRIs shows the advantage of 3D. An important feature of starbyte registration is that it can be easily extended to 3D.

Acknowledgements

I sincerely thank my supervisor, Dr Richard Gordon, without whose constant help, advice and encouragement none of this work would have been possible. Several times during my Ph.D. program, he went out of his way to help me both personally and professionally. He was there with me every step of the way, right from problem formulation to the final stage of thesis writing.

I thank my committee members Dr. M. Pawlak and Dr. E. A. Lyons as well as Dr M. H. Reed, who helped me with numerous ideas and suggestions at various stages of my Ph. D. program. I also thank Dr. M. H. Reed for his financial help during the program.

I sincerely thank my External Examiner, Dr. A. Fenster, for his superlative comments. His valuable suggestions helped to improve the thesis.

I thank Dr. L. Eide, Director of the International Center for Students, who helped me morally and financially during my program.

I thank Sanofi-Winthrop Diagnostic Imaging, Ontario, whose funding helped me to start my research program. I thank Cancer Research Society Inc., Montreal for providing me with a two-year fellowship which enabled me to continue my research. I thank the University of Manitoba for providing me with a Fellowship which enabled me to complete my research program.

I sincerely thank Mr. Mount-first Ng, whose timely and invaluable help, allowed me to produce the final printout of the thesis.

Many of the images in this thesis were provided by courtesy of the National Expert

and Training Centre for Breast Cancer Screening at the Department of Radiology at the University of Nijmegen, the Netherlands. I also thank Drs. D. Hoult and T. Boguslaw and Ms Xiaoman Chen for providing me with the 3D MRI datasets.

I thank Drs. T. Boguslaw and H. Guan for the various discussions we had which helped clarify my many doubts. I also thank the following people who helped answer my questions at various stages of my research program: Drs. A. Rosenfeld, A. Miller, F. L. Bookstein, R. M. Rangayyan, R. M. Nishikawa, R. A. Peters, M. W. Vannier, S. Koscielny, N. Karssemeijer, J. A. Spratt and A. Mazur.

I thank my husband Dr N. S. Shashidhar, who helped me during every stage of my research program. He helped me greatly both personally and in my work. The numerous discussions we had, helped to improve the quality of the thesis. I thank my precious little son, Nandan, who put up cheerfully with my long absences from home and is a model baby. I thank my mother who stayed with us for a long time, just to look after the house and the baby, so that I could complete my research work. Without my husband's, Nandan's and my mother's co-operation and support, I could not have completed my work. I also thank my father, my sister, my nephews, my uncle Col. Dr M. S. Krishnamurthy and my father-in-law Mr N. L. Sathyanarayana, who have always given me their unconditional love, support and encouragement.

Table of Contents

Abstract	i
Acknowledgements	iii
Table of Contents	v
List of Tables	x
List of Figures	xv
Nomenclature	xxxvii
1 INTRODUCTION	1
1.1 Image registration	6
1.1.1 Control point selection	8
1.1.2 Determination of the mapping function	8
1.1.3 Geometric transformation with grey level resampling	11
1.2 Organization of the thesis	12
2 THE STARBYTE TRANSFORMATION	14
2.1 Introduction	14
2.2 The starbyte transformation procedure	15
2.2.1 Size of the neighborhood	20
2.3 Results and Discussion	21
2.3.1 Starbyte-transformed images for a bilinear distortion	21
2.3.2 Starbyte-transformed images for other types of distortions	42

2.3.3	Starbyte-transformed images for a mammogram with a different texture	51
2.4	Conclusions	60
3	DENSITY SLICING AND REGION GROWING	62
3.1	Introduction	62
3.2	Raw Density Slicing	64
3.3	Morphological operations	69
3.3.1	Some basic definitions	69
3.3.2	Dilation	69
3.3.3	Erosion	69
3.3.4	Opening and closing	70
3.4	Density Slices after OC operation	71
3.5	Hamming distance and Hamming neighbors	91
3.6	Region Growing or Pixel aggregation	99
3.7	Conclusions	111
4	MANUALLY-AIDED DETECTION OF CONTROL POINTS	112
4.1	Introduction	112
4.2	Description of UICP	113
4.2.1	Separating concatenated regions	114
4.2.2	Identifying matching regions in the two slices	115
4.3	Bilinear distortion, protocol 3	117
4.3.1	Slice 3	117
4.3.2	Slice 4	120
4.3.3	Slice 6	121
4.3.4	Slice 9	125

4.3.5	Slice 12	129
4.4	Bilinear distortion, protocol 1	134
4.5	Translation	139
4.6	Rotation	144
4.7	Scaling	149
4.8	TPS distortion	153
4.9	Sinusoidal distortion	160
4.10	Iterative UICP	163
4.11	Conclusions	167
5	AUTOMATIC DETECTION OF CONTROL POINTS	168
5.1	Introduction	168
5.2	Description of AUTOCP	168
5.2.1	Acceptance/Rejection criteria	171
5.2.2	Iterative AUTOCP	173
5.3	Bilinear	174
5.4	Translation	178
5.5	Rotation	180
5.6	Scaling	185
5.7	TPS	190
5.8	Sinusoid	194
5.9	Conclusions	199
6	IMAGE REGISTRATION	201
6.1	Introduction	201
6.2	Bilinear distortion	205
6.3	Translation	209

6.4	Rotation	213
6.5	Scaling	217
6.6	TPS distortion	221
6.7	Sinusoidal distortion	225
6.8	Conclusions	232
7	CONTROL POINT ACCURACY WITH ADDED NOISE	234
7.1	Introduction	234
7.2	Bilinear distortion	238
7.3	Sinusoidal distortion	248
7.4	Conclusions	257
8	STARBYTE PERFORMANCE WITH SIMULATED TUMORS	259
8.1	Introduction	259
8.2	Bilinear distortion	260
8.3	Sinusoidal distortion	266
8.4	Conclusions	271
9	REGISTERING REAL IMAGES: THE NEED FOR 3D IMAGING	272
9.1	Introduction	272
9.2	3D breast imaging	273
9.3	Results with approximately corresponding portions of a pair of tempo- ral mammograms	278
9.4	Results with slices of MRI 3D breast data sets	283
9.4.1	Results with the MRI slice pair from the first volunteer	284
9.4.2	Results with the MRI slice pair from the second volunteer . .	290
9.5	Calculation of projection images	294
9.5.1	Results with projection images from the first volunteer	295

9.5.2 Results with projection images from the second volunteer . . .	300
9.6 Conclusions	306
10 CONCLUSIONS AND FUTURE WORK	307
10.1 Conclusions	307
10.2 Future Work	308
10.2.1 Generalization of the starbyte transformation concept	308
10.2.2 Mathematical proof of the commutativity property of the star- byte transformation	309
10.2.3 Alternate methods of registration using the starbyte transfor- mation	309
10.2.4 Use with other types of images	310
10.2.5 Improvements in the automatic procedure	310
10.2.6 More extensive studies with complicated distortions, noise and artificial tumors	310
10.2.7 Use with a parallel machine	312
10.2.8 Improvements in the unwarping procedure	312
10.2.9 Extension to 3D	313
References	315
Appendix	329

List of Tables

3.1	<i>Column 1:</i> Region number. <i>Column 2 and 3:</i> x and y coordinates of centroids of the regions in the target image. <i>Columns 4 and 5:</i> x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). <i>Column 6 and 7:</i> x and y coordinates of centroids of the regions in the reference image. <i>Column 8:</i> D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). <i>Column 9:</i> E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).	102
3.2	<i>Column 1:</i> Region number. <i>Column 2 and 3:</i> x and y coordinates of centroids of the regions in the target image. <i>Columns 4 and 5:</i> x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). <i>Column 6 and 7:</i> x and y coordinates of centroids of the regions in the reference image. <i>Column 8:</i> D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). <i>Column 9:</i> E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).	106

- 3.3 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 108
- 3.4 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 110
- 4.1 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 119

- 4.2 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 121
- 4.3 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 124
- 4.4 *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error). 128

4.5	<i>Column 1:</i> Region number. <i>Column 2 and 3:</i> x and y coordinates of centroids of the regions in the target image. <i>Columns 4 and 5:</i> x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). <i>Column 6 and 7:</i> x and y coordinates of centroids of the regions in the reference image. <i>Column 8:</i> D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). <i>Column 9:</i> E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).	131
4.6	<i>Column 1:</i> Type of distortion. <i>Column 2:</i> N , the number of control points obtained using the user-interactive program. <i>Column 3:</i> Largest error in pixels. <i>Column 4:</i> Distribution of control points over different error (E) ranges measured in pixels.	164
5.1	<i>Column 1:</i> Type of distortion. <i>Column 2:</i> N , the number of control points obtained using AUTOCP. <i>Column 3:</i> Largest error in pixels. <i>Column 4:</i> Distribution of control points over different error (E) ranges measured in pixels.	199
6.1	<i>Column 1:</i> Type of distortion. <i>Column 2:</i> N , the number of control points obtained using UICP (M) and AUTOCP (A). <i>Column 3:</i> G , the root mean square error per pixel between the target and the reference (U), target and the final unwarped images for the manual (M) and automatic (A) case.	230

6.2	<i>Column 1:</i> Type of distortion. <i>Column 2 (Manual): Subcolumn 1:</i> N , the number of control points obtained using UICP. <i>Subcolumn 2:</i> E_{CP} , average geometrical error calculated control points. <i>Subcolumn 3:</i> E_T , average geometrical error calculated over all pixels. <i>Column 3 (Automatic): Subcolumn 1:</i> N , the number of control points obtained using AUTOCP. <i>Subcolumn 2:</i> E_{CP} , average geometrical error calculated over control points. <i>Subcolumn 3:</i> E_T , average geometrical error calculated over all pixels.	231
7.1	<i>Column 1:</i> Noise Level. A and M stand for “Automatic” and “Manual”. <i>Column 2:</i> ST method used. F stands for “Filtered”. <i>Column 3:</i> I , number of iterations. <i>Column 4:</i> Largest error in pixels. <i>Column 4:</i> N , the number of control points obtained using the user-interactive program. <i>Column 5:</i> Break up of control points over different error (E) ranges measured in pixels.	242
7.2	Sinusoidal Distortion: <i>Column 1:</i> Noise Level. A and M stand for “Automatic” and “Manual”. <i>Column 2:</i> ST method used. F stands for “Filtered”. <i>Column 3:</i> I , number of iterations. <i>Column 4:</i> Largest error in pixels. <i>Column 5:</i> N , the number of control points obtained using the user-interactive program. <i>Column 6:</i> Break up of control points over different error (E) ranges measured in pixels.	257
9.1	<i>Column 1:</i> Volunteer number. <i>Column 2:</i> root mean square error per pixel for the slices for the unregistered case (U), using UICP (M) and AUTOCP (A). <i>Column 3:</i> root mean square error per pixel for the projections for the unregistered case (U), using UICP (M) and AUTOCP (A).	305

List of Figures

2.1	<i>a</i> : Continuous space origin of the concept of starbyte patterns. Each sector of a neighborhood is assigned a bit 1 or 0 according to whether its average density is above or below the density at the center of the circle, or above or below the average over the circle. <i>b</i> : 3×3 neighborhood around point <i>P</i> (labeled #8). <i>c</i> – <i>f</i> : Sectors used for protocols 1 to 4 (See equations for protocols 1 to 4).	17
2.2	128×128 blocks cropped out a mammogram with and without microcalcifications respectively.	21
2.3	Figures 2.2a and b, warped using the same bilinear transformation. .	22
2.4	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 3×3 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 3×3 neighborhood.	26
2.5	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 3×3 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 3×3 neighborhood.	27
2.6	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 9×9 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 9×9 neighborhood.	28
2.7	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 9×9 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 9×9 neighborhood.	29

2.8	<i>a – h</i> : Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 15×15 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 15×15 neighborhood. . .	30
2.9	<i>a – h</i> : Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 15×15 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 15×15 neighborhood. . .	31
2.10	<i>a – h</i> : Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for an adaptive neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for an adaptive neighborhood. .	32
2.11	<i>a – h</i> : Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for an adaptive neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for an adaptive neighborhood. .	33
2.12	<i>a to d</i> : Distribution of the neighborhood sizes shown as a bar chart for Figures 2.2a and b and Figures 2.3a and b respectively.	35
2.13	<i>a</i> and <i>b</i> are the same as Figure 2.6k and Figure 2.7k. <i>c</i> is the result of applying the same bilinear warp on <i>a</i> as was applied Figures 2.2 (to generate Figures 2.3). <i>d</i> shows the absolute difference between <i>a</i> and <i>b</i> , while <i>e</i> shows the absolute difference between <i>b</i> and <i>c</i>	37
2.14	<i>a</i> and <i>b</i> are the same as Figure 2.6i and Figure 2.7i. <i>c</i> is the result of applying the same bilinear warp on <i>a</i> as was applied Figures 2.2 (to generate Figures 2.3). <i>d</i> shows the absolute difference between <i>a</i> and <i>b</i> , while <i>e</i> shows the absolute difference between <i>b</i> and <i>c</i>	40
2.15	<i>a</i> is the same as Figure 2.2b. <i>b</i> : <i>a</i> translated by 8 pixels upwards and 10 pixels leftwards. <i>c</i> : <i>a</i> rotated by 10 deg anti-clockwise. <i>d</i> : <i>a</i> enlarged by 10 %.	43

2.16	<i>a – h</i> : Starbyte transformed images of Figure 2.15b (8 pixels upward- and 10 pixels leftward-translated version of Figure 2.15a), using protocols 1 to 8, for a 9×9 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.15b, using protocols 1 to 8, for a 15×15 neighborhood.	44
2.17	<i>a – h</i> : Starbyte transformed images of Figure 2.15c (10 deg anti-clockwise rotated version of Figure 2.15a), using protocols 1 to 8, for a 9×9 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.15c, using protocols 1 to 8, for a 15×15 neighborhood.	45
2.18	<i>a – h</i> : Starbyte transformed images of Figure 2.15d (10% expanded version of Figure 2.15a), using protocols 1 to 8, for a 9×9 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.15d, using protocols 1 to 8, for a 15×15 neighborhood.	46
2.19	<i>a</i> : Same as Figure 2.2b. <i>b</i> : Distorting <i>a</i> using thin plate splines (map of corresponding points given in Figure 2.20a). <i>c</i> : Distorting <i>a</i> using the sinusoidal distortion given by equation (2.2).	47
2.20	Map of corresponding points for the distortion between Figures 2.19a and b. ‘+’ denotes points in the reference image. ‘o’ denotes points in the target image.	48
2.21	<i>a – h</i> : Starbyte transformed images of Figure 2.19b (TPS-distorted version of Figure 2.2b), using protocols 1 to 8, for a 9×9 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.19b, using protocols 1 to 8, for a 15×15 neighborhood.	49
2.22	<i>a – h</i> : Starbyte transformed images of Figure 2.19c (sinusoidally-distorted version of Figure 2.2b), using protocols 1 to 8, for a 9×9 neighborhood. <i>i – p</i> : Starbyte transformed images of Figure 2.19c, using protocols 1 to 8, for a 15×15 neighborhood.	50

2.23	Distortions used: <i>a</i> : Bilinear given by equation (2.1). <i>b</i> : Translation (8 pixels upwards and 10 pixels leftwards). <i>c</i> : Rotation (10 deg counter-clockwise). <i>d</i> : Scaling (10 % enlargement). <i>e</i> : TPS distortion. <i>f</i> : Sinusoidal distortion given by equation (2.2).	52
2.24	<i>a</i> and <i>b</i> : 128 × 128 regions cropped out of the image “c12c.ima” in [86] with and without microcalcifications. <i>c</i> and <i>d</i> : <i>a</i> and <i>b</i> warped using equation (2.1). <i>e</i> and <i>f</i> : <i>a</i> and <i>b</i> warped using the transformation depicted in Figure 2.20c.	53
2.25	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24a, using protocols 1 to 8, for a 9 × 9 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24b, using protocols 1 to 8, for a 9 × 9 neighborhood.	54
2.26	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24c, using protocols 1 to 8, for a 9 × 9 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24d, using protocols 1 to 8, for a 9 × 9 neighborhood.	55
2.27	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24e, using protocols 1 to 8, for a 9 × 9 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24f, using protocols 1 to 8, for a 9 × 9 neighborhood.	56
2.28	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24a, using protocols 1 to 8, for a 15 × 15 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24b, using protocols 1 to 8, for a 15 × 15 neighborhood.	57
2.29	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24c, using protocols 1 to 8, for a 15 × 15 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24d, using protocols 1 to 8, for a 15 × 15 neighborhood.	58
2.30	<i>a</i> – <i>h</i> : Starbyte transformed images of Figure 2.24e, using protocols 1 to 8, for a 15 × 15 neighborhood. <i>i</i> – <i>p</i> : Starbyte transformed images of Figure 2.24f, using protocols 1 to 8, for a 15 × 15 neighborhood.	59

3.1	<i>a</i> and <i>b</i> : Starbyte images of Figure 2.2b and Figure 2.3b obtained by applying protocol 3 with a 9×9 neighborhood size.	65
3.2	<i>a</i> to <i>p</i> : Figure 3.1a density sliced at the levels from 0 to 15.	66
3.3	<i>a</i> to <i>p</i> : Figure 3.1b density sliced at the levels from 0 to 15.	67
3.4	<i>a</i> to <i>p</i> : Density slices in Figure 3.2 after OC operation with a “plus” SE.	72
3.5	<i>a</i> to <i>p</i> : Density slices in Figure 3.3 after OC operation with a “plus” SE.	73
3.6	<i>a</i> to <i>p</i> : Density slices in Figure 3.2 after OC operation with a 3×3 SE.	74
3.7	<i>a</i> to <i>p</i> : Density slices in Figure 3.3 after OC operation with a 3×3 SE.	75
3.8	<i>a</i> to <i>p</i> : Density slices in Figure 3.2 after OC operation with a 5×5 SE.	76
3.9	<i>a</i> to <i>p</i> : Density slices in Figure 3.3 after OC operation with a 5×5 SE.	77
3.10	<i>a</i> and <i>b</i> : Starbyte images of Figure 2.2b and Figure 2.3b obtained by applying protocol 1 with a 9×9 neighborhood size.	78
3.11	<i>a</i> to <i>p</i> : Density slices in Figure 3.10a after OC operation with a “plus” SE.	79
3.12	<i>a</i> to <i>p</i> : Density slices in Figure 3.10b after OC operation with a “plus” SE.	80
3.13	<i>a – d</i> : Starbyte images of Figures 2.15a to d of the previous chapter using the 3rd protocol on 9×9 neighborhoods. Figures 2.15b to d were obtained by translating (8 pixels upwards and 10 pixels leftwards), rotating (10 deg) and scaling (10 %) Figure 2.15a.	81
3.14	<i>a</i> to <i>p</i> : Density slices of Figure 3.13b after OC operation with a “plus” SE.	82
3.15	<i>a</i> to <i>p</i> : Density slices of Figure 3.13c after OC operation with a “plus” SE.	83
3.16	<i>a</i> to <i>p</i> : Density slices of Figure 3.13d after OC operation with a “plus” SE.	84

3.17 <i>a</i> and <i>b</i> : Starbyte images of Figure 2.24a and its TPS-distorted version obtained by applying protocol 1 with a 9×9 neighborhood size. . . .	85
3.18 <i>a</i> to <i>p</i> : Density slices of Figure 3.17a after OC operation with a “plus” SE.	86
3.19 <i>a</i> to <i>p</i> : Density slices of Figure 3.17b after OC operation with a “plus” SE.	87
3.20 <i>a</i> and <i>b</i> : Starbyte images of Figure 2.2a and its sinusoidally-distorted version (using equation (2.2) obtained by applying protocol 1 with a 9 $\times 9$ neighborhood size.	88
3.21 <i>a</i> to <i>p</i> : Density slices of Figure 3.20a after OC operation with a “plus” SE.	89
3.22 <i>a</i> to <i>p</i> : Density slices of Figure 3.20b after OC operation with a “plus” SE.	90
3.23 <i>a</i> to <i>d</i> : Hamming neighbors at distances 1, 2, 3 and 4 for the 16 density levels represented as hexadecimal numbers (0 to 9, “A” to “F”). . . .	93
3.24 <i>a</i> to <i>p</i> : Density slices of Figure 3.1a combined with their Hamming neighbors at distance 1.	95
3.25 <i>a</i> to <i>p</i> : Density slices of Figure 3.1b combined with their Hamming neighbors at distance 1.	96
3.26 <i>a</i> to <i>p</i> : Density slices of Figure 3.1a combined with their Hamming neighbors at distance 1 after an OC operation with a “plus” SE. . . .	97
3.27 <i>a</i> to <i>p</i> : Density slices of Figure 3.1b combined with their Hamming neighbors at distance 1 after an OC operation with a “plus” SE. . . .	98
3.28 <i>a</i> : Figure 3.5d shown with 23 prominent regions numbered. <i>b</i> : Figure 3.4d with corresponding regions in <i>a</i> shown. Fragmented regions are iden- tified by drawing a boundary around the fragments.	103

3.29	<i>a</i> : Figure 3.28 shown after extraneous portion in region 22 was removed. <i>b</i> : Same as Figure 3.28	104
3.30	<i>a</i> : Figure 3.5g shown with 14 prominent regions numbered. <i>b</i> : Figure 3.4g with corresponding regions in <i>a</i> shown. Fragmented regions are identified by drawing a boundary around the fragments.	105
3.31	<i>a</i> : Figure 3.5j shown with 14 prominent regions numbered. <i>b</i> : Figure 3.4j with corresponding regions in <i>a</i> shown. Fragmented regions are identified by drawing a boundary around the fragments.	107
3.32	<i>a</i> : Figure 3.5m shown with 15 prominent regions numbered. <i>b</i> : Figure 3.4m with corresponding regions in <i>a</i> shown. Fragmented regions are identified by drawing a boundary around the fragments.	109
4.1	<i>a</i> and <i>b</i> : 26 regions obtained using UICP, shown for slice 3 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. <i>a</i> was obtained by modifying Figure 3.28a using UICP, while <i>b</i> is the same as Figure 3.28b.	117
4.2	<i>a</i> : Same as Figure 3.28a. <i>b</i> : <i>a</i> modified using seven region separation operations using UICP. The seven lines of separations have also been drawn for clarity.	118
4.3	<i>a</i> and <i>b</i> : 9 regions obtained using UICP, shown for slice 4 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b.	120
4.4	<i>a</i> and <i>b</i> : 18 regions obtained using UICP, shown for slice 6 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. <i>a</i> and <i>b</i> were obtained by modifying Figures 3.30a and b respectively using UICP.	122

4.5	<i>a</i> : Same as Figure 3.30a. <i>b</i> : <i>a</i> modified using three region separation operations using UICP. The three lines of separations have also been drawn for clarity.	123
4.6	<i>a</i> : Same as Figure 3.30b. <i>b</i> : <i>a</i> modified using two region separation operations using UICP. The two lines of separations have also been drawn for clarity.	123
4.7	<i>a</i> and <i>b</i> : 18 regions obtained using UICP, shown for slice 9 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. <i>a</i> and <i>b</i> were obtained by modifying Figures 3.31a and b respectively using UICP.	125
4.8	<i>a</i> : Same as Figure 3.31a. <i>b</i> : <i>a</i> modified using one region separation operation using UICP. The single line of separation has also been drawn for clarity.	127
4.9	<i>a</i> : Same as Figure 3.31b. <i>b</i> : <i>a</i> modified using three region separation operations using UICP. The three lines of separations have also been drawn for clarity.	127
4.10	<i>a</i> and <i>b</i> : 18 regions obtained using UICP, shown for slice 12 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. <i>a</i> was obtained by modifying Figure 3.32a using UICP, while <i>b</i> is the same as Figure 3.32b.	130
4.11	<i>a</i> : Same as Figure 3.32a. <i>b</i> : <i>a</i> modified using three region separation operations using UICP. The three lines of separation have also been drawn for clarity.	130
4.12	<i>a</i> and <i>b</i> : D and E for the case where the two original images (Figure 2.2b and Figure 2.3b) are related by equation (2.1), and the starbyte transformation was used with protocol 3.	132

4.13 <i>a</i> : Bilinear distortion shown as a grid diagram (equation (2.1)). <i>b</i> : Registration achieved after using UICP.	133
4.14 <i>a</i> and <i>b</i> : 21 regions obtained using UICP, shown for slice 3 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12d and Figure 3.11d using UICP.	136
4.15 <i>a</i> and <i>b</i> : 12 regions obtained using UICP, shown for slice 5 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12f and Figure 3.11f using UICP.	136
4.16 <i>a</i> and <i>b</i> : 24 regions obtained using UICP, shown for slice 10 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12k and Figure 3.11k using UICP.	137
4.17 <i>a</i> and <i>b</i> : 17 regions obtained using UICP, shown for slice 10 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12m and Figure 3.11m using UICP.	137
4.18 <i>a</i> and <i>b</i> : D and E for the case where the two original images (Figure 2.2b and Figure 2.3b) are related by equation (2.1), and the starbyte trans- formation was used with protocol 1.	138
4.19 <i>a</i> : Bilinear distortion shown as a grid diagram (equation (2.1)). <i>b</i> : Registration achieved after using UICP with protocol 1.	138
4.20 <i>a</i> and <i>b</i> : Figure 3.14d and Figure 3.4d with 17 regions marked.	141
4.21 <i>a</i> and <i>b</i> : Figure 3.14g and Figure 3.4g with 17 regions marked.	141
4.22 <i>a</i> and <i>b</i> : Figure 3.14j and Figure 3.4j with 20 regions marked.	142

4.23 <i>a</i> and <i>b</i> : Figure 3.14m and Figure 3.4m with 16 regions marked. . . .	142
4.24 <i>a</i> and <i>b</i> : D and E for the original images (Figures 2.15a and b) related by translation (8 pixels upwards and 10 pixels leftwards).	143
4.25 <i>a</i> : Translation (8 pixels upwards and 10 pixels leftwards) shown as a grid diagram. <i>b</i> : Registration achieved after using UICP.	143
4.26 <i>a</i> and <i>b</i> : Figure 3.15d and Figure 3.4d with 12 regions marked after being modified by UICP.	146
4.27 <i>a</i> and <i>b</i> : Figure 3.15g and Figure 3.4g with 12 regions marked after being modified by UICP.	146
4.28 <i>a</i> and <i>b</i> : Figure 3.15j and Figure 3.4j with 20 regions marked after being modified by UICP.	147
4.29 <i>a</i> and <i>b</i> : Figure 3.15m and Figure 3.4m with 16 regions marked after being modified by UICP.	147
4.30 <i>a</i> and <i>b</i> : D and E for the original images (Figures 2.15a and c) related by rotation (10 deg counter-clockwise).	148
4.31 <i>a</i> : Rotation (10 deg counter-clockwise) shown as a grid diagram. <i>b</i> : Registration achieved after using UICP.	148
4.32 <i>a</i> and <i>b</i> : Figure 3.16b and Figure 3.4b with 10 regions marked. . . .	150
4.33 <i>a</i> and <i>b</i> : Figure 3.16d and Figure 3.4d with 24 regions marked. . . .	150
4.34 <i>a</i> and <i>b</i> : Figure 3.16g and Figure 3.4g with 22 regions marked after being modified by UICP.	151
4.35 <i>a</i> and <i>b</i> : Figure 3.16j and Figure 3.4j (after modification with UICP) with 18 regions marked.	151
4.36 <i>a</i> and <i>b</i> : Figure 3.16m (after modification with UICP) and Figure 3.4m with 11 regions marked.	152

4.37 <i>a</i> and <i>b</i> : <i>D</i> and <i>E</i> for the original images (Figures 2.15a and c) related by a 10 % enlargement.	152
4.38 <i>a</i> : Scaling (10 % enlargement) shown as a grid diagram. <i>b</i> : Registration achieved after using UICP.	153
4.39 <i>a</i> and <i>b</i> : Figure 3.19c and Figure 3.18c with 10 regions marked. . . .	155
4.40 <i>a</i> and <i>b</i> : Figure 3.19d and Figure 3.18d with 21 regions marked. . . .	155
4.41 <i>a</i> and <i>b</i> : Figure 3.19f and Figure 3.18f with 10 regions marked after being modified by UICP.	156
4.42 <i>a</i> and <i>b</i> : Figure 3.19i and Figure 3.18i (after modification with UICP) with 14 regions marked.	156
4.43 <i>a</i> and <i>b</i> : Figure 3.19k and Figure 3.18k with 31 regions marked after being modified by UICP.	157
4.44 <i>a</i> and <i>b</i> : Figure 3.19m (after modification with UICP) and Figure 3.18m with 24 regions marked.	157
4.45 <i>a</i> and <i>b</i> : Figure 3.19n and Figure 3.18n with 10 regions marked. . . .	158
4.46 <i>a</i> and <i>b</i> : <i>D</i> and <i>E</i> for the original images (Figure 2.24a and its TPS-distorted version using the distortion depicted in Figure 2.23e.	158
4.47 <i>a</i> : TPS distortion shown as a grid diagram. <i>b</i> : Registration achieved after using UICP. <i>c</i> : Registration achieved after using UICP with boundary condition.	159
4.48 <i>a</i> and <i>b</i> : Figure 3.22d and Figure 3.21d with 20 regions marked after being modified by UICP.	161
4.49 <i>a</i> and <i>b</i> : Figure 3.22f and Figure 3.21f with 20 regions marked. . . .	161
4.50 <i>a</i> and <i>b</i> : Figure 3.22k and Figure 3.21k with 18 regions marked after being modified by UICP.	162

4.51	<i>a</i> and <i>b</i> : Figure 3.22m and Figure 3.21m with 12 regions marked after being modified by UICP.	162
4.52	<i>a</i> and <i>b</i> : D and E for the original images (Figure 2.24a and its sinusoidally-distorted version using equation (2.2)).	163
4.53	<i>a</i> to <i>c</i> : Cleaned-up STs of the original, intermediate and target images for the sinusoidal distortion.	165
4.54	<i>a</i> and <i>b</i> : D and E for the intermediate and target images for the sinusoidal distortion.	165
4.55	<i>a</i> : Sinusoidal distortion shown as a grid diagram (equation (2.2)). <i>b</i> : Registration achieved after the first pass of UICP. <i>c</i> : Registration achieved after the second pass of UICP. <i>d</i> : Registration achieved after the second pass of UICP with boundary condition.	166
5.1	<i>a</i> and <i>b</i> : D and E for the reference and target images for the bilinear distortion.	174
5.2	<i>a</i> to <i>c</i> : Cleaned-up STs of the original, intermediate and target images for the bilinear distortion.	176
5.3	<i>a</i> and <i>b</i> : D and E for the intermediate and target images for the bilinear distortion.	177
5.4	<i>a</i> : Bilinear distortion shown as a grid diagram (equation (2.1)). <i>b</i> : Registration achieved after the first pass of AUTOCP. <i>c</i> : Registration achieved after the second pass of AUTOCP.	177
5.5	<i>a</i> and <i>b</i> : D and E for the reference and target images for translation.	179
5.6	<i>a</i> : Translation shown as a grid diagram (equation (2.1)). <i>b</i> : Registration achieved after one run of AUTOCP.	179
5.7	<i>a</i> and <i>b</i> : D and E for the reference and target images for 10 deg counter-clockwise rotation.	181

5.8	<i>a</i> to <i>c</i> : Cleaned-up STs of the original, intermediate and target images for rotation.	182
5.9	<i>a</i> and <i>b</i> : D and E for the intermediate and target images for rotation.	183
5.10	<i>a</i> : 10 deg counter-clockwise rotation shown as a grid diagram. <i>b</i> : Registration achieved after the first pass of AUTOCP. <i>c</i> : Registration achieved after the second pass of AUTOCP.	184
5.11	<i>a</i> and <i>b</i> : D and E for the reference and target images for 10 % enlargement.	186
5.12	<i>a</i> to <i>c</i> : Cleaned-up STs of the original, intermediate and target images for scaling.	187
5.13	<i>a</i> and <i>b</i> : D and E for the intermediate and target images for scaling.	188
5.14	<i>a</i> : 10 % enlargement shown as a grid diagram. <i>b</i> : Registration achieved after the first pass of AUTOCP. <i>c</i> : Registration achieved after the second pass of AUTOCP.	189
5.15	<i>a</i> to <i>c</i> : Cleaned-up STs of the original, intermediate and target images for the TPS distortion.	191
5.16	<i>a</i> and <i>b</i> : D and E for the reference and target images for the TPS distortion.	192
5.17	<i>a</i> and <i>b</i> : D and E for the intermediate and target images for the TPS distortion.	192
5.18	<i>a</i> : TPS distortion shown as a grid diagram. <i>b</i> : Registration achieved after the first pass of AUTOCP. <i>c</i> : Registration achieved after the second pass of AUTOCP. <i>d</i> : Registration achieved after the second pass of AUTOCP after imposing boundary condition.	193
5.19	<i>a</i> and <i>b</i> : D and E for the reference and target images for the sinusoidal distortion.	195

5.20	<i>a</i> to <i>d</i> : Cleaned-up STs of the original, first intermediate, second intermediate and target images for the sinusoidal distortion.	196
5.21	<i>a</i> and <i>b</i> : D and E for the first intermediate and target images for the sinusoidal distortion.	197
5.22	<i>a</i> and <i>b</i> : D and E for the second intermediate and target images for the sinusoidal distortion.	197
5.23	<i>a</i> : Sinusoidal distortion shown as a grid diagram. <i>b</i> : Registration achieved after the first pass of AUTOCP. <i>c</i> : Registration achieved after the second pass of AUTOCP. <i>d</i> : Registration achieved after the third pass of AUTOCP. <i>e</i> : Registration achieved after the third pass of AUTOCP after imposing boundary condition.	198
6.1	Flow chart showing various stages of the starbyte image registration algorithm using UICP.	203
6.2	Flow chart showing various stages of the starbyte image registration algorithm using AUTOCP.	204
6.3	<i>a</i> to <i>d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for the bilinear distortion.	207
6.4	<i>a</i> to <i>c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the bilinear distortion. The grey scale bar below shows the range of values in the difference images.	208
6.5	<i>a</i> : Bilinear distortion shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	209
6.6	<i>a</i> to <i>d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for translation (8 pixels upwards and 10 pixels leftwards).	211

6.7	<i>a to c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for translation (8 pixels upwards and 10 pixels leftwards). The grey scale bar below shows the range of values in the difference images.	212
6.8	<i>a</i> : Translation (8 pixels upwards and 10 pixels leftwards) shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	213
6.9	<i>a to d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for rotation (10 deg counter-clockwise). .	215
6.10	<i>a to c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for rotation (10 deg counter-clockwise). The grey scale bar below shows the range of values in the difference images.	216
6.11	<i>a</i> : 10 deg counter-clockwise rotation shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	217
6.12	<i>a to d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for scaling (10 % enlargement).	219
6.13	<i>a to c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for scaling (10 % enlargement). The grey scale bar below shows the range of values in the difference images.	220
6.14	<i>a</i> : 10 % enlargement shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	221
6.15	<i>a to d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for the TPS-distortion.	223

6.16	<i>a</i> to <i>c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the TPS-distortion. The grey scale bar below shows the range of values in the difference images.	224
6.17	<i>a</i> : TPS-distortion shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	225
6.18	<i>a</i> to <i>d</i> : Reference, target and unwarped images obtained using UICP and AUTOCP respectively for the sinusoidal distortion.	227
6.19	<i>a</i> to <i>c</i> : Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the sinusoidal distortion. The grey scale bar below shows the range of values in the difference images.	228
6.20	<i>a</i> : Sinusoidal distortion shown as a grid diagram. <i>b</i> : Registration achieved with UICP. <i>c</i> : Registration achieved with AUTOCP.	229
7.1	<i>a</i> and <i>b</i> : Reference and target images for the bilinear distortion. <i>c</i> and <i>d</i> : <i>a</i> and <i>b</i> with 1% noise. <i>e</i> and <i>f</i> : <i>a</i> and <i>b</i> with 5 % noise. <i>g</i> and <i>h</i> : <i>a</i> and <i>b</i> with 10% noise.	239
7.2	<i>a</i> to <i>d</i> : 9 x 9 and 15 x 15 STs of the reference-target pair for the bilinear distortion. <i>e</i> and <i>f</i> : 9 x 9 STs of filtered reference and target images. <i>g</i> to <i>l</i> : Corresponding ST images for the reference-target pair with 1% noise. <i>m</i> to <i>r</i> : Corresponding ST images for the reference-target pair with 5% noise. <i>s</i> to <i>x</i> : Corresponding ST images for the reference-target pair with 10% noise.	240
7.3	<i>a</i> and <i>b</i> : D and E for 1 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.	241

7.4	<i>a</i> and <i>b</i> : D and E for 1 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.	241
7.5	<i>a</i> and <i>b</i> : D and E for 1 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.	241
7.6	<i>a</i> and <i>b</i> : D and E for 5 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.	245
7.7	<i>a</i> and <i>b</i> : D and E for 5 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.	245
7.8	<i>a</i> and <i>b</i> : D and E for 5 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.	245
7.9	<i>a</i> and <i>b</i> : D and E for 10 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.	246
7.10	<i>a</i> and <i>b</i> : D and E for 10 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.	246
7.11	<i>a</i> and <i>b</i> : D and E for 10 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.	246
7.12	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.	247
7.13	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.	247
7.14	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.	247
7.15	<i>a</i> and <i>b</i> : Reference and target images for the sinusoidal distortion. <i>c</i> and <i>d</i> : <i>a</i> and <i>b</i> with 1 % noise. <i>e</i> and <i>f</i> : <i>a</i> and <i>b</i> with 5 % noise. <i>g</i> and <i>h</i> : <i>a</i> and <i>b</i> with 10% noise.	249

7.16 <i>a</i> to <i>d</i> : 9 x 9 and 15 x 15 STs of the reference-target pair for the sinusoidal distortion. <i>e</i> and <i>f</i> : 9 x 9 STs of filtered reference and target images. <i>g</i> to <i>l</i> : Corresponding ST images for the reference-target pair with 1 % noise. <i>m</i> to <i>r</i> : Corresponding ST images for the reference-target pair with 5 % noise. <i>s</i> to <i>x</i> : Corresponding ST images for the reference-target pair with 10 % noise.	250
7.17 <i>a</i> to <i>c</i> : Variation of the number of control points with iteration number for the sinusoidal distortion with 5 % noise using the 9 x 9, 15 x 15, and filtered 9 x 9 ST.	251
7.18 <i>a</i> and <i>b</i> : D and E for 1 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.	253
7.19 <i>a</i> and <i>b</i> : D and E for 1 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.	253
7.20 <i>a</i> and <i>b</i> : D and E for 1 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.	253
7.21 <i>a</i> and <i>b</i> : D and E for 5 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.	254
7.22 <i>a</i> and <i>b</i> : D and E for 5 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.	254
7.23 <i>a</i> and <i>b</i> : D and E for 5 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.	254
7.24 <i>a</i> and <i>b</i> : D and E for 10 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.	255
7.25 <i>a</i> and <i>b</i> : D and E for 10 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.	255

7.26	<i>a</i> and <i>b</i> : D and E for 10 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.	255
7.27	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.	256
7.28	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.	256
7.29	<i>a</i> and <i>b</i> : D and E obtained using UICP for 10 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.	256
8.1	<i>a</i> and <i>b</i> : Reference and target images for the bilinear distortion. <i>c</i> : <i>b</i> with tumor image added. <i>d</i> and <i>e</i> : Unwarped images obtained using UICP and AUTOCP.	261
8.2	<i>a</i> and <i>b</i> : STs of the reference and target images for the bilinear distortion using a 9 x 9 neighborhood size.	262
8.3	<i>a</i> and <i>b</i> : D and E vector diagrams for the bilinear distortion obtained using UICP, with 4 simulated tumors present in the target image. . .	263
8.4	<i>a</i> and <i>b</i> : D and E vector diagrams for the bilinear distortion obtained using AUTOCP, with 4 simulated tumors present in the target image.	263
8.5	<i>a</i> : Absolute difference image between the reference and target for the bilinear distortion. <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases. <i>d</i> : Actual tumor image for the bilinear distortion.	265
8.6	<i>a</i> and <i>b</i> : Reference and target images for the sinusoidal distortion. <i>c</i> : <i>b</i> with tumor image added. <i>d</i> and <i>e</i> : Unwarped images obtained using UICP and AUTOCP.	266
8.7	<i>a</i> and <i>b</i> : STs of the reference and target images for the sinusoidal distortion using a 9 x 9 neighborhood size.	267

8.8	<i>a</i> and <i>b</i> : D and E vector diagrams for the sinusoidal distortion obtained using UICP, with 4 simulated tumors present in the target image. . .	268
8.9	<i>a</i> and <i>b</i> : D and E vector diagrams for the sinusoidal distortion obtained using AUTOCP, with 4 simulated tumors present in the target image.	268
8.10	<i>a</i> : Absolute difference image between the reference and target for the sinusoidal distortion. <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases. <i>d</i> : Actual tumor image for the sinusoidal distortion.	270
9.1	<i>a</i> and <i>b</i> : Portions of actual temporal mammograms of the same patient. <i>c</i> and <i>d</i> : Unwarped images obtained with and without the starbyte transformation.	279
9.2	<i>a</i> and <i>b</i> : STs of the original images.	280
9.3	<i>a</i> and <i>b</i> : Reference and target images with 25 control points obtained using UICP superimposed.	281
9.4	<i>a</i> and <i>b</i> : Reference and target images with 14 control points obtained directly from the original images, superimposed.	281
9.5	<i>a</i> : Absolute difference image between the reference and target. <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images obtained with and without the starbyte transformation. Circles on <i>b</i> and <i>c</i> show the location where microcalcifications are present in the target. Darker regions in the difference image correspond to lower difference values.	282
9.6	<i>a</i> and <i>b</i> : Corresponding slices taken from 3D MRI data sets of first volunteer. <i>c</i> and <i>d</i> : Unwarped images obtained with UICP and AUTOCP.	285

9.7	<i>a</i> and <i>b</i> : STs of the MRI slices from the first volunteer using a 9 x 9 neighborhood size. <i>c</i> and <i>d</i> : STs of filtered reference and target images using a 15 x 15 neighborhood size. The echo is apparent.	286
9.8	<i>a</i> and <i>b</i> : Reference and target images (corresponding MRI slices of first volunteer) with 17 control points obtained using UICP superimposed.	288
9.9	<i>a</i> and <i>b</i> : Reference and target images (corresponding MRI slices of first volunteer) with 6 control points obtained using AUTOCP superimposed.	288
9.10	<i>a</i> : Absolute difference image between the reference and target for the MRI slice pair from the first volunteer. <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases.	289
9.11	<i>a</i> and <i>b</i> : Corresponding slices taken from 3D MRI data sets of second volunteer. <i>c</i> and <i>d</i> : Unwarped images obtained with UICP and AUTOCP.	290
9.12	<i>a</i> and <i>b</i> : STs of the MRI slices from the second volunteer using a 9 x 9 neighborhood size. <i>c</i> and <i>d</i> : STs of filtered slices using a 15 x 15 neighborhood size.	291
9.13	<i>a</i> and <i>b</i> : Reference and target images (corresponding MRI slices of second volunteer) with 16 control points obtained using UICP superimposed.	292
9.14	<i>a</i> and <i>b</i> : Reference and target images (corresponding MRI slices of second volunteer) with 5 control points obtained using AUTOCP superimposed.	292
9.15	<i>a</i> : Absolute difference image between the reference and target for the MRI slice pair from the second volunteer. <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases.	293

9.16 <i>a</i> and <i>b</i> : Projection images for the first volunteer. <i>c</i> and <i>d</i> : Unwarped images obtained with UICP and AUTOCP.	295
9.17 <i>a</i> and <i>b</i> : STs of the projections for the first volunteer using a 9 x 9 neighborhood size. <i>c</i> and <i>d</i> : STs of filtered projections using a 15 x 15 neighborhood size. Echos are again apparent.	296
9.18 <i>a</i> and <i>b</i> : Projection images of first volunteer with 11 control points obtained using UICP superimposed.	298
9.19 <i>a</i> and <i>b</i> : Projection images of first volunteer with 3 control points obtained using AUTOCP superimposed.	298
9.20 <i>a</i> : Absolute difference image between the reference and target image (projection images for the first volunteer). <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases.	299
9.21 <i>a</i> and <i>b</i> : Projection images for the second volunteer. <i>c</i> and <i>d</i> : Unwarped images obtained with UICP and AUTOCP.	300
9.22 <i>a</i> and <i>b</i> : STs of the projections for the second volunteer using a 9 x 9 neighborhood size. <i>c</i> and <i>d</i> : STs of filtered projections using a 15 x 15 neighborhood size.	301
9.23 <i>a</i> and <i>b</i> : Projection images of second volunteer with 13 control points obtained using UICP superimposed.	303
9.24 <i>a</i> and <i>b</i> : Projection images of second volunteer with 6 control points obtained using AUTOCP superimposed.	303
9.25 <i>a</i> : Absolute difference image between the reference and target image (projection images for the second volunteer). <i>b</i> and <i>c</i> : Absolute difference images between the target and unwarped images for the manual and automatic cases.	304

Nomenclature

AUTOCP	Automatic program to extract control points from the starbyte-transformed images
CT	Computed tomography
DFT	Discrete Fourier transform
EIT	Electical impedance tomography
FLASH	Fast low angle single shot
IBD	Institute of Biodiagnostics
MRI	Magnetic resonance imaging
NRC	National research council
OC	Open and close morphological operation
PET	Positron emission tomography
SE	Structuring element
ST	Starbyte transformation
STs	Starbyte-transformed images
TPS	Thin plate spline
TPSs	Thin plate splines
UICP	User-interactive program to extract control points from the starbyte-transformed images

Chapter 1

INTRODUCTION

Currently one in nine women in the United States [121] and one in ten women in Canada can be expected to develop breast cancer in their lifetime [15]. Since the means to prevent breast cancer have not yet been found, early detection is very important [112]. Mammography is presently the first-line imaging technique for the detection and diagnosis of breast lesions [66], [75].

Radiologists inspect individual mammograms and look for characteristic signs of potential cancer like microcalcifications, masses, and architectural distortions. Radiologists also compare two mammograms taken at different screenings, so as to see if any abnormality has developed in the time interval between the screens.

However, mammographers miss about 10% of all cancers [4], [120], [83]. Also the overall yield of breast cancers per number of breast biopsies recommended on the basis of screening mammograms ranges between roughly 10% to 50% [103], [120]. X-ray mammography is especially ineffective in detecting cancers in radiographically dense breasts [59].

Mammograms are obtained by applying compression to the breast in order to equalize the optical path length, using two plates parallel to the imaging plane. The limitations of mammography are mainly because three-dimensional structure in the breast is projected onto a two-dimensional plane. Breast cancers are missed because of the presence of intervening tissue above and below an abnormality which make visibility poor. This intervening tissue, which occludes the visibility of an abnormality

creates “structural noise” in the images. The large amount of structural noise makes it difficult for a radiologist to detect an abnormality [96]. The lack of contrast and the noisy nature of the images compounds the problem.

The only way in which the tumor can be visualized clearly without the superimposition of tissue is to image the breast in 3D. Possible 3D imaging methods include computed tomography (CT), magnetic resonance imaging (MRI), breast sonography and electrical impedance tomography (EIT). While none of these procedures have been able to replace mammography for now, the need for 3D imaging of the breast is being recognized by many researchers [95], [19], [103].

When 3D imaging becomes a reality, radiologists will continue to study individual 3D images of the breast to look for signs of abnormality. However radiologists will also compare two 3D longitudinal images of the same patient to check for the growth of any abnormality in the time interval. We have shown [112] that the breast cancer problem can be “cured” with a 99.4% success rate if all tumors of size 0.4 cm are detected by the time they reach that size. Given the fact that many of the structures found within the breast will be of this magnitude (apart from the small tumors we are looking for), the best way to view any changes in the two 3D images would be to subtract out the structural noise in the two images.

However, two longitudinal 3D breast images would be almost guaranteed to be misregistered because the loose and non-rigid tissue of a woman’s breast simply cannot be placed in the imaging unit in exactly the same way twice. Hence it is necessary to solve the image registration problem for the 3D case.

Image registration research in connection with breast cancer has been 2D in nature [125], [78], [104]. Rigorous testing of algorithms proposed for image registration should involve evaluating the performance of the algorithm for a variety of simple and complicated simulated distortions. Since the transformation relating the two

images would be known in these cases, one would be able to estimate the accuracy of performance of the algorithm. For example, accuracy of control points obtained using the method could be estimated if the transformation relating the given pair of images was known. The performance of the algorithm would also need to be studied when controlled noise is added to the original pair of images. Studies would also need to be performed when artificial structures are added to one or both the images. A true test of the algorithm would be if abnormalities are added to the images which are not clearly visible by mere observation of the images, and will become visible only on registration. However, none of the above-mentioned studies have rigorously tested their proposed algorithms. Also, none of the methods proposed in these studies can be easily extended to 3D.

While Zhou [125] has discussed the need for 3D imaging of the breast, the other two studies have not. Sallam's study [104], deals exclusively with matching of corresponding mammograms (bilateral and longitudinal). Sallam says in her abstract "The mammogram registration technique was successful in generating difference images in which at least 80% of the abnormalities were clearly identified by simple thresholding". However, the abnormality detection rate reaches 80% only for ill-defined masses. For other types of abnormalities, the detection rate is significantly lower and is accompanied by a large number of false positives. Besides, since she has used a publicly available mammogram database with extensive ground truth details, all the mammograms in the database are ones for which the radiologists were clearly able to see the abnormalities. Hence, her registration algorithm does not detect any abnormality not already detected by the radiologist and in fact, detects many less abnormalities on an average. She does not attempt to perform any registration of images where the abnormalities cannot be seen by the radiologists, to attempt to discover them. Also

the sizes of most abnormalities considered are larger than a centimeter. Abnormalities are missed by Sallam because she has attempted to register standard projection mammograms. Tissue structure projects itself differently and arbitrarily when two longitudinal mammograms of the same breast are taken and is dependent upon the particular way in which the breast was positioned for the particular image. Tissue structure in the left and right breast is inherently different. Hence comparing and registering mammograms directly cannot give us any meaningful results. Thus there is a need to develop 3D image registration techniques to register 3D breast images, where both images are of the same breast.

While there is a need for a 3D breast image registration algorithm, it is sufficient to first work out the methodology of finding the geometric transformation in two dimensions and then generalize the algorithm to three dimensions. In this thesis, we present an image registration algorithm for aligning breast images, which is 2D in its current form. However, an important characteristic of the algorithm is that it can be simply extended to 3D.

The thesis demonstrates the use of a class of textural transformations for breast image registration. While textural transformations have been used traditionally in such image processing tasks as image enhancement, feature extraction and image compression, few studies have used textural features for image registration. When an image gets distorted, pixels in the image are relocated to other positions. However, when the pixel moves to a new location, it carries along with it, its local neighborhood. In other words, given two images A and B related by a geometric distortion T , local neighborhoods of a point in A will continue to be local neighborhoods of the same point in B because of continuity considerations. Thus when textural features of local neighborhoods are calculated for pixels in A , these textural features are also found in B . Hence image registration can be performed by matching these textural features

in the two images. This is the central idea behind the thesis.

This thesis is devoted primarily to the solution of the image registration problem (with particular emphasis on breast images) using a textural transformation called the *starbyte transformation*. The starbyte transformation of a given image is generated using local neighborhood properties of pixels in the image. A textural value at a pixel was obtained as a bit string using binary comparisons of average grey values over different sectors in the local neighborhood of a pixel. However textural transformations can be done in several other different ways. When applied to breast images, the starbyte transformation converts them to texture maps with clearly demarcable regions. The starbyte transformation has the property that the deformation or warp relating the original images is left virtually untouched so that matching of the transformed images is equivalent to matching of the original images. Accurate control points can thus be easily identified by extracting them from the starbyte-transformed images.

A complete registration algorithm is presented which consists of first starbyte-transforming the original images, extracting control points, and performing unwarping using thin plate splines.

A major portion of this thesis consists in systematically studying the performance of the starbyte registration algorithm for different distortions. Portions of mammograms were used to conduct this study. This is not only because these are the most commonly available breast images, but also because the texture in the images could be used to study the performance of the algorithm. Furthermore, in the absence of high resolution (0.1 mm) CT data, mammograms are the best available approximation to the X-ray texture of the breast. Since the starbyte values at every pixel denote a textural property of its neighborhood, and since mammograms are, by nature low contrast and noisy, by using mammograms for this study we were able to evaluate

if this textural property alone could be used for image registration. The portions of mammograms were distorted using several simple and complicated simulated distortions. The reference and target images for each distortion were starbyte-transformed. Control points were then identified manually as well as automatically on the starbyte-transformed images. A manual and an automatic algorithm for identification of control points from the starbyte-transformed images have been presented in detail with several examples. Limited studies, with noise and simulated tumors added to the original images, have also been carried out.

We also discuss an important issue: namely, the need for 3D breast imaging. Controlled preliminary studies with two pairs of roughly corresponding slices taken from MRI 3D data sets of volunteers have been presented. We have also presented an important result using a pair of approximately corresponding portions of longitudinal mammograms to show that obtaining a difference image after matching them cannot give us any useful results. What we have thus shown is that if we view a projection mammogram strictly as a 2D picture, then registration of a pair of such pictures is effectively impossible. It is clear that radiologists using a baseline mammogram are inferring some 3D structure in order to make comparisons.

We shall now discuss image registration briefly only dealing with those issues which pertain directly to the thesis. For a general and comprehensive review of previous work on image registration, please refer to [13], [118].

1.1 Image registration

Image registration is the process of overlaying two images of the same scene, one of which is called the *reference* image A and the other called the *sensed* or *target* image B . The two images are related by a geometrical transformation T such that

$$T(A) = B.$$

When considering 2D images, we assume that T is planar and that it is a 1-1 mapping. However, the latter assumption is violated most commonly at the edges of the image, where portions of A have been mapped to regions outside the boundary of B . We have considered examples of a few such transformations in the next few chapters. The 1-1 mapping may also be violated, in breast images, if there are actual differences in the two images, apart from differences created because of a geometric distortion. For e.g., due to surgery, tumor growth, menstrual cycle, lactation, or aging, images A and B may not be related by a 1-1 mapping. The breakdown of the 1-1 mapping can also be created by a breakdown of continuity in one of the images. For e.g., if T were a large and severe distortion, then several pixels in A may be mapped to one pixel in B and vice versa.

Image registration is the process of finding or estimating T . The problem of estimating T is equivalent to the problem of determining an “unwarped” image, B^* which is close in some numerical sense, to B . Image registration can also be formulated as the problem of estimating T^{-1} , in which case the unwarped image would be numerically close to the reference image. The two formulations are equivalent. In this thesis, we have chosen the first formulation. Estimating T , in general (but cf. [79]) involves 3 steps [32], [13]:

1. selection of control points in the reference and target images and determination of the correspondence between them;
2. determination of the type and parameters of the mapping function using known coordinates of control points;
3. geometric transformation of the reference image by means of the mapping function to obtain the unwarped image with gray level resampling.

1.1.1 Control point selection

The problem of control point selection has been extensively studied in the literature. Intrinsic control points are markers which are placed on the object being imaged just for the purpose of registration and can be easily identified in both images. However, whenever intrinsic control points are not available (as in diagnostic images), they have to be identified extrinsically either manually or automatically. Extrinsic control points are unambiguous point features which can be identified in both the images. Typical features used as control points are line intersections [115], window centres [5], and centres of gravity of closed boundary regions [47]. Then correspondence between the control points is done by clustering [115], image correlation [5] and relaxation [47]. Most of these techniques for control point selection are proposed for simple rigid or affine transformations and also for images where unambiguous features are present which can be identified as control points. Low-contrast and noisy images such as breast images lack prominent landmarks which can be specified or identified as control points. Visual examination of the images also does not yield many control points.

1.1.2 Determination of the mapping function

The determination of the mapping function has also been extensively studied in the literature. It involves two stages: selection of the appropriate type of function and determination of its parameters from given control point coordinates. In the past, polynomials of low degrees were used as mapping functions [47], [40], [114], [119], [69]. Adams [1] presented hardware for geometric unwarping using a quadratic unwarping function. The coefficients of the global polynomial are determined by minimizing the

sum-of-squares error at the control points. However, the use of a single set of transformation functions to register the whole image assumes that the original distortion relating the two images is global and has no local variation. Hence the use of global polynomials as mapping functions is restrictive. Besides, only lower order polynomials can be used, because their behavior becomes unpredictable when the degree is increased [34]. While the use of orthogonal polynomials alleviates this problem to some extent [50], even then the major limitation of using a global transformation is not removed. However recent advances in local orthogonal basis functions, particularly wavelet bases, can improve the global nature of classical orthogonal expansions [3].

For correcting local distortions, interpolation over the set of control points has to be performed. A spatial mapping function for each coordinate is specified which interpolates between the matched coordinate values. Consider the 2-D case. Let (X_i, Y_i) and (x_i, y_i) represent control points in the reference and target images, for $i = 1 \dots N$. Two bivariate smooth functions F_x and F_y are constructed as:

$$\begin{aligned} X_i &= F_x(x_i, y_i) \\ Y_i &= F_y(x_i, y_i) \end{aligned} \tag{1.1}$$

These functions are to be selected in such a way that they interpolate smoothly between the control points.

Methods used for correcting local distortions must be designed for irregularly spaced data points since the control points will, in general, not be a regular grid. The problem of constructing interpolation functions for sets of scattered data in 2-D has been extensively studied. The problem of obtaining a smooth interpolation function for scattered data can be viewed as a surface approximation problem. Important survey papers include [105] and [6]. Dyn *et al* [30] have proposed a very efficient

iterative scheme for computing interpolation surfaces. Franke [35] has conducted a detailed study and evaluation of several surface approximation techniques. Most of the methods evaluated by Franke use the general spline approach to piecewise interpolation. This requires the selection of a set of basis functions and a set of constraints to be satisfied so that solving a system of simultaneous linear equations will specify the interpolating function. In general, most approaches come under three classes:

1. *Inverse-distance weighted interpolation:* In this case, a weighted sum of neighboring points is computed where the weights are related inversely with distance. A classical method using inverse-distance weighted interpolation is by Shepard [108].
2. *Triangle-based methods:* These methods are based on the partitioning of the image and the weights for interpolation are chosen based on the properties of each subregion (created by the partitioning). A common type of partition used is triangulation. A complicated transformation can be represented by a simple piecewise approximation by defining a set of contiguous quadrilaterals or triangles across the image made by connecting neighboring control points and allowing a different polynomial transformation within each triangle or quadrilateral. Optimal triangulation, itself is a well-researched area (refer to [10] for a good introduction), with readily available and published computer programs [109], [117] and algorithms [2], [68] for the same. Piecewise linear [46] and piecewise cubic [48] mapping functions (after triangulation of control points) have been proposed for image registration.
3. *Global basis function type methods:* These techniques include the surface spline techniques which interpolate the data by representing it as the surface of an

infinite plate under the imposition of point loads, i.e. the data. Because of this property, they are also called *thin plate splines* (TPSs).

Among the methods studied by Franke, the surface splines or TPSs gave the most accurate results and hence have been used as mapping functions for image registration [49]. Franke ([36], [37]) discusses interpolation of scattered data using TPS. Their direct use is computationally intensive and evaluation algorithms have become available which have reduced the computational complexity ([91],[9]). Flusser [32] has proposed a modification to the direct use of TPSs, which also reduces the computational complexity. For the derivation of TPSs, please refer to [12], [32]. We have used TPSs as the mapping functions throughout this thesis.

1.1.3 Geometric transformation with grey level resampling

Geometric transformation of the reference image to the unwarped image not only involves calculating the geometric correspondence of points in the reference and unwarped image (using the mapping function), but also involves grey level resampling [125]. In general, when the unwarped image is constructed from the reference image, grid (pixel) values in the unwarped image will correspond to non-grid locations in the reference image. If a pixel in the unwarped image maps to a position between four pixels in the reference image, then its grey level is computed from these four pixels by means of interpolation. Zhou [125] discusses different techniques for grey level interpolation. While the simplest and most primitive is the nearest neighbor approach, more accurate methods of interpolation are bilinear interpolation, truncated sinc, Lagrange polynomials, cubic splines, cubic convolution interpolations, etc. We have used bilinear interpolation throughout for grey level resampling because this was the simplest to implement.

We shall now briefly discuss the subject matter of the thesis.

1.2 Organization of the thesis

The organization of the thesis is as follows:

Chapter 2 discusses the starbyte transformation in detail with several examples of image pairs related by different distortions or transformations. Generalizations to the starbyte transformation concept are also discussed. The distortions studied in this thesis are a bilinear warp, translation, rotation, scaling, a TPS distortion, and a sinusoidal distortion.

The 3rd chapter discusses preprocessing of the starbyte-transformed images (STs). This preprocessing or “cleaning up” of the STs is required so that they can be subsequently used for control point extraction. The STs have a small and manageable number of density levels. Hence, the STs can be density sliced at each density level to create binary images. These binary images contain very clear and identifiable regions with characteristic and recognizable shapes. The 3rd chapter presents these density slices for six sets of image pairs related by the six distortions. We also show, numerically, in the 3rd chapter, that the centroids of the regions in each slice are good control points.

The 4th and 5th chapters discuss, in detail, a manual and automatic procedure to extract control points from the STs. Results are presented for the same six pairs of images as in the 3rd chapter.

The 6th chapter discusses image registration for the six distortions in detail. Once the control points are obtained, the unwarped image is calculated using TPSs as mapping functions. The unwarped images obtained using the manual as well as automatic case for the six distortions are presented and compared.

The 7th chapter performs a limited study on the accuracy of control points when noise is added to the original images. Additive noise having a Gaussian distribution is used. This study is conducted for two of the six distortions, namely the bilinear and sinusoidal distortion. These two distortions are selected because they represent distortions of large magnitude and are also reasonably complicated.

The 8th chapter performs a limited study on the performance of the starbyte registration algorithm when simulated tumors are added to the target image. The tumors have been added to the images in such a way that they are not clearly visible on the target images as well as are not clearly visible on plain subtraction of the reference and target. In a sense, these are analogous to those kinds of tumors which are missed by mammography, and hence can become visible only after registration and subtraction.

The 9th chapter discusses the need for 3D imaging and presents preliminary results on the matching of two pairs of roughly corresponding MRI slices using the starbyte registration algorithm. These were taken from 3D data sets of two volunteers who were imaged twice. This chapter also presents an important example to show that matching of longitudinal mammograms does not give us any meaningful results.

The 10th chapter discusses future work which can be done based on the work done in this thesis.

We have also included an appendix which contains copies of papers which are related to the subject matter of the thesis but do not repeat the material presented in the thesis.

Chapter 2

THE STARBYTE TRANSFORMATION

2.1 Introduction

The previous chapter dealt with the need for 3D breast imaging and also identified image registration as an important problem in breast imaging. This chapter introduces a new textural transformation, called the *starbyte transformation* (ST). When applied to breast images, the starbyte transformation converts them to texture maps with clearly demarcable regions. We shall show by examples, that the starbyte transformation has the property that the deformation or warp relating the original images is left virtually untouched so that matching of the transformed images is equivalent to matching of the original breast images, i.e., the starbyte transformation and warping are effectively commutative. Since matching the starbyte-transformed images (STs) is equivalent to matching the original image pair, and since matching the STs is easier than matching the original image pair, the breast image matching problem has been significantly simplified by the use of the starbyte transformation.

As we shall show in the next section, starbyte transformations can be calculated in different ways. We shall call each way or method of calculating a starbyte-transformation a *protocol*. This chapter will present the STs for different image pairs related by different distortions or warps for different protocols. Subsequent chapters will then discuss the problem of extraction of control points from the STs of a given image pair related by different distortions. In subsequent chapters, we will

restrict ourselves to mammographic regions related by simulated distortions. In the 9th chapter, we shall present results with two pairs of real breast MRI image slice pairs.

2.2 The starbyte transformation procedure

The starbyte transformation procedure assigns a number to each pixel in an image, designating the local pattern of which it is part. While this number can represent any property of the local neighborhood of the pixel, we have chosen to generate starbyte transformations in the way described below:

We first consider the continuous domain. For a spatially continuous picture (a function in R^2), a circular neighborhood around each point can be divided into n sectors [60] (Figure 2.1a). Each sector is filled in with black or white, depending on whether the average value in the sector is below or above the picture density at the center of this neighborhood (or the average density over the whole neighborhood). Some such patterns will look like “stars”, with black taking the value 0, and white taking the value 1. The bits assigned to the black and white sectors, when concatenated into a binary number, form a “byte” (not necessarily of 8 bits unless there are 8 sectors). If f is the original image, a new picture s is made such that $s(x, y)$ takes on the value of the binary number formed from $f(x, y)$ and its sectorized neighborhood. Hence we call these patterns “starbytes” and we use the name “starbyte transformation” to generalize this pattern-generating scheme which transforms $f(x, y)$ to $s(x, y)$.

Starbytes, as defined in terms of sectors of a neighborhood, capture some of the local structure in an image. For a digital image, we take the neighborhood around a pixel as a square region of a certain size. This neighborhood is first divided into

n "sectors". The bits in the starbyte pattern are then generated by binary comparisons between the average grey level over each sector taken in turn with either the average grey level over the entire neighborhood or the grey level of the pixel under consideration (which is at the center of the neighborhood). Thus for n sectors in a neighborhood, n binary comparisons result in a set of n bits forming a bit string. There are 2^n local patterns or "starbytes" defined in this way, which thus segment any picture into regions with 2^n kinds of labels.

The "sectors" can be generated in different ways, as subsets of the pixels in a neighborhood. Each way of designating the subsets gives a different starbyte pattern map for the same image.

Figure 2.1b (page 17) shows a 3×3 neighborhood around a point P . The pixels in this 3×3 matrix are labeled X_m for $m = 0$ to 8 with pixel X_8 being the point P itself for which the starbyte pattern is being calculated. Let the symbol " $Av(Q_i)$ " represent the average grey level of subset Q_i , $i = 0, \dots, n - 1$ of pixels X_0, X_2, \dots, X_m . The binary function $B(\beta, \alpha)$ will be used to generate the bit pattern. $B(\beta, \alpha)$ evaluates to 1 when $\alpha > \beta$, and evaluates to 0 otherwise.

In general, we set $b_i = B(X_m, Av(Q_i))$ or $b_i = B(Av(X_0, \dots, X_m), Av(Q_i))$. For the n sectors, there are $n!$ permutations by which we could map the bits b_i , $i = 0, \dots, n - 1$, into a binary number containing n binary digits, i.e., there are $n!$ ways of generating scalar numerical labels for the starbytes. Hence, even when the bits making up the pattern for a particular pixel are determined, there are $n!$ possible bitmaps. The sequence or ordering of bits may be fixed ahead and maintained constant for all pixels in the image. Since it is not clear in advance which permutation will generate the best starbyte pattern, let us denote $S(b_0, b_1, \dots, b_n)$ as the set of bits denoting a pattern when we do not want to explicitly specify the actual permutation.

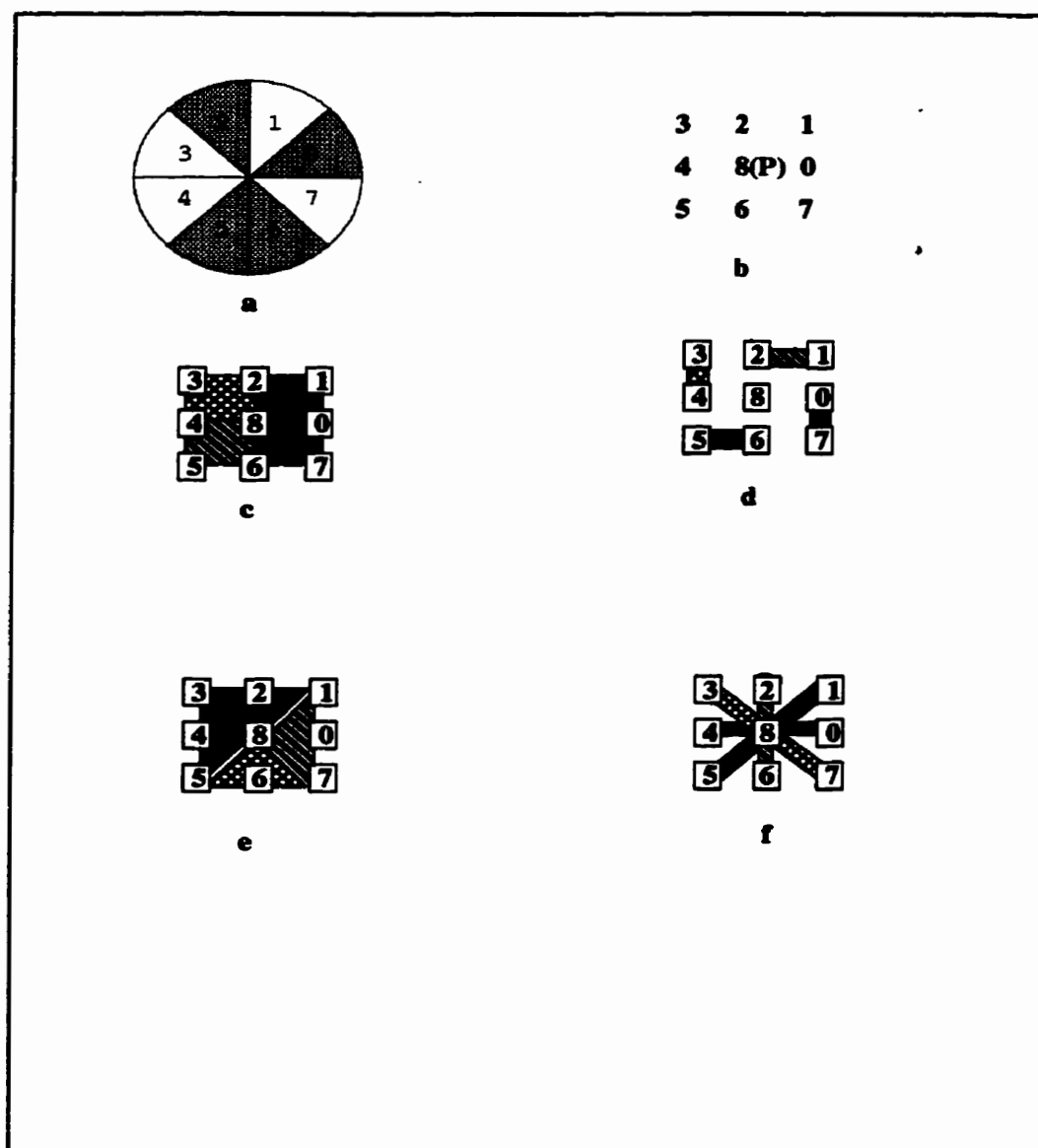


Figure 2.1: *a*: Continuous space origin of the concept of starbyte patterns. Each sector of a neighborhood is assigned a bit 1 or 0 according to whether its average density is above or below the density at the center of the circle, or above or below the average over the circle. *b*: 3×3 neighborhood around point *P* (labeled #8). *c* – *f*: Sectors used for protocols 1 to 4 (See equations for protocols 1 to 4).

Using the above notations, the following are examples of 4-bit starbyte patterns, where we just use i to designate pixel X_i :

1.

$$\begin{aligned} &S(B(Av(0...8), Av(2, 3, 4, 8)), \\ &B(Av(0...8), Av(0, 1, 2, 8)), \\ &B(Av(0...8), Av(4, 5, 6, 8)), \\ &B(Av(0...8), Av(0, 6, 7, 8))) \end{aligned}$$

Here B is evaluated by comparing the average grey level over the entire neighborhood with averages over each of 4 sectors generated by dividing the neighborhood into 4 equal regions at P (See Figure 2.1c).

2.

$$\begin{aligned} &S(B(8, Av(3, 4)), \\ &B(8, Av(1, 2)), \\ &B(8, Av(5, 6)), \\ &B(8, Av(0, 7))) \end{aligned}$$

Here B is evaluated by comparisons between the grey level at pixel P with the averages over 4 regions where each region is a group of 2 pixels (See Figure 2.1d).

3.

$$\begin{aligned}
&S(B(Av(0...8), Av(1, 2, 3, 8))), \\
&B(Av(0...8), Av(0, 1, 7, 8)), \\
&B(Av(0...8), Av(5, 6, 7, 8)), \\
&B(Av(0...8), Av(3, 4, 5, 8)))
\end{aligned}$$

The average over the whole neighborhood is compared with averages over 4 sectors created by dividing the neighborhood across its diagonals (See Figure 2.1e).

4.

$$\begin{aligned}
&S(B(8, Av(3, 7, 8))), \\
&B(8, Av(1, 5, 8)), \\
&B(8, Av(2, 6, 8)), \\
&B(8, Av(0, 4, 8)))
\end{aligned}$$

Here the grey level at P is compared with the average over groups of 3 pixels (See Figure 2.1f).

In patterns 1 and 3 (Figures 2.1c and e), the sector grey levels are compared with the average grey level of the entire neighborhood of P , while in 2 and 4 (Figure 2.1d and f), the sector grey levels are compared with the grey level at P alone. For a neighborhood of size $s \times s$, there are $2^s - 1$ non-null definable subsets of its pixels, each subset representing a potential “sector”. Any subset of the set of sectors can be used to create a starbyte pattern and the bit patterns can be permuted in any

way. Note that we have not required that the sectors chosen form a partition of the neighborhood.

From the above, it is clear that there are many ways to generate starbyte patterns for the pixels in an image.

2.2.1 Size of the neighborhood

If we use the same protocol to generate the starbytes, different patterns will emerge if the local neighborhood size is varied. In order to use the same logic or protocol for bit-generation as described above, the neighborhood size is maintained as an odd multiple of 3. Thus a $3k \times 3k$ neighborhood ($k = 1, 3, 5, \dots$) can be converted to a 3×3 neighborhood by dividing the $3k \times 3k$ neighborhood matrix into a 3×3 block matrix, each block being of size $k \times k$. Then each block is replaced by its average grey level, thus reducing the $3k \times 3k$ original neighborhood to a 3×3 neighborhood, so that the bit-generating scheme outlined above can be used without change. The size of the neighborhood can remain fixed for all pixels in the image or the size can be adaptively determined at each pixel by optimizing a contrast measure versus neighborhood size [43], [82], [107], [25], [94], [89].

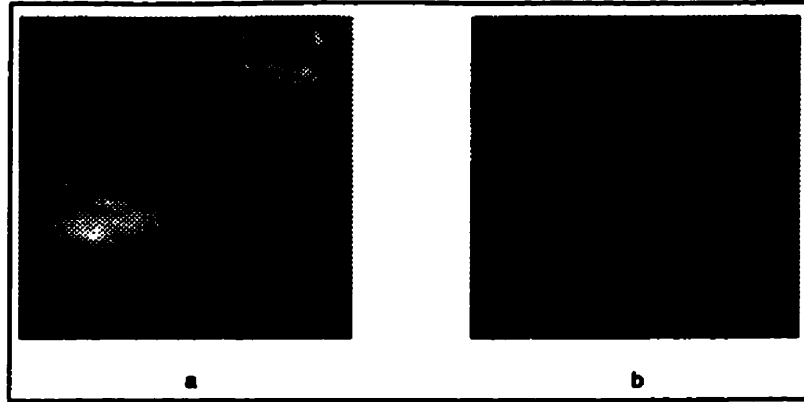


Figure 2.2: 128×128 blocks cropped out a mammogram with and without microcalcifications respectively.

2.3 Results and Discussion

2.3.1 Starbyte-transformed images for a bilinear distortion

Figures 2.2a and b are 128×128 blocks cropped out a mammogram (Patient # 18, oblique view from [86]). Since this was a mammogram which contained microcalcifications, we decided to use two blocks, one with, and one without microcalcifications. Figure 2.2a is a region in the original mammogram containing microcalcifications, while Figure 2.2b from the same image, contains no microcalcifications.

Figures 2.2a and b are warped using the same bilinear transformation to obtain Figures 2.3a and b (page 22).

The warped images were calculated by constraining the points $(40,40)$, $(40,80)$, $(80,40)$ and $(80,80)$ in the original image to go to the points $(36,44)$, $(43,84)$, $(76,46)$ and $(85,84)$ respectively in the warped image through a bilinear map. The bilinear inverse map [122] used was:

$$\hat{x} = 8.594463 + 1.076072x - 0.113649y - 0.001473xy$$

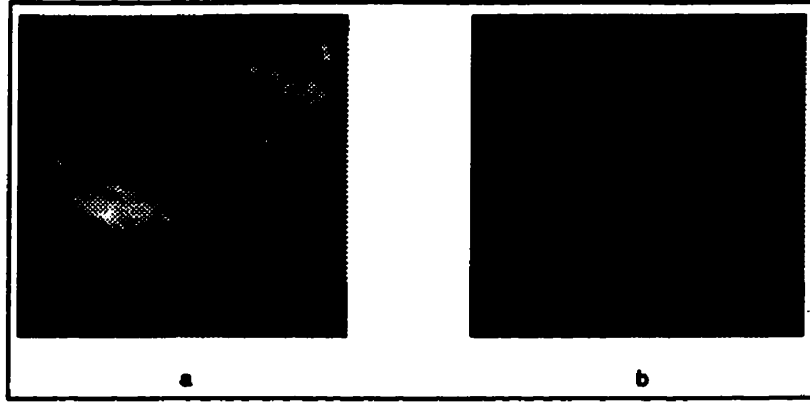


Figure 2.3: Figures 2.2a and b, warped using the same bilinear transformation.

$$\hat{y} = -0.021511 - 0.110521x + 0.952639y + 0.001316xy \quad (2.1)$$

where (\hat{x}, \hat{y}) are the coordinates in the original image and (x, y) are the coordinates in the transformed image. This representation is so that, the grey values at each pixel (x, y) in the transformed image can be obtained by direct substitution of x and y values on the right hand side of the equations to obtain the corresponding \hat{x} and \hat{y} values in the reference image. The numbers \hat{x} and \hat{y} would typically be floating point numbers. Since the grey values at all pixels in the reference image are known, we would then be able to perform simple interpolation to obtain the grey value at the pixel (x, y) in the transformed image.

The bilinear map above created a maximum distortion of about 20 pixels in the x direction and about 15 pixels in the y direction.

In the following examples, 8 starbyte protocols are used with different neighborhood sizes. The first 4 are as described in section 2.2, while the next four are:

5.

$S(B(Av(0...8), Av(1, 2, 3)),$
 $B(Av(0...8), Av(0, 1, 7)),$
 $B(Av(0...8), Av(5, 6, 7)),$
 $B(Av(0...8), Av(3, 4, 5)),$
 $B(Av(0...8), Av(3, 7, 8)),$
 $B(Av(0...8), Av(1, 5, 8)))$

6.

$S(B(Av(0...8), Av(1, 2, 3)),$
 $B(Av(0...8), Av(0, 4, 8)),$
 $B(Av(0...8), Av(5, 6, 7)),$
 $B(Av(0...8), Av(3, 4, 5)),$
 $B(Av(0...8), Av(2, 6, 8)),$
 $B(Av(0...8), Av(0, 1, 7)))$

7.

$$\begin{aligned}
 &S(B(8, 3), \\
 &\quad B(8, 2), \\
 &\quad B(8, 1), \\
 &\quad B(8, 4), \\
 &\quad B(8, 0), \\
 &\quad B(8, 5), \\
 &\quad B(8, 6), \\
 &\quad B(8, 7)))
 \end{aligned}$$

8.

$$\begin{aligned}
 &S(B(Av(0...8), Av(2, 3, 4, 8)), \\
 &\quad B(Av(0...8), Av(0, 1, 2, 8)), \\
 &\quad B(Av(0...8), Av(4, 5, 6, 8)), \\
 &\quad B(Av(0...8), Av(0, 6, 7, 8)), \\
 &\quad B(Av(0...8), Av(1, 2, 3, 8)), \\
 &\quad B(Av(0...8), Av(0, 1, 7, 8)), \\
 &\quad B(Av(0...8), Av(5, 6, 7, 8)), \\
 &\quad B(Av(0...8), Av(3, 4, 5, 8)))
 \end{aligned}$$

Protocols 5 and 6 are 6-bit starbyte patterns while protocols 7 and 8 are 8-bit patterns. Figures 2.4a to h and Figures 2.4i to p show the 8 starbyte images for Figures 2.2a and b, generated using these 8 protocols for a 3 x 3 neighborhood. Figures 2.5a to h and Figures 2.5i to p show the same for their warped counterparts. Figures 2.6a to h

and Figures 2.6i to p, and Figures 2.7a to h and Figures 2.7i to p show the same for a 9 x 9 neighborhood. Figures 2.8a to h and Figures 2.8i to p, and Figures 2.9a to h and Figures 2.9i to p show the same for a 15 x 15 neighborhood. Figures 2.10a to h and Figures 2.10i to p, and Figures 2.11a to h and Figures 2.11i to p show the same for an adaptive neighborhood. The next few pages show Figures 2.4 to Figures 2.11. The text continues on page 34.

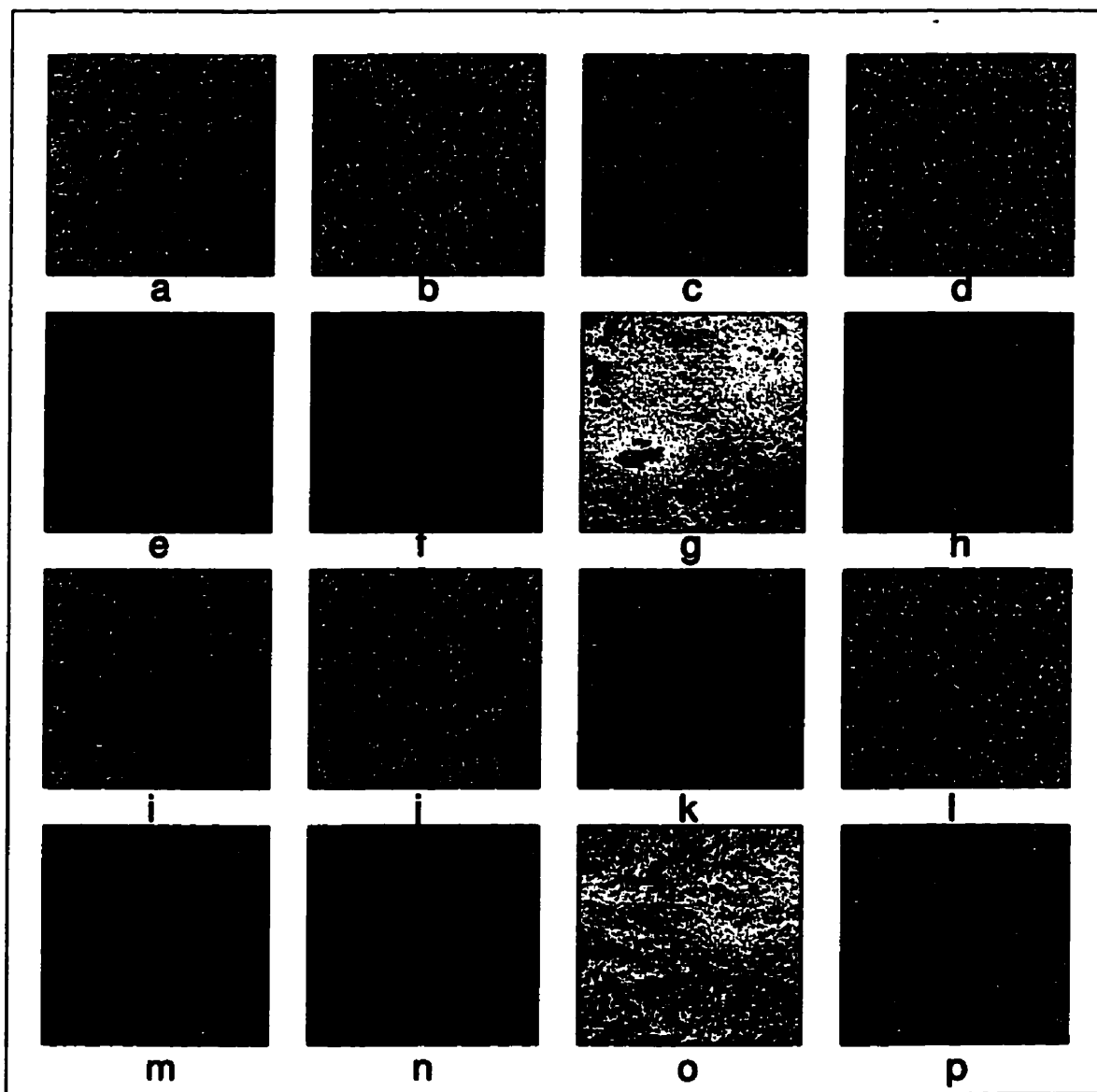


Figure 2.4: *a – h*: Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 3×3 neighborhood. *i – p*: Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 3×3 neighborhood.

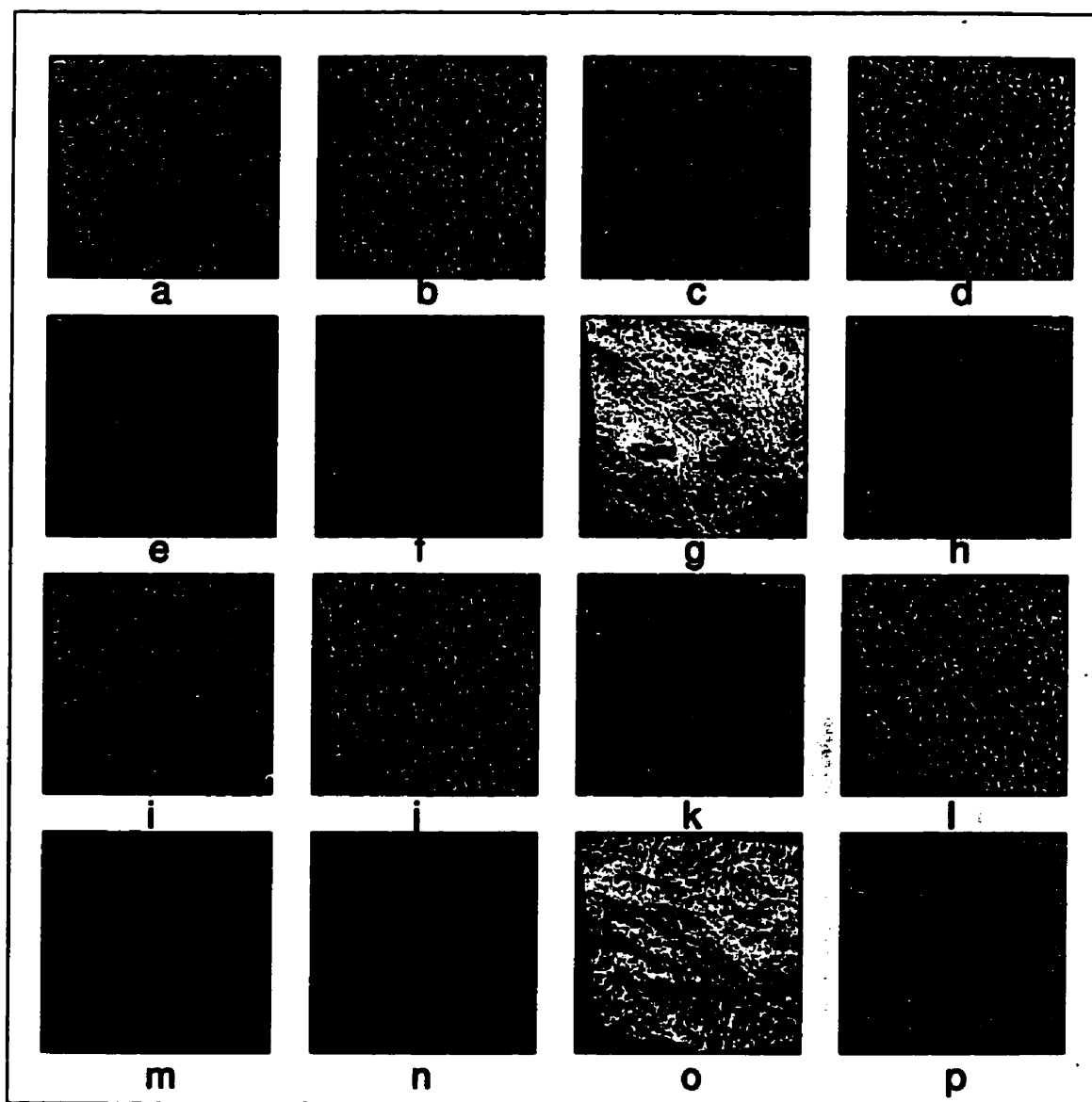


Figure 2.5: *a – h*: Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 3×3 neighborhood. *i – p*: Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 3×3 neighborhood.

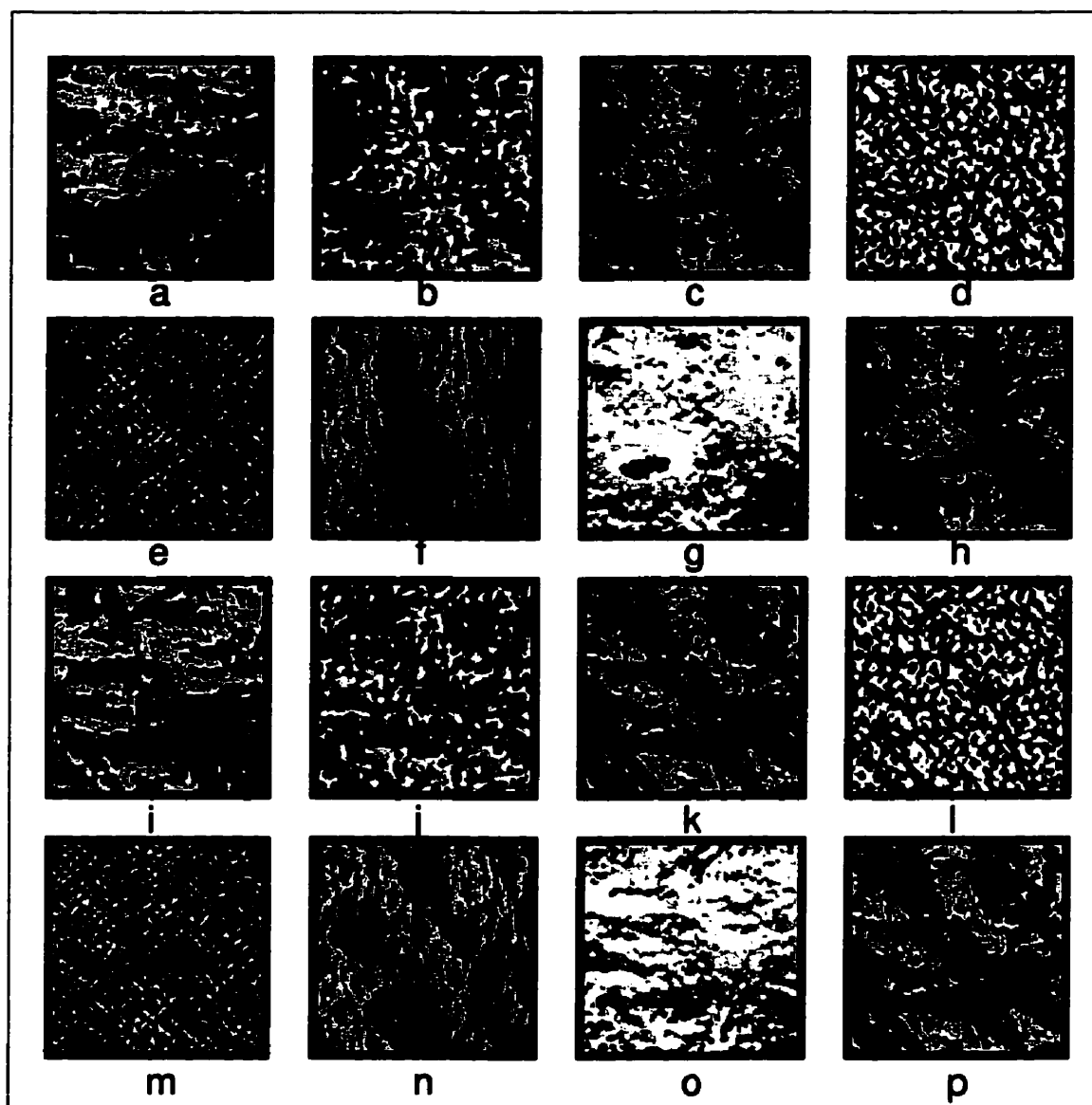


Figure 2.6: *a – h*: Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 9×9 neighborhood.

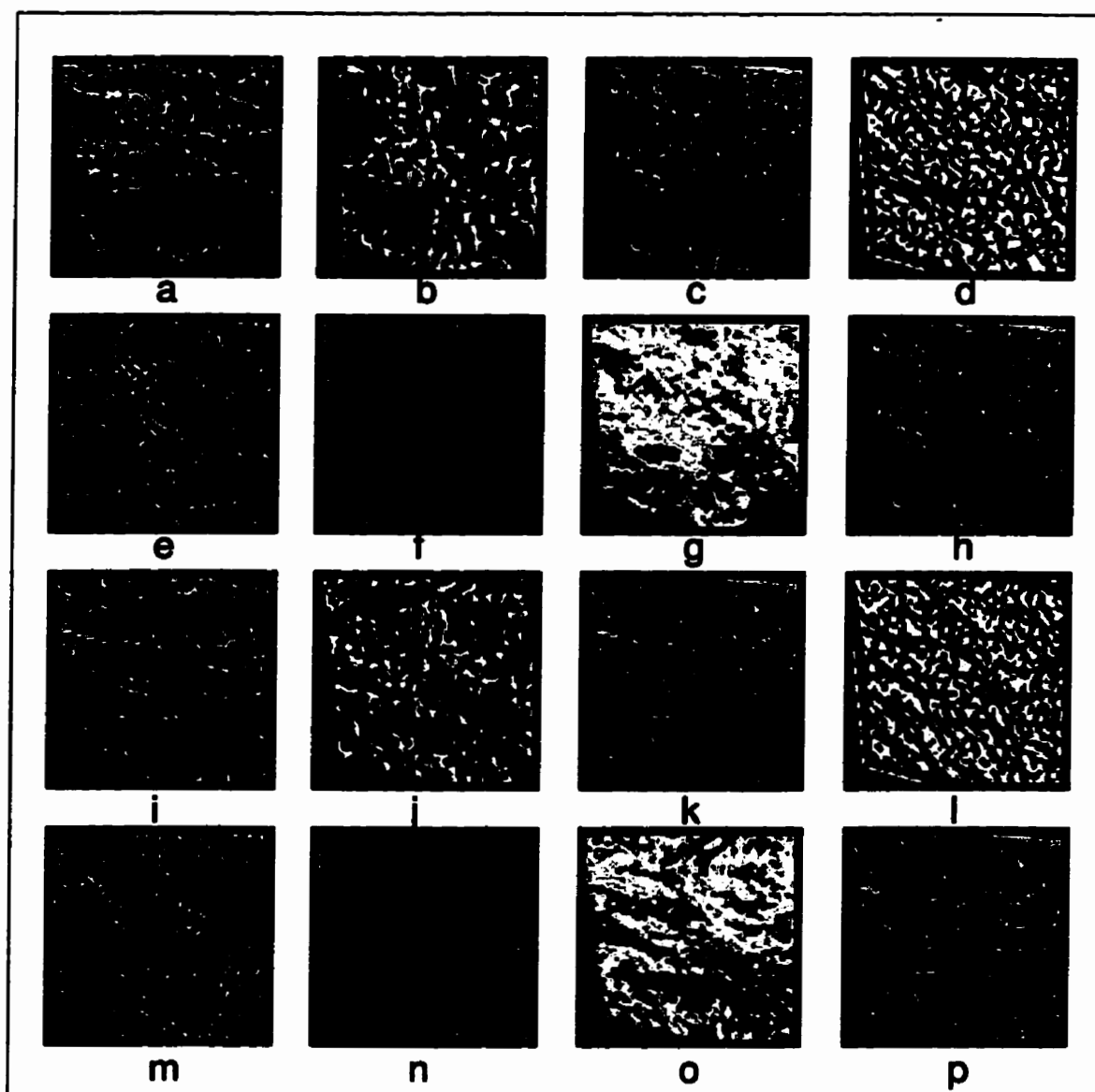


Figure 2.7: *a – h*: Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 9×9 neighborhood.

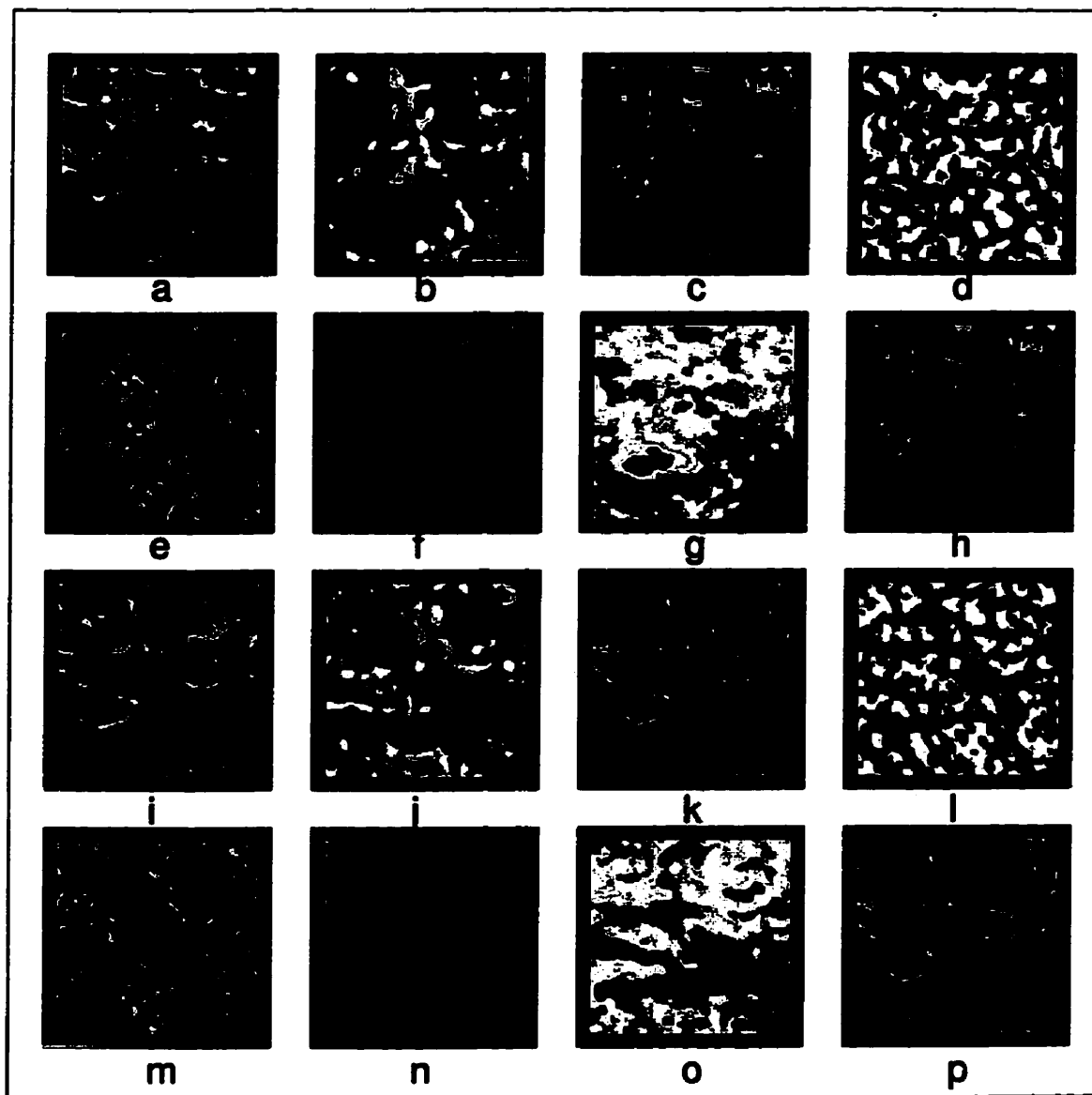


Figure 2.8: *a* – *h*: Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for a 15×15 neighborhood. *i* – *p*: Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for a 15×15 neighborhood.

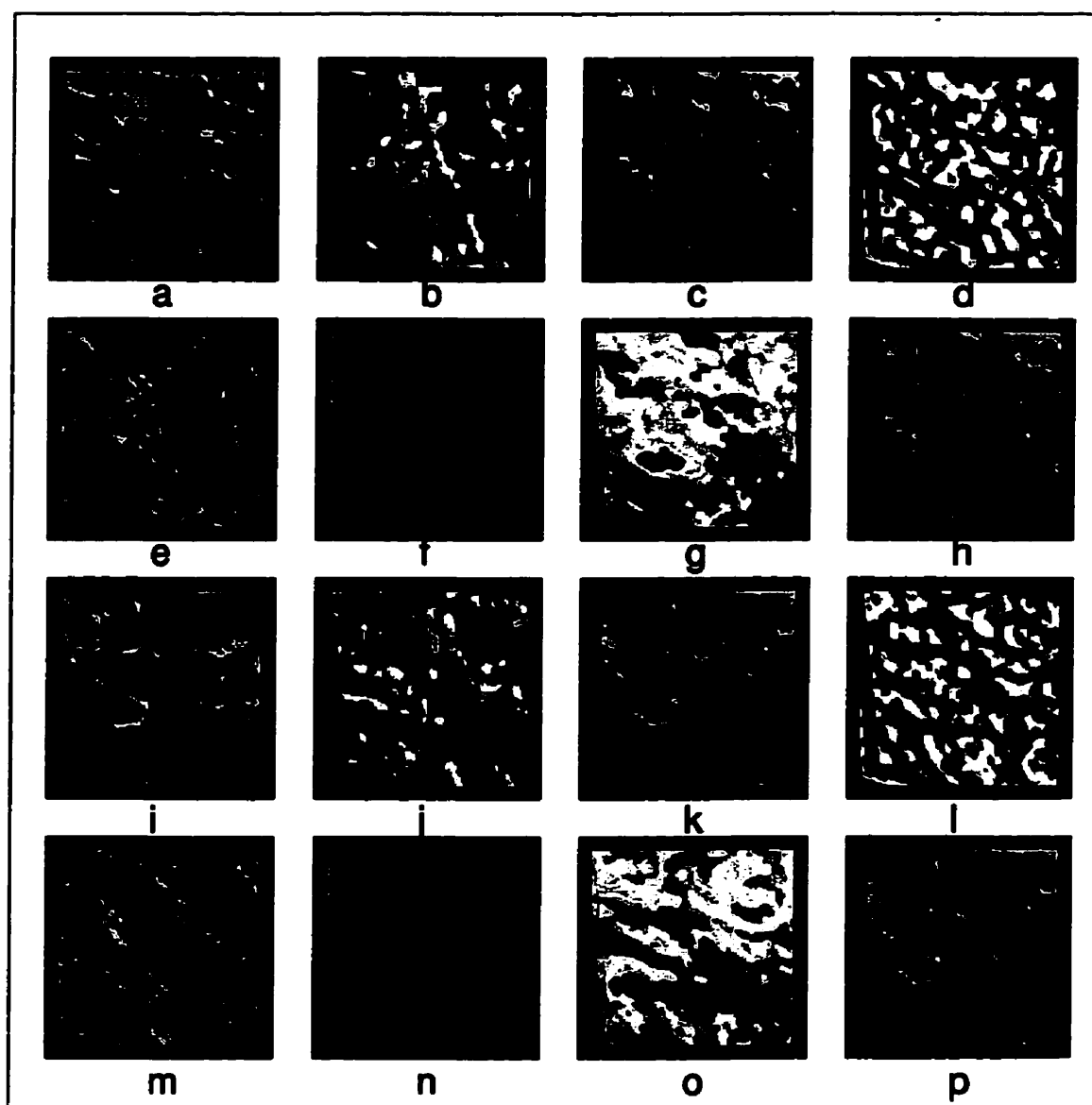


Figure 2.9: *a – h*: Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for a 15×15 neighborhood. *i – p*: Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for a 15×15 neighborhood.

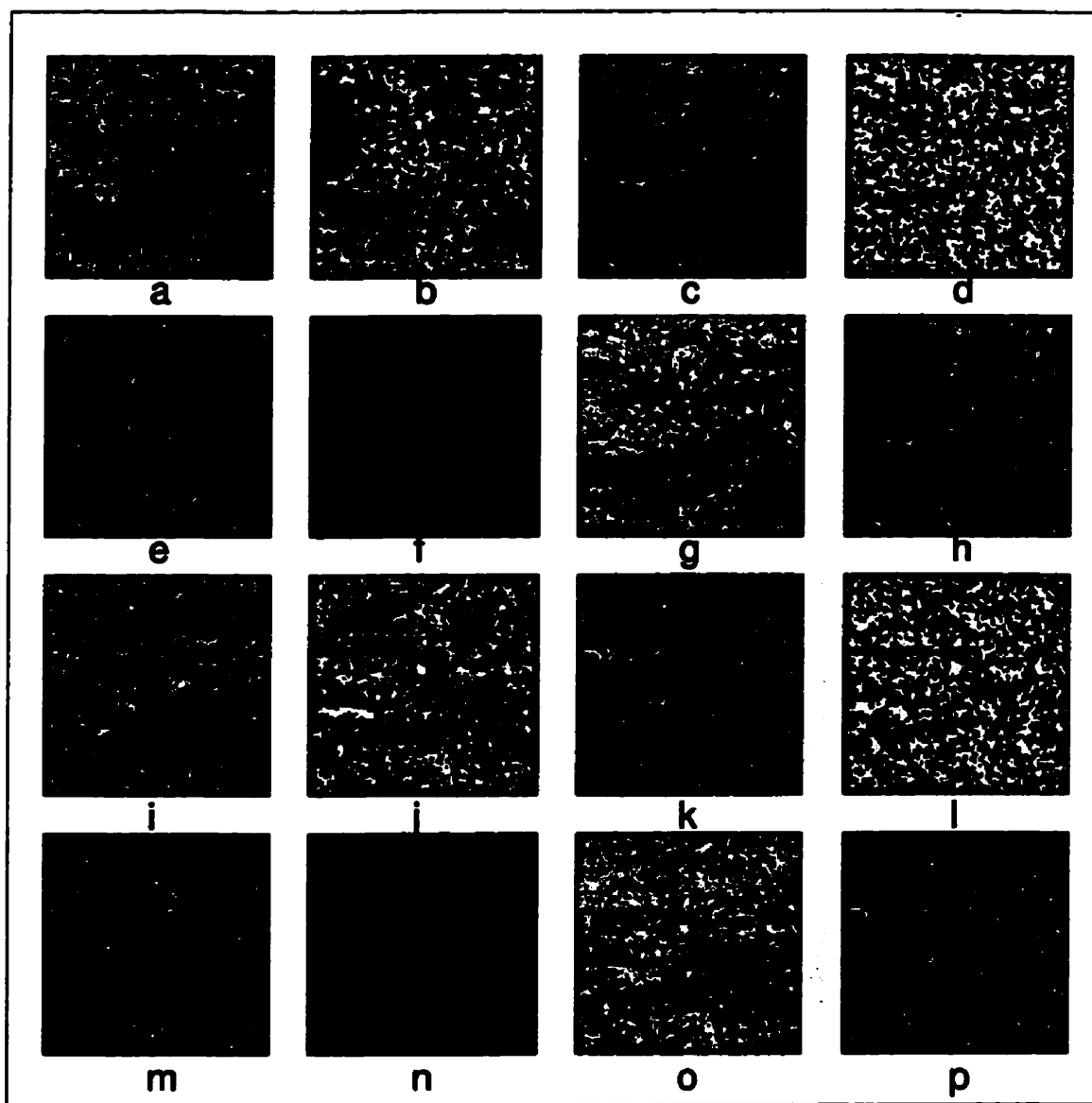


Figure 2.10: *a – h*: Starbyte transformed images of Figure 2.2a, using protocols 1 to 8, for an adaptive neighborhood. *i – p*: Starbyte transformed images of Figure 2.2b, using protocols 1 to 8, for an adaptive neighborhood.

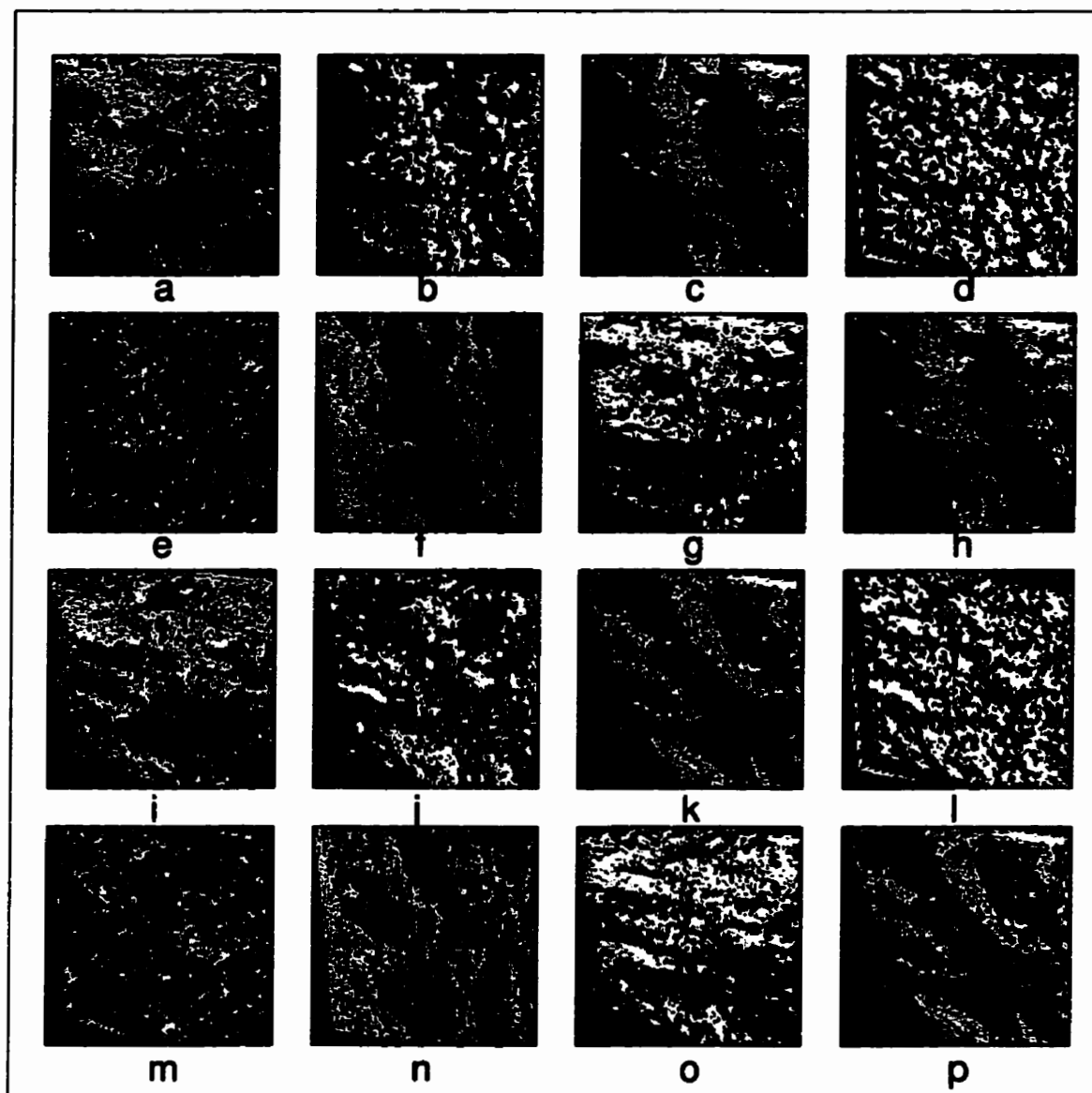


Figure 2.11: *a – h*: Starbyte transformed images of Figure 2.3a, using protocols 1 to 8, for an adaptive neighborhood. *i – p*: Starbyte transformed images of Figure 2.3b, using protocols 1 to 8, for an adaptive neighborhood.

The neighborhood size for each pixel is adaptively determined by evaluating a contrast function at each pixel for different neighborhood sizes [25]. That size corresponding to the first maximum in the contrast function curve represents the correct neighborhood size for the pixel. Since the local texture at each pixel may vary, by using an adaptive neighborhood, we select that neighborhood size which corresponds closest to the size of the local texture at the pixel.

Figures 2.12a and b (page 35) show the distribution of neighborhood sizes as a bar chart for Figures 2.2a and b. Figures 2.12c and d show the distribution of neighborhood sizes as a bar chart Figures 2.3a and b. Since the neighborhoods are always odd multiples of 3, the bars are shown exactly at the corresponding neighborhood size values.

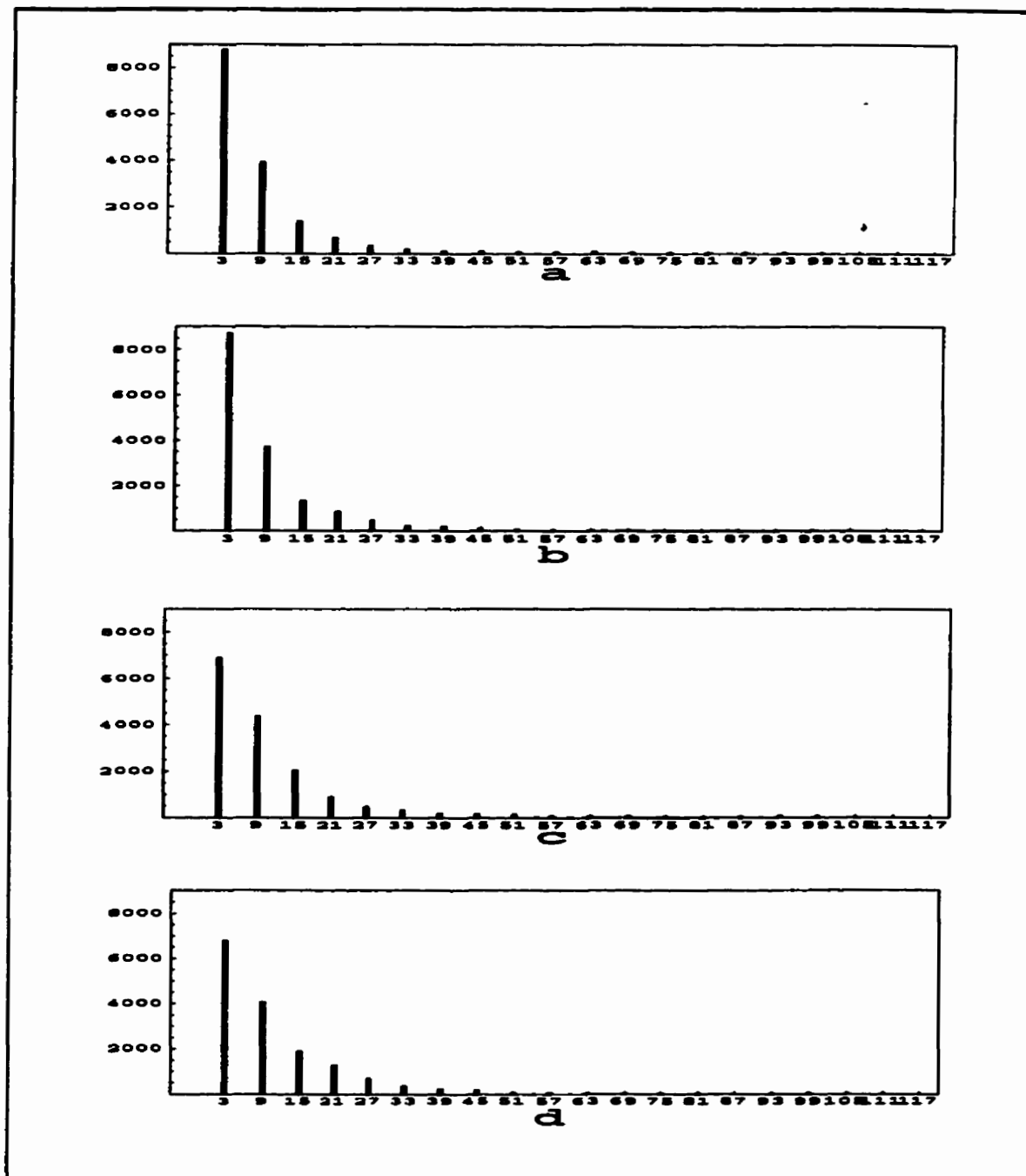


Figure 2.12: *a* to *d*: Distribution of the neighborhood sizes shown as a bar chart for Figures 2.2a and b and Figures 2.3a and b respectively.

It is obvious that the starbyte-transformed images have visually demarcable regions. It is also clear that each protocol generates widely differing images. Changing the neighborhood size, while maintaining the same protocol, changes the texture and granularity markedly, without essentially changing the regions of similar shape (for example, compare Figure 2.4h, Figure 2.6h, Figure 2.8h and Figure 2.10h). Note also that the granularity increases with decreasing neighborhood size. •

For the starbyte images shown here, the bit ordering was exactly as listed in the various protocol definitions, with the least significant bit on top and the most significant bit at the bottom. The most significant bits dominate the image, and so the order in which we choose the bits makes an enormous difference in what we see. For example, protocols 3 and 8 appear to produce similar patterns (compare all “c” and “h” figures, and all “k” and “p” figures) even though protocol 3 is a 4-bit pattern while protocol 8 is an 8-bit pattern. This is because the dominant 4 bits in protocol 8 are indeed the 4 bits of protocol 3.

If we choose sectors in an order specified by their being as parallel as possible to a given direction, we effectively produce a directional filter. For example, the sixth protocol appears to produce a vertically directional starbyte pattern (see all “f” and “n” figures). This is because of the particular ordering of the bits. If the permutation of the bits were changed, the dominant direction would change. As of now it is not clear if there is an “optimal” permutation (analogous to the 1D Fast Fourier Transform bit-ordering used in [52]), which will give the best results.

An interesting observation is that protocol 7 applied on Figure 2.2a (Figure 2.4g, Figure 2.6g, Figure 2.8g, Figure 2.10g) and Figure 2.3a (Figure 2.5g, Figure 2.7g, Figure 2.9g, Figure 2.11g) enhances the microcalcifications. Thus, while it is not our intention to use starbyte patterns for object segmentation, in some cases they might serve this purpose.

Close examination of the starbyte pattern maps of the original and warped images for corresponding cases shows that the deformation relating the original and warped images is almost preserved in the starbyte transformation. In other words, starbyte enhancement and warping are effectively commutative. This is illustrated in Figure 2.13.

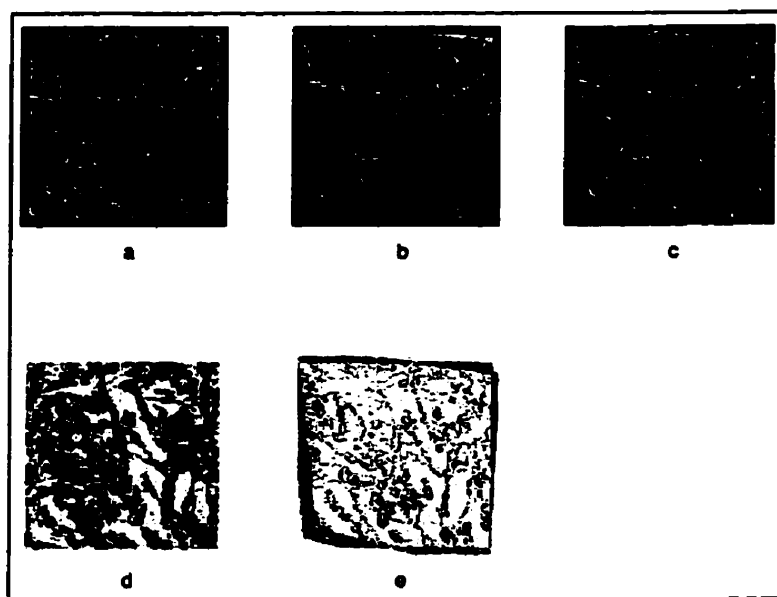


Figure 2.13: *a* and *b* are the same as Figure 2.6k and Figure 2.7k. *c* is the result of applying the same bilinear warp on *a* as was applied Figures 2.2 (to generate Figures 2.3). *d* shows the absolute difference between *a* and *b*, while *e* shows the absolute difference between *b* and *c*.

Figures 2.13a and b are the same as Figure 2.6k and Figure 2.7k (i.e. the 4-bit starbyte transformation on 9×9 neighborhoods using protocol 3 on Figure 2.2b and Figure 2.3b). Figure 2.13c is the result of applying the same bilinear warp on Figure 2.13a as was applied on Figures 2.2a and b (to generate Figures 2.3a and b). It is clear that Figure 2.13c is close to Figure 2.13b. Figure 2.13d is the absolute difference between Figures 2.13a and b, while Figure 2.13e is the absolute difference between Figures 2.13b and c. Since a 4-bit protocol has been used here, the difference images have values between 0 and 15. The difference images are shown such that they are white where the absolute difference is 0, and black where the difference is maximum (15), with graded gray levels in between. It is clear that most of the misregistration noise present in Figure 2.13d has been removed in Figure 2.13e.

Figure 2.14 (page 40) shows another example of the commutativity property of the starbyte transformation. Figures 2.14a and b are the same as Figure 2.6i and Figure 2.7i (i.e. the 4-bit starbyte transformation on 9×9 neighborhoods using protocol 1 on Figure 2.2b and Figure 2.3b). Figure 2.14c is the result of applying the same bilinear warp on Figure 2.14a as was applied on Figures 2.2a and b (to generate Figures 2.3a and b). It is clear that Figure 2.14c is close to Figure 2.14b. Figure 2.14d is the absolute difference between Figures 2.14a and b, while Figure 2.14e is the absolute difference between Figures 2.14b and c. These images are also shown such that they are white where the absolute difference is 0, and black where the difference is maximum, with graded gray levels in between. As in the previous example, it is clear that most of the misregistration noise present in Figure 2.14d has been removed in Figure 2.14e.

It is seen from Figure 2.13e and Figure 2.14e that the commutativity property holds only approximately. If S_t represents the starbyte transformation operation, and T represents the warp relating the given images, then $\|S_t(T(Reference)) -$

$T(S_t(Reference))$ is not zero. It is observed from Figure 2.13e and Figure 2.14e that the difference image contains non-zero values only at the border of every significant region. Since, the size and shape of a region depends on the neighborhood size selected to calculate the starbyte transformation and on the original images, as well as on the warp relating the original images, $\|S_t(T(Reference)) - T(S_t(Reference))\|$ depends on all these factors. However, again by observation of Figure 2.13e and Figure 2.14e, it is clear that the error at the border of a region is approximately of the order of a pixel.

From both these examples it is seen that the misregistration error is at the boundaries of the regions in the STs. This is because the boundaries are irregular. This issue is addressed in the next chapter, where we demonstrate that using the morphological binary open and close operation [40] smooths these boundaries.

It is obvious that Figure 2.2a and Figure 2.3a, and Figure 2.2b and Figure 2.3b cannot be matched visually because control points are almost impossible to discern. Since they also lack contrast or easily identifiable structure, a computerized method to automatically detect control points is difficult. The starbyte transformation converts the original images to texture maps. These texture maps contain visually demarcable regions, and since the original warp relating the two images is practically unaffected by the starbyte transformation, control points can be easily identified in the starbyte images, and these can be used to match the original image pair.

The potential of starbyte transformations for image registration can be seen by comparing Figures 2.13a and b as well as Figures 2.14a and b. It is easy to see that they are made up of several identifiable regions. The centroids of these regions can be used as control points [47].

Thus, in summary, the starbyte transformation has three important properties,

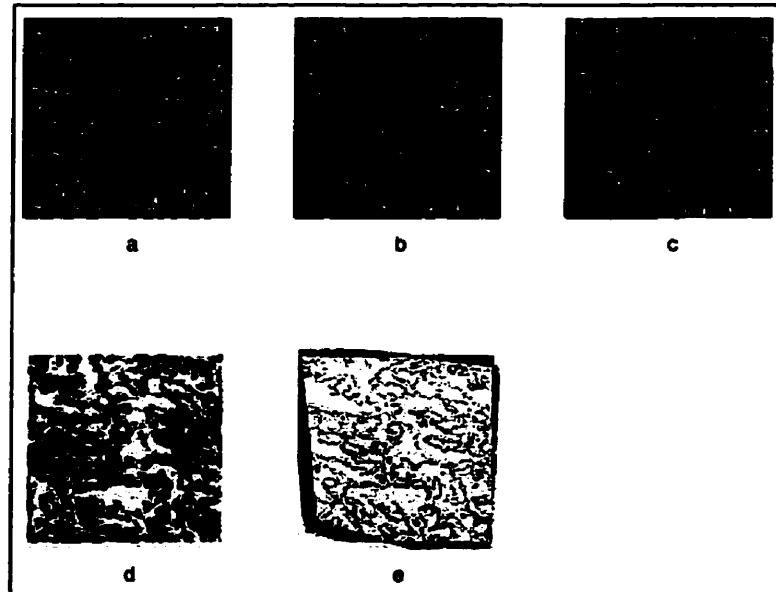


Figure 2.14: *a* and *b* are the same as Figure 2.6i and Figure 2.7i. *c* is the result of applying the same bilinear warp on *a* as was applied Figures 2.2 (to generate Figures 2.3). *d* shows the absolute difference between *a* and *b*, while *e* shows the absolute difference between *b* and *c*.

which work together to render the starbyte-transformed image pair more registrable than the original image pair. These properties are :

1. While the original images get transformed to another domain, the distortion relating the images remains undisturbed so that registering the images in the new domain is equivalent to registering the original images.
2. Information in the original images is simplified by the starbyte transformation because information is essentially classified into various “labels”, so that while the new images contain less information, portions of the two images can be identified or matched with each other because of the labels.
3. For many starbyte transformations, there are labels that clump into discrete regions that can be readily equated in a pair of images.

It is possible to generalize the starbyte transformation concept by using other properties of the local neighborhood of a pixel. For e.g., the transformation could be viewed as applying spatial masks over different sectors in the neighborhood of a pixel. Weights could be assigned to the spatial masks to obtain differential operators which could be used to enhance features in selected directions [40]. Other properties at a pixel can be used to generate the starbyte values, for e.g., the sectors could be annular regions in the centered Discrete Fourier Transform (DFT) of the local neighborhood of a pixel, so that the bit pattern could be generated by comparing the energies in these annular regions to the total energy in the DFT. Else, the starbyte of a pixel could just represent the average energy in a specified window of the DFT of the neighborhood of the pixel. Any other local decomposition by basis functions, such as Zernike polynomials [125] could be thresholded to produce a transformation.

Alternatively, other models or representations used in texture analysis could be used here [55]. The range images in [25] could be used as texture transformations. The starbyte at a pixel could also represent textural energy calculated using one of Laws's kernels [55], [67].

It would be interesting to see how many of these other different methods will preserve all three properties of the starbyte transformation. However, for the purpose of this thesis, we have generated the starbyte-transformed images using only a few of the spatial protocols described in this chapter.

Selection of control points from the STs and determination of the correspondence between them is discussed in the 4th and 5th chapter.

2.3.2 Starbyte-transformed images for other types of distortions

Until now, we have presented the starbyte-transformed images for one transformation, namely the bilinear transformation given by equation (2.1). We will now present the starbyte-transformed images for other transformations. These include simple transformations like translation, rotation and scaling, as well as more complicated transformations created using thin plate splines and a sinusoidal distortion.

Figure 2.15a is the same as Figure 2.2b. Figure 2.15b was obtained by translating Figure 2.2b by 8 pixels row-wise (upwards) and 10 pixels column-wise (leftwards). Figure 2.15c was obtained by rotating Figure 2.2b by 10 deg anti-clockwise. Figure 2.15d was obtained by scaling Figure 2.2b by 10 %. Figures 2.16a to h show 8 starbyte-transformed images of Figure 2.15b for the 8 protocols, using a neighborhood size of 9×9 while Figures 2.16i to p show the same for a neighborhood size of 15×15 . Figures 2.17a to h show 8 starbyte-transformed images of Figure 2.15b for the 8 protocols, using a neighborhood size of 9×9 while Figures 2.17i to p show the same for a neighborhood size of 15×15 . Figures 2.18a to h show 8 starbyte-transformed images of Figure 2.15b for the 8 protocols, using a neighborhood size of 9×9 while Figures 2.18i to p show the same for a neighborhood size of 15×15 . The next few pages show Figures 2.15 to Figures 2.18. The text continues on page 47.

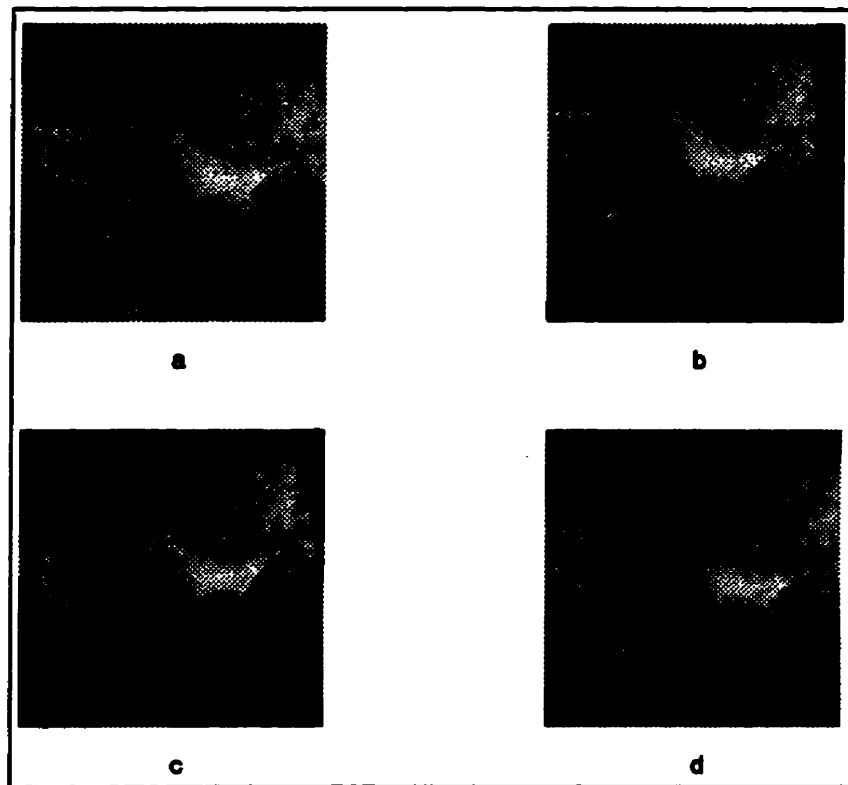


Figure 2.15: *a* is the same as Figure 2.2b. *b*: *a* translated by 8 pixels upwards and 10 pixels leftwards. *c*: *a* rotated by 10 deg anti-clockwise. *d*: *a* enlarged by 10 %.

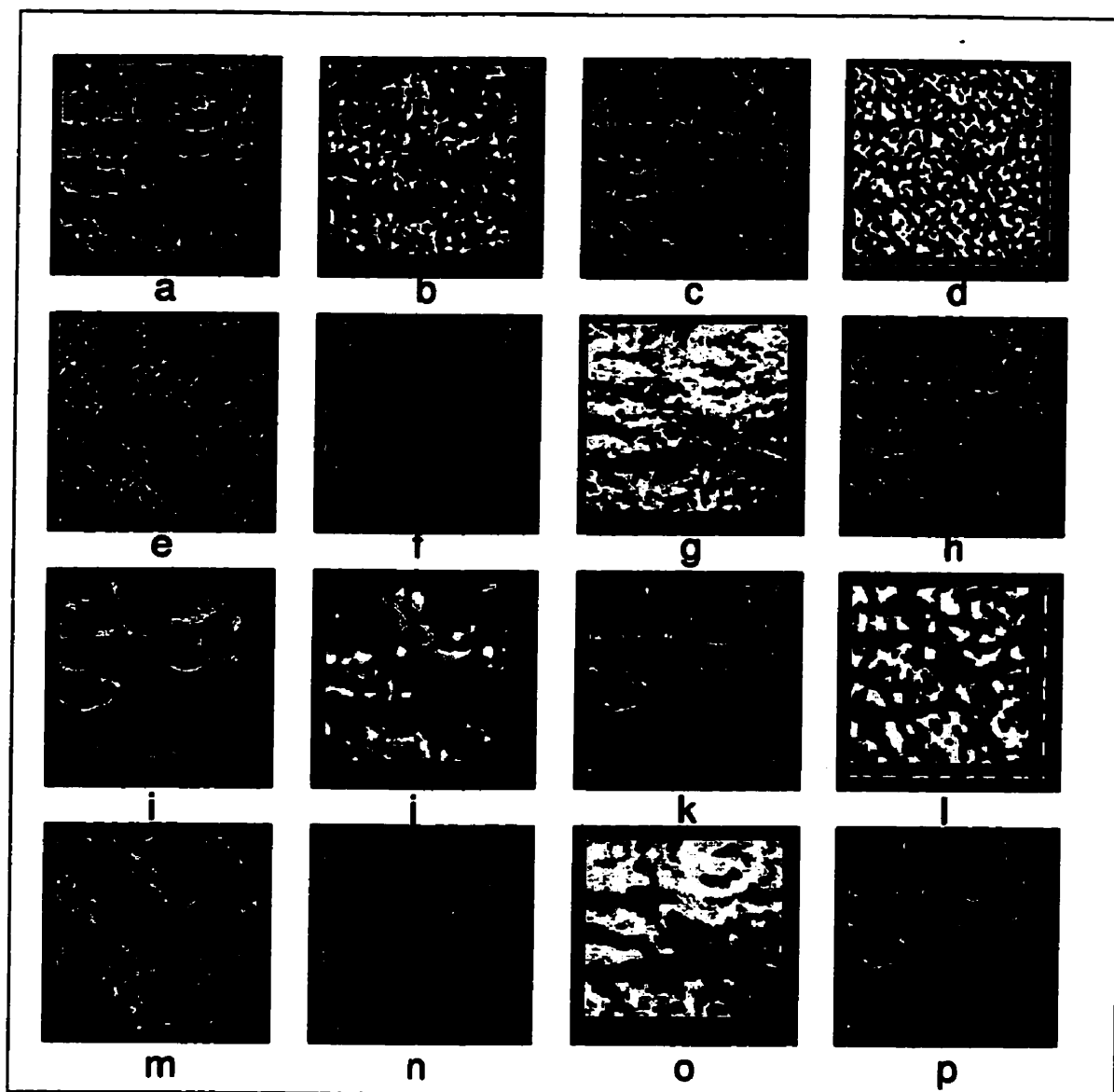


Figure 2.16: *a – h*: Starbyte transformed images of Figure 2.15b (8 pixels upward- and 10 pixels leftward-translated version of Figure 2.15a), using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.15b, using protocols 1 to 8, for a 15×15 neighborhood.

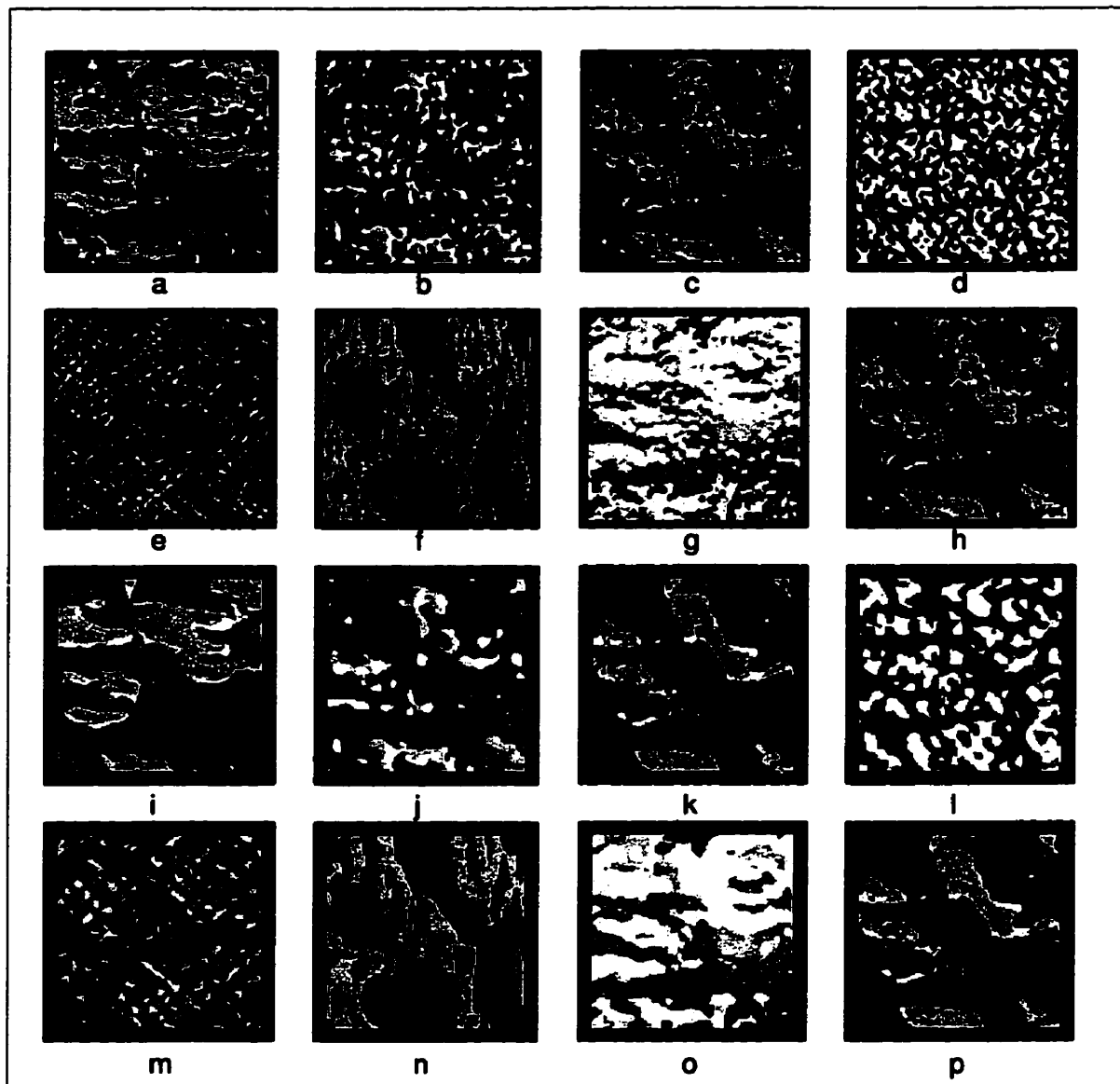


Figure 2.17: *a – h*: Starbyte transformed images of Figure 2.15c (10 deg anti-clockwise rotated version of Figure 2.15a) , using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.15c, using protocols 1 to 8, for a 15×15 neighborhood.

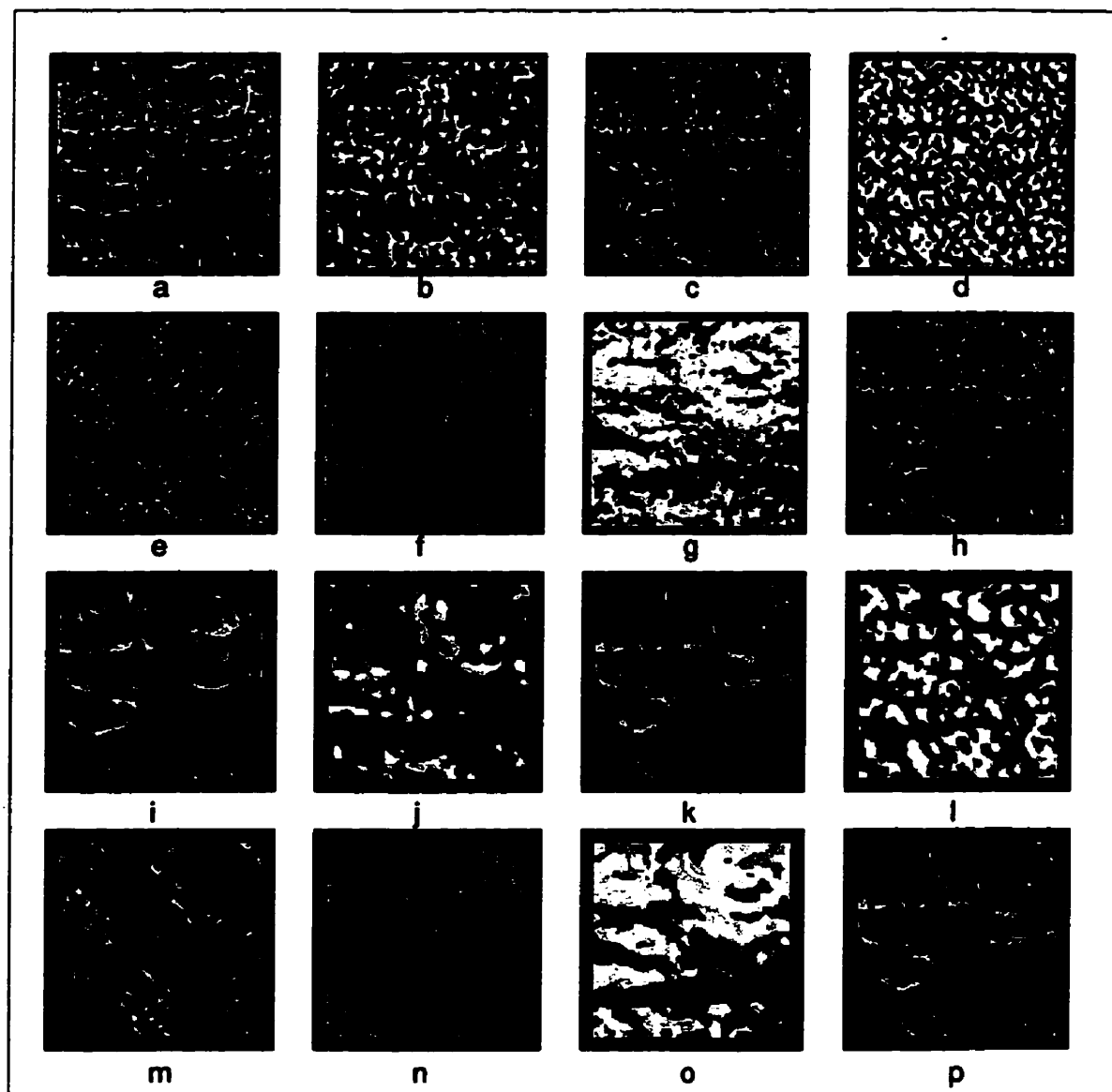


Figure 2.18: *a – h*: Starbyte transformed images of Figure 2.15d (10% expanded version of Figure 2.15a), using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.15d, using protocols 1 to 8, for a 15×15 neighborhood.

Figure 2.19a is the same as Figure 2.2b. Figure 2.19b was obtained by distorting Figure 2.19a using thin plate splines [8]. The distortion was calculated by first specifying corresponding points in the reference and target images. The target image (Figure 2.19b) was then calculated from the original image (Figure 2.2b) using the equations in [8]. The map of corresponding points is given in Figure 2.20, where points in the reference image are denoted by the '+' symbol, while points in the target image are denoted by the 'o' symbol. From Figure 2.20, it is clear that the boundary of the reference image maps to the boundary of the target image.

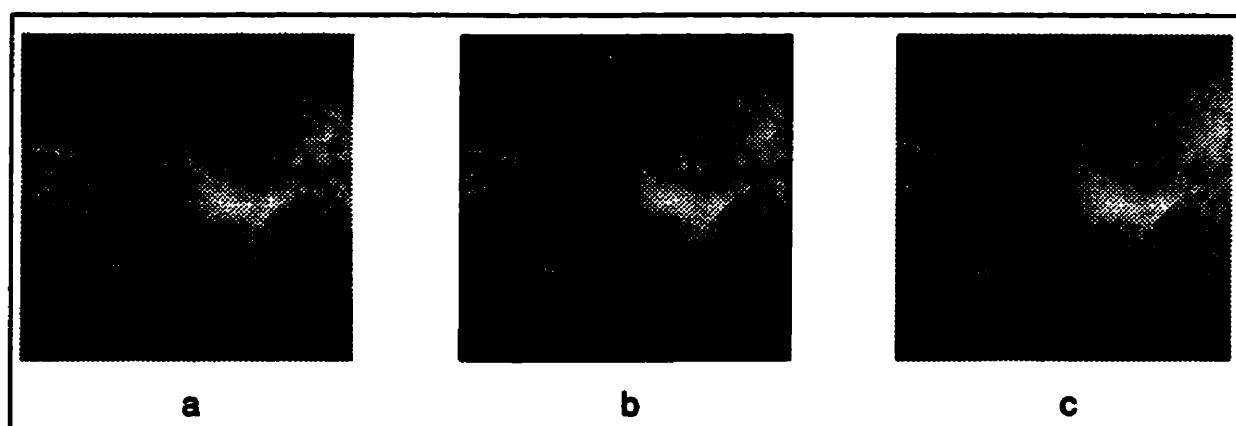


Figure 2.19: *a*: Same as Figure 2.2b. *b*: Distorting *a* using thin plate splines (map of corresponding points given in Figure 2.20a). *c*: Distorting *a* using the sinusoidal distortion given by equation (2.2).

Figure 2.19c was obtained by distorting Figure 2.19a using the following sinusoidal distortion:

$$\begin{aligned}\hat{x} &= x + 0.06 \sin \pi x \sin 3\pi y \\ \hat{y} &= y - 0.05 \sin 2\pi x \sin \pi y\end{aligned}\tag{2.2}$$

As in the TPS distortion, the boundary of the reference image maps on to the boundary of the target image.

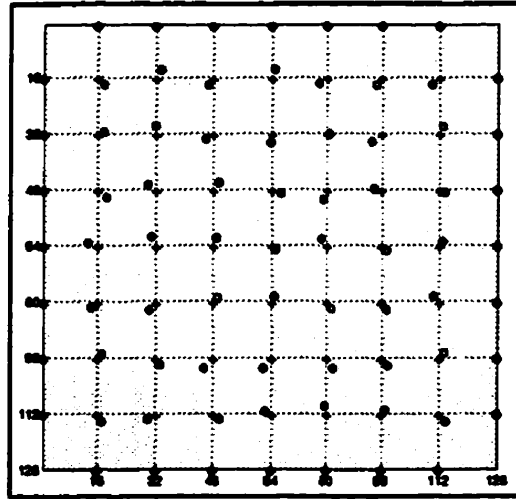


Figure 2.20: Map of corresponding points for the distortion between Figures 2.19a and b. '+' denotes points in the reference image. 'o' denotes points in the target image.

Figures 2.21a to h and Figures 2.21i to p (page 49) show 8 starbyte-transformed images of Figure 2.19b for the 8 protocols for 9×9 and 15×15 neighborhoods respectively. Figures 2.22 (page 50) shows the same for Figure 2.19c.

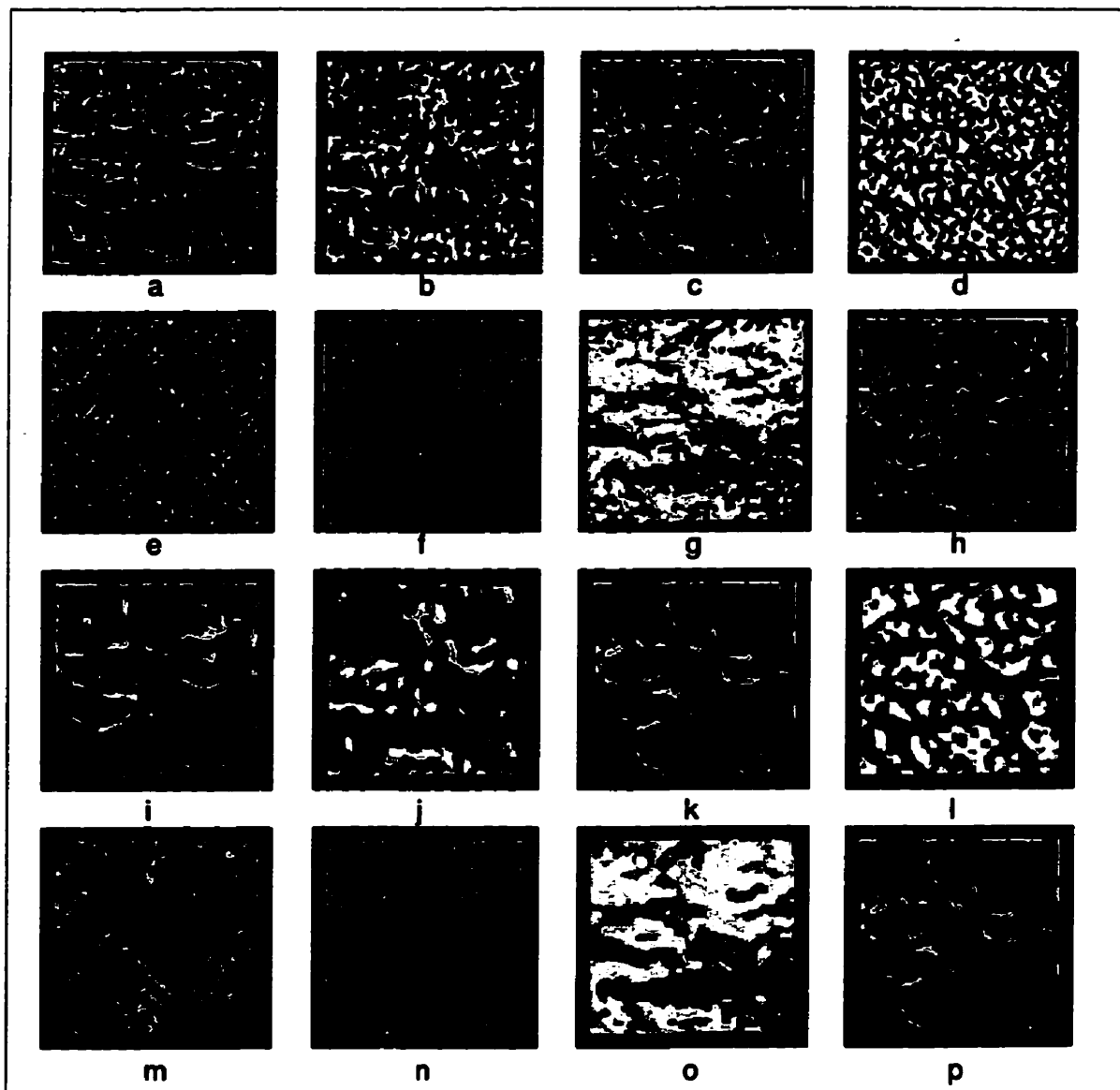


Figure 2.21: *a – h*: Starbyte transformed images of Figure 2.19b (TPS-distorted version of Figure 2.2b), using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.19b, using protocols 1 to 8, for a 15×15 neighborhood.

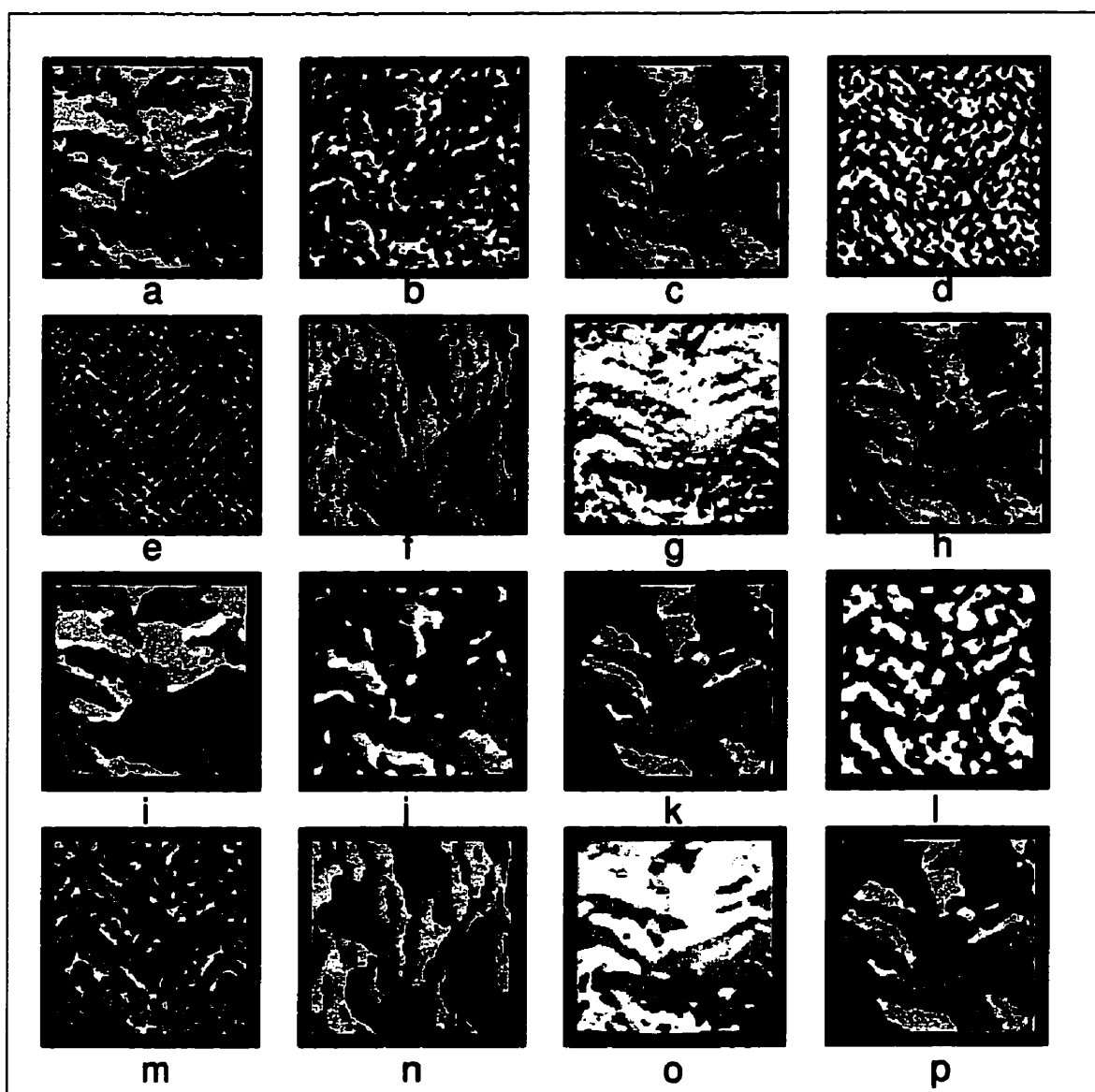


Figure 2.22: *a – h*: Starbyte transformed images of Figure 2.19c (sinusoidally-distorted version of Figure 2.2b), using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.19c, using protocols 1 to 8, for a 15×15 neighborhood.

It can be seen from Figure 2.16, Figure 2.17, Figure 2.18, as well as, Figure 2.21 and Figure 2.22, that regions present in the corresponding starbyte-transformed images of Figure 2.2b (Figures 2.6i to p and Figures 2.8i to p) are present in the corresponding starbyte-transformed images for all the transformations considered above and that the commutativity property still holds.

The six distortions considered so far have been shown in Figure 2.23 as grid distortions. Figure 2.23a depicts the bilinear transformation given by equation (2.1), Figures 2.23b to d depicts translation (8 pixels upwards and 10 pixels leftwards), rotations (10 deg counter-clockwise) and scaling (10 % enlargement). Figure 2.23e shows the deformation obtained using thin plate splines, while Figure 2.23f denotes the transformation given by equation (2.2).

2.3.3 Starbyte-transformed images for a mammogram with a different texture

We now present starbyte transformations for a mammogram with a texture quite different from the one considered so far (Patient # 12, CC view, [86]) and demonstrate that regions in the starbyte image of the original are easily identifiable in the starbyte image of the transformed.

Figures 2.24a and b (page 53) show two regions of size 128×128 cropped out of the image "c12c.ima" in [86], Figure 2.24a containing microcalcifications while Figure 2.24b does not contain microcalcifications. Figures 2.24c and d were obtained by warping Figures 2.24a and b using the bilinear transformation described by equation (2.1). Figures 2.24e and f were obtained by warping Figures 2.24a and b using the transformation described by equation (2.2).

Figures 2.25a to h and i to p show the 8 starbyte-transformed images for the 8 protocols for 9×9 neighborhoods for Figure 2.24a and Figure 2.24b respectively. Figures 2.26a

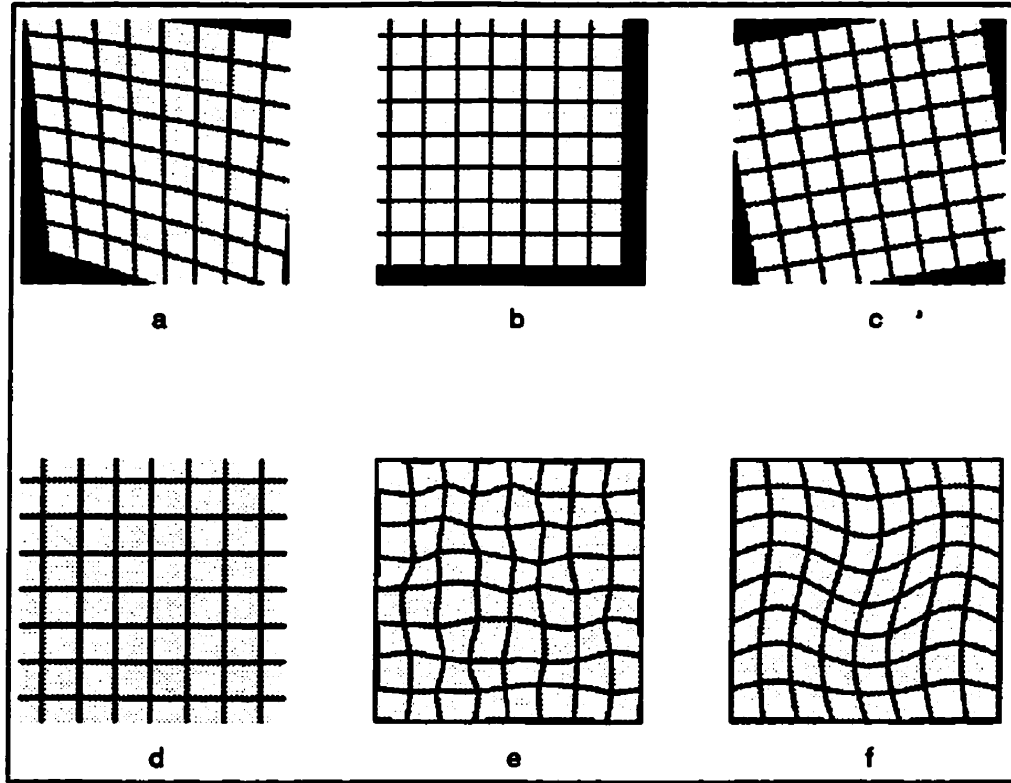


Figure 2.23: Distortions used: *a*: Bilinear given by equation (2.1). *b*: Translation (8 pixels upwards and 10 pixels leftwards). *c*: Rotation (10 deg counter-clockwise). *d*: Scaling (10 % enlargement). *e*: TPS distortion. *f*: Sinusoidal distortion given by equation (2.2).

to h and i to p show the same for Figure 2.24c and Figure 2.24d. Figures 2.27a to h and i to p show the same for Figure 2.24e and Figure 2.24f.

Figures 2.28a to h and i to p show the 8 starbyte-transformed images for the 8 protocols for 15×15 neighborhoods for Figure 2.24a and Figure 2.24b respectively. Figures 2.29a to h and i to p show the same for Figure 2.24c and Figure 2.24d. Figures 2.30a to h and i to p show the same for Figure 2.24e and Figure 2.24f. The next six pages show Figures 2.25 to Figures 2.30. The text continues on page 60.

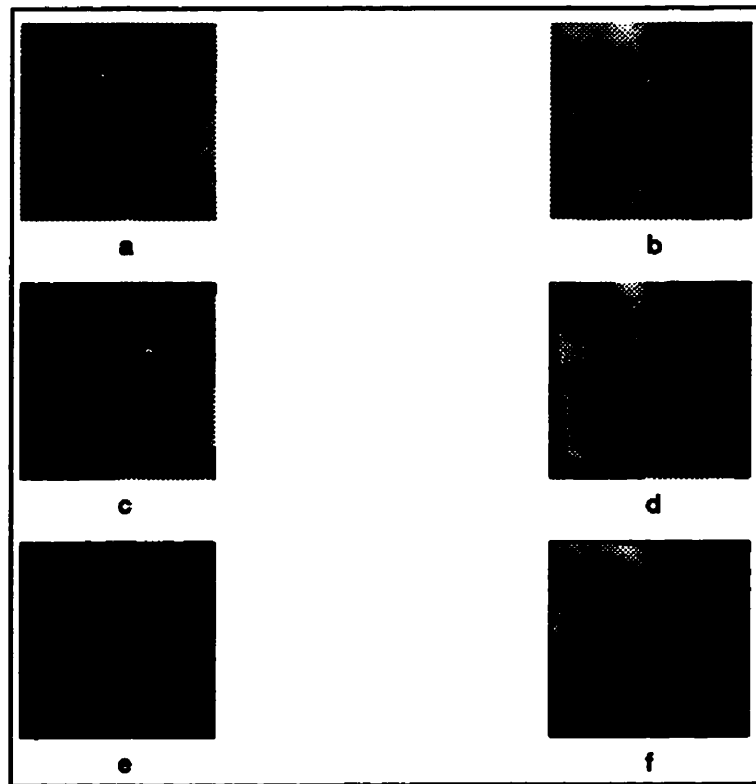


Figure 2.24: *a* and *b*: 128×128 regions cropped out of the image “c12c.ima” in [86] with and without microcalcifications. *c* and *d*: *a* and *b* warped using equation (2.1). *e* and *f*: *a* and *b* warped using the transformation depicted in Figure 2.20c.

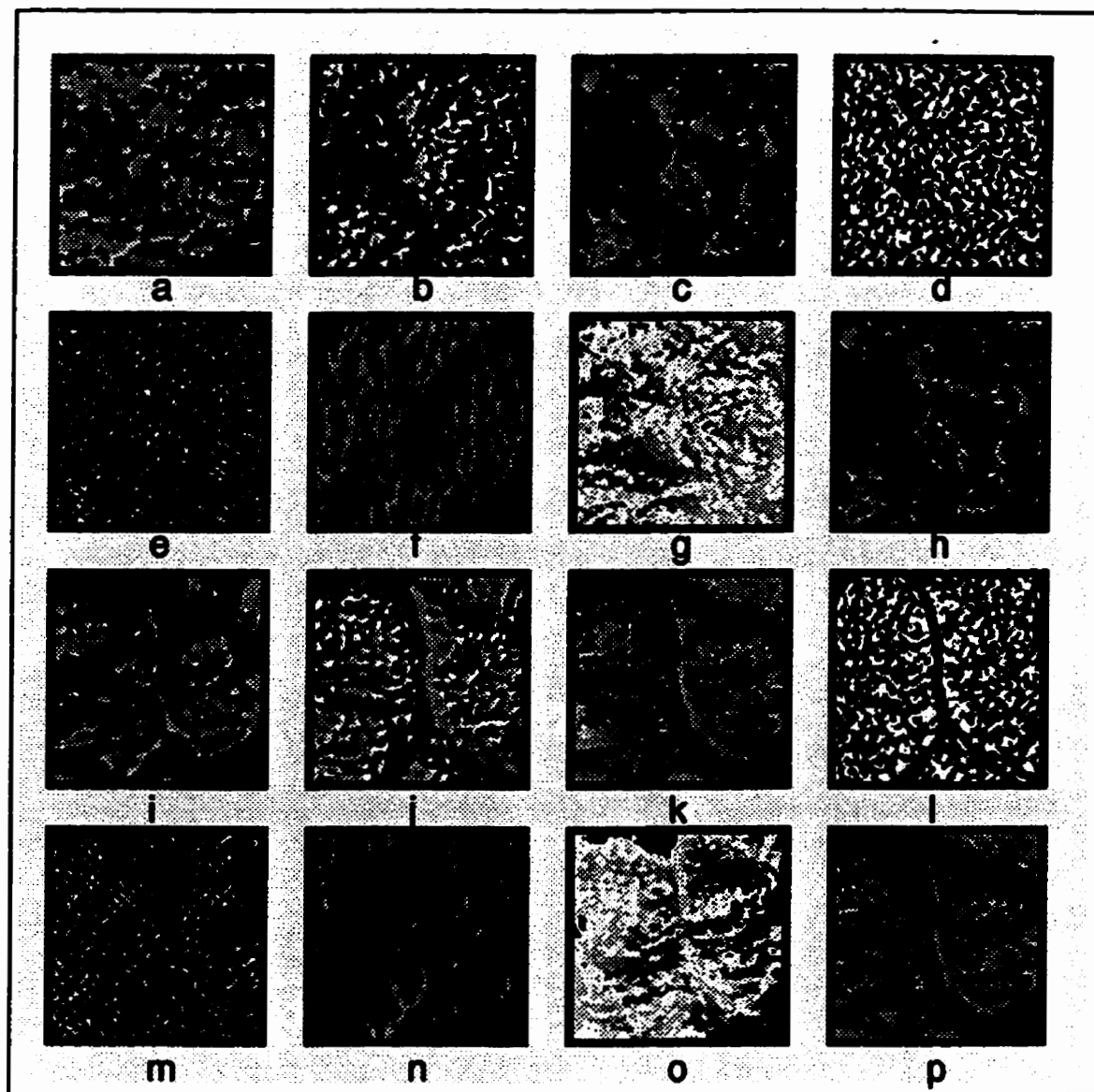


Figure 2.25: *a – h*: Starbyte transformed images of Figure 2.24a, using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24b, using protocols 1 to 8, for a 9×9 neighborhood.

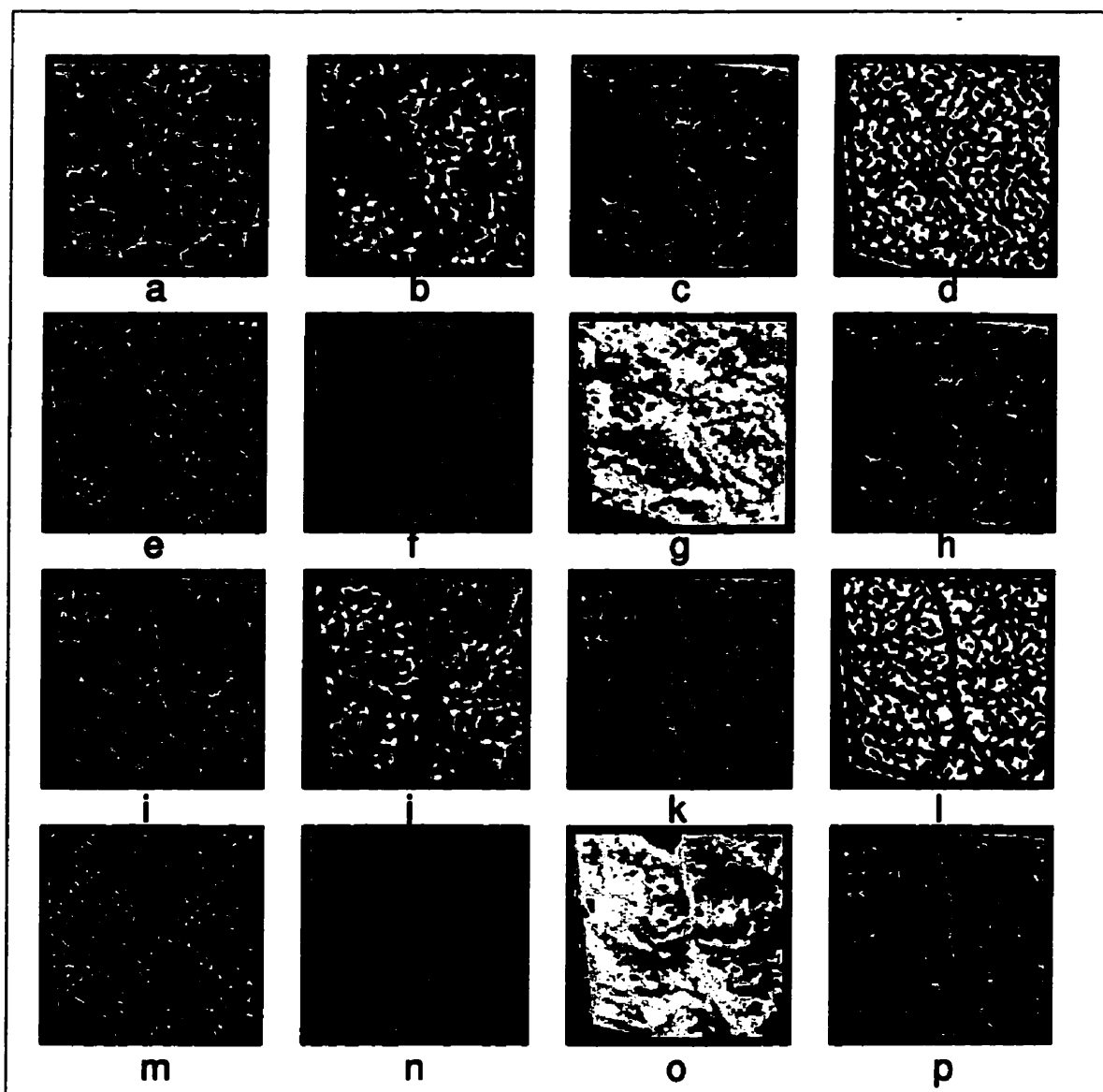


Figure 2.26: *a – h*: Starbyte transformed images of Figure 2.24c, using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24d, using protocols 1 to 8, for a 9×9 neighborhood.

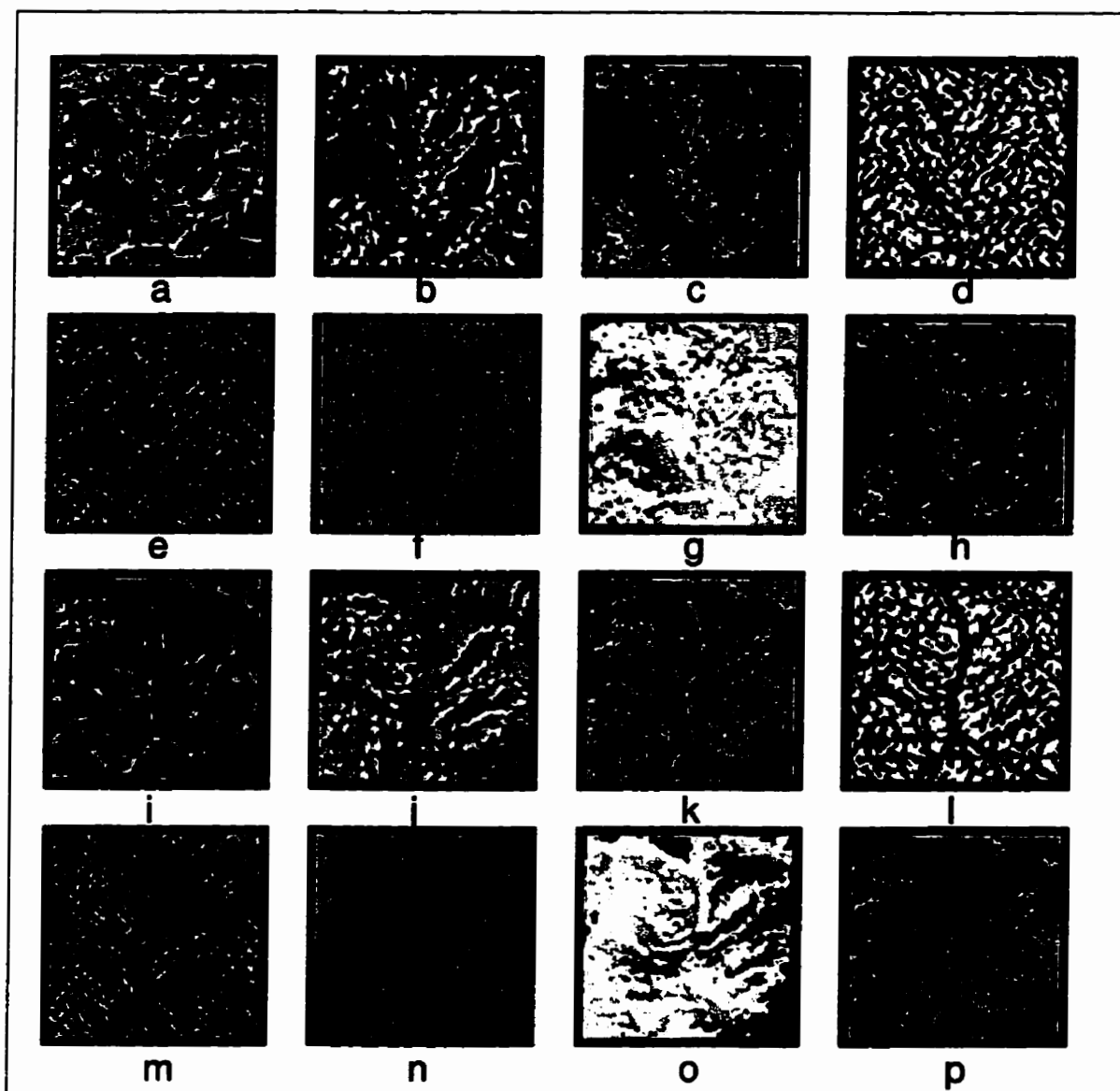


Figure 2.27: *a – h*: Starbyte transformed images of Figure 2.24e, using protocols 1 to 8, for a 9×9 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24f, using protocols 1 to 8, for a 9×9 neighborhood.

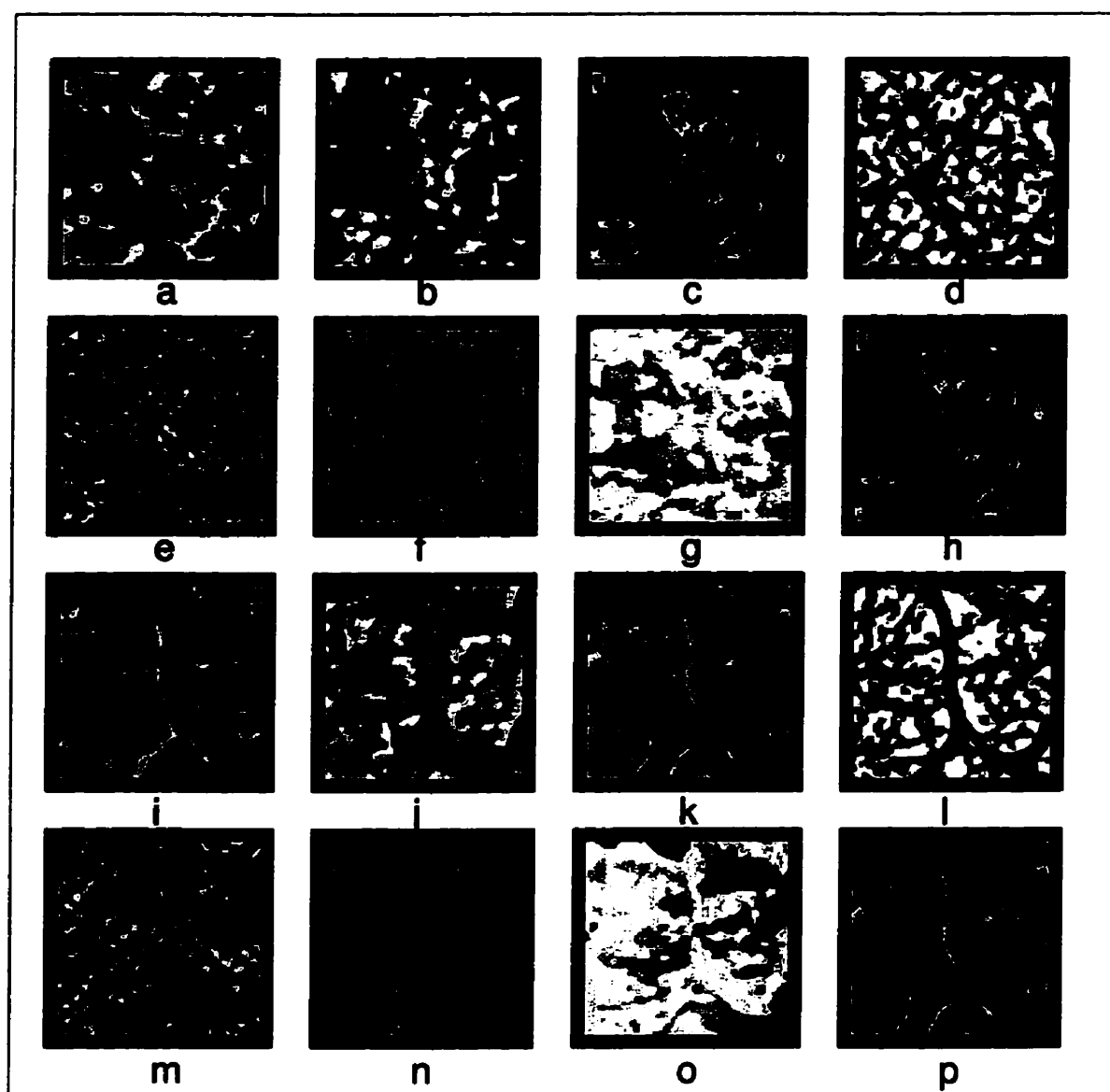


Figure 2.28: *a – h*: Starbyte transformed images of Figure 2.24a, using protocols 1 to 8, for a 15×15 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24b, using protocols 1 to 8, for a 15×15 neighborhood.

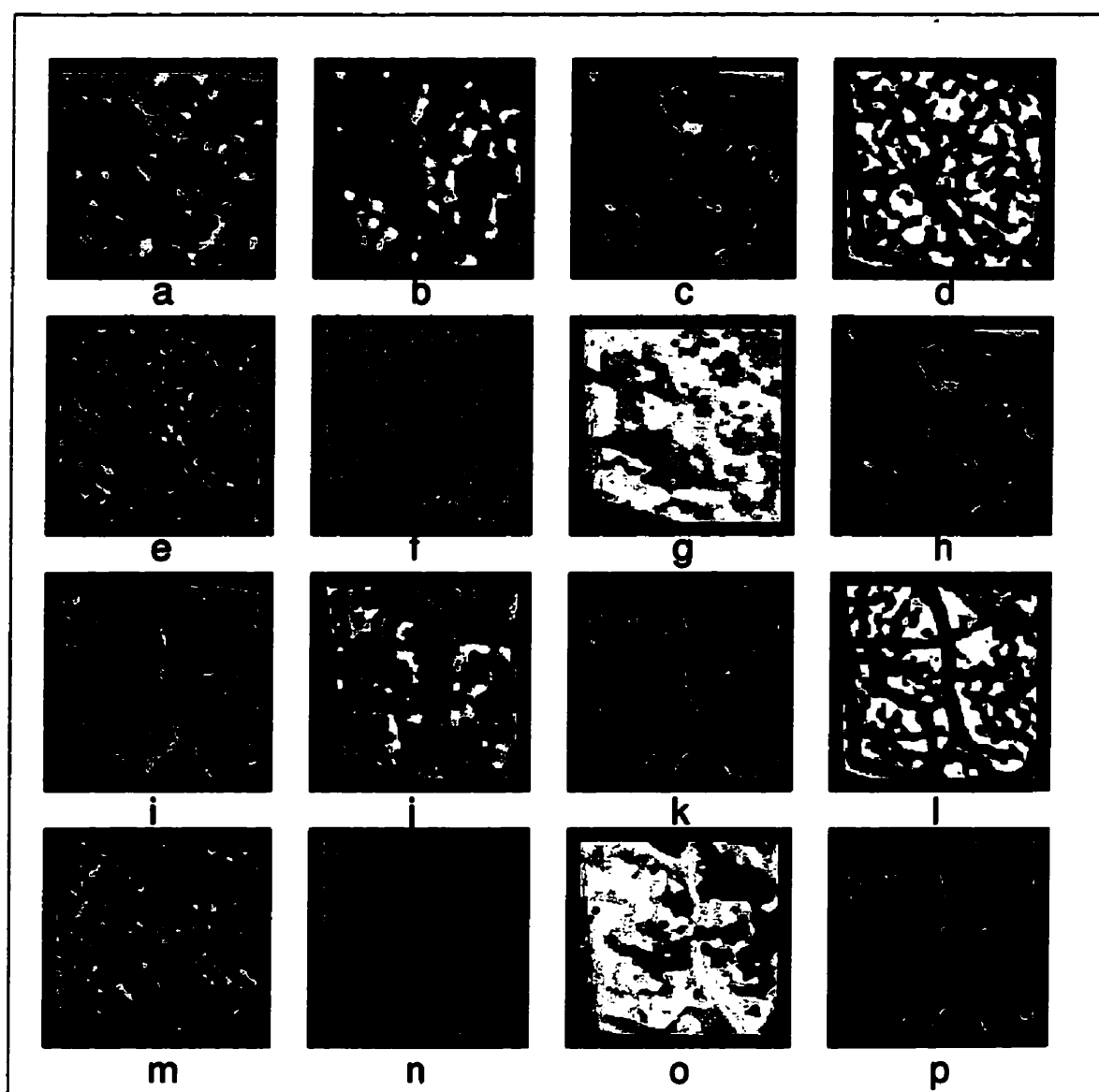


Figure 2.29: *a – h*: Starbyte transformed images of Figure 2.24c, using protocols 1 to 8, for a 15×15 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24d, using protocols 1 to 8, for a 15×15 neighborhood.

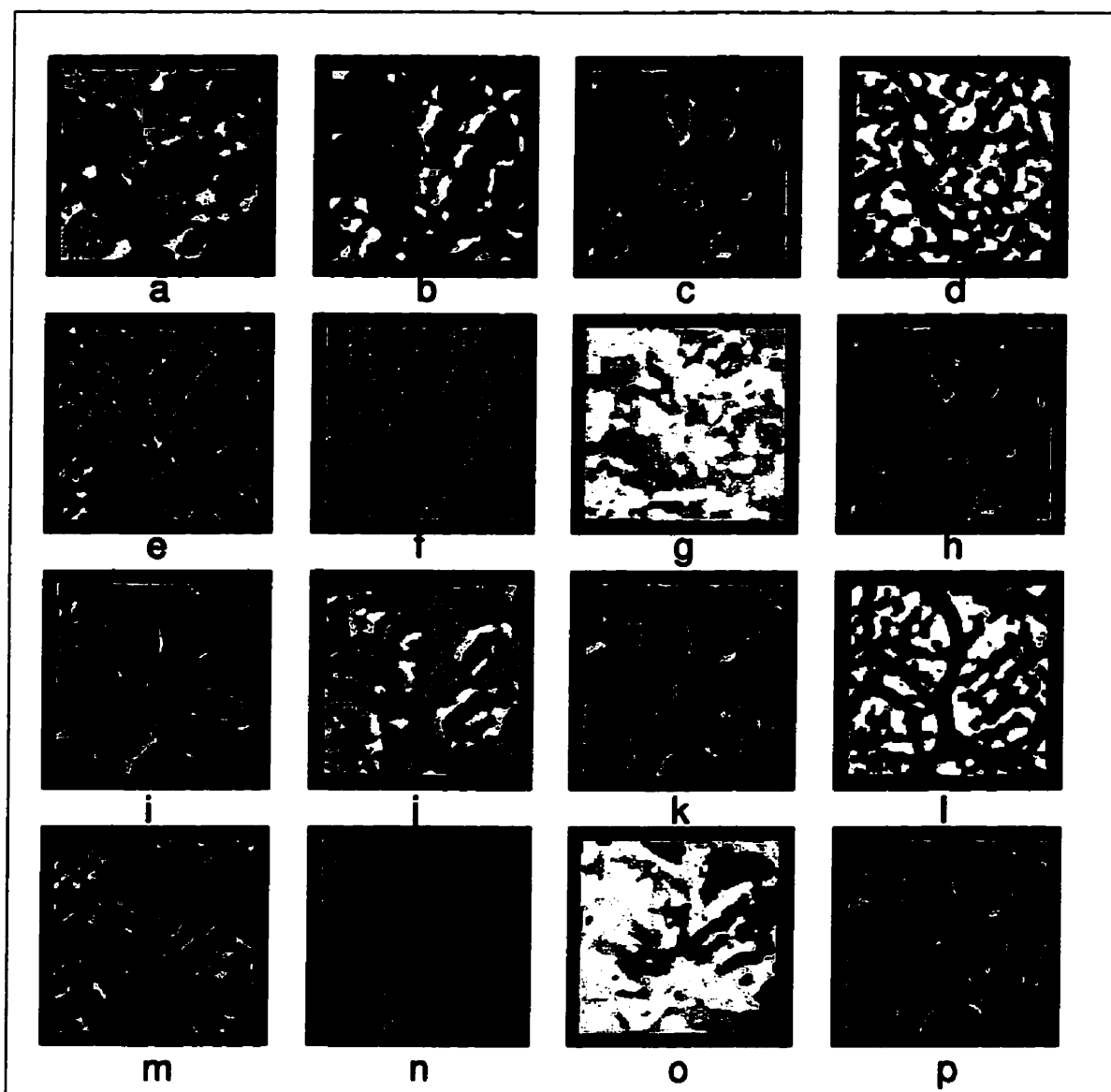


Figure 2.30: *a – h*: Starbyte transformed images of Figure 2.24e, using protocols 1 to 8, for a 15×15 neighborhood. *i – p*: Starbyte transformed images of Figure 2.24f, using protocols 1 to 8, for a 15×15 neighborhood.

Even in these cases, it is obvious that the starbyte transformation converts the original images to texture maps with clearly identifiable regions which can be matched up visually in corresponding images.

Starbyte images can thus be used to generate a large number of control points, but working with such a large set of points in each image, makes matching corresponding points in the two images a computationally intensive problem. This problem can be overcome by obtaining binary maps by density slicing any one starbyte image at different grey levels. This generates binary images which are independent of each other. The independent pairs of binary maps can then be used to generate manageable sets of control points. These independent sets of control points can then be pooled together. Once enough corresponding point pairs are successfully established, unwarping can be performed by interpolation using thin plate splines.

In the next chapter, we shall discuss density slicing and region-growing. We shall also demonstrate numerically that the centroids of the regions extracted are good control points.

2.4 Conclusions

Image registration is an important problem in breast imaging. The matching of breast images directly is difficult, because of the low contrast and noisy nature of the images. This makes identification of control points extremely difficult. We propose a new textural transformation called the "starbyte transformation", which when applied to breast images, converts them to texture maps with clearly demarcable regions, where control points can be more easily identified both visually as well as by computer techniques. We present the application of these starbyte transformations on regions cropped out of two mammograms of different textural characteristics. We have

considered regions with and without microcalcifications. These starbyte images have been obtained for six different types of distortions, namely a bilinear warp, translation, rotation, scaling, a TPS distortion and a sinusoidal distortion. We demonstrate that deformation of the original images is effectively commutative with the starbyte transformation. Due to this property, matching of the transformed images is equivalent to matching of the original images. The results demonstrate the potential of the starbyte transformation and its applicability to breast image registration.

Chapter 3

DENSITY SLICING AND REGION GROWING

3.1 Introduction

In the previous chapter, we discussed the starbyte transformation in detail. Given two images to be registered (a reference and a target), we demonstrated that the starbyte transformation converts them to images with clearly demarcable regions. These regions can be easily recognized and matched up in the two images. We demonstrated that the “starbyte” transformation and the warp or distortion relating the two images are almost commutative so that matching of the starbyte-transformed images is equivalent to matching the original images. We noted that the centroids of the different regions could be used as control points. But we said that, since the starbyte transformation results in the generation of a large number of regions at different density levels, working with all the regions of different density levels simultaneously yields a large set of control points in both the images and would make matching corresponding points in the two images a computationally intensive problem.

When an image is distorted, points in the original image are not relocated in the distorted image in an isolated way. In other words, it is not the point alone which moves to the new location, but a small neighborhood around the point will also be carried to the new location. In other words, local neighborhoods of a point in the original image will continue to be local neighborhoods of the same point in

the distorted image because of continuity considerations. Of course, this continuity property will break down in cases of severe distortions, where regions are “torn” apart so severely that we cannot make any assumptions about neighborhoods being preserved. But, for reasonable distortions, continuity properties hold.

Since the starbyte value at a pixel is calculated using its local neighborhood, regions having the same density value in the starbyte images are actually regions belonging to the same neighborhood. These also denote areas in the original image having the same texture. Thus regions having a certain density value in the starbyte image of the reference will have the same density value in the starbyte image of the target. Thus, rather than trying to perform the registration on all the regions (having different density values) simultaneously, it is better to treat each density value independently from the other values. In the previous chapter, we demonstrated that even simple 4-bit protocols (1 and 3) generate starbyte patterns with sufficient detail, so that corresponding regions in the two images can be visually identified. A 4-bit protocol results in 16 independent density slices. Computation-wise, this is a manageable number of slices.

For many of the examples in the subsequent chapters, we have considered only 4-bit protocols. Hence, if we density slice the starbyte images of the reference and target at these 16 different density levels (for e.g. obtained using protocol 3), we will get 16 pairs of independent images. Control points from each pair can be obtained independently of the other pairs. These control points can then be pooled together.

In this chapter, we shall display the starbyte images obtained using protocol 1 and 3 at different slices for the bilinear transformation considered in the previous chapter. The raw density slices are very noisy. But they can be cleaned up effectively using mathematical morphological operations. In fact, a simple binary open and close operation together form an excellent filter. For simplicity of notation, we shall refer to

a binary open and close morphological operation as an “OC” operation. Once these slices are “cleaned up”, the centroids of the different regions are obtained. We shall demonstrate, by a few examples in this chapter, that the centroids are good control points.

3.2 Raw Density Slicing

Given an image, a *density slice* at a particular density or gray level is a binary image in which all image pixels having that particular density level are “white” or 255 and the rest of the image pixels are “black” or 0. This is different from *thresholding* an image [40]. In a *thresholded* image, values between two density levels (lower and upper threshold levels) are “whitened”, while all other pixels are set to 0. Hence density slicing is a special case of thresholding, where the upper and lower threshold levels are equal. Figures 3.1a and b show the starbyte images corresponding to the images Figure 2.2b and Figure 2.3b of the previous chapter. Figure 3.1a and Figure 3.1b are the same as Figure 2.6k and Figure 2.7k respectively of the previous chapter (and were obtained by applying protocol 3 with a 9 x 9 neighborhood size on Figure 2.2b and Figure 2.3b respectively). As was mentioned earlier, the 2 original images are related by the bilinear transformation given by equation (2.1). Figures 3.2a to p (page 66) show Figure 3.1a density sliced at the density levels from 0 to 15, while Figures 3.3a to p show Figure 3.1b at the same density levels.

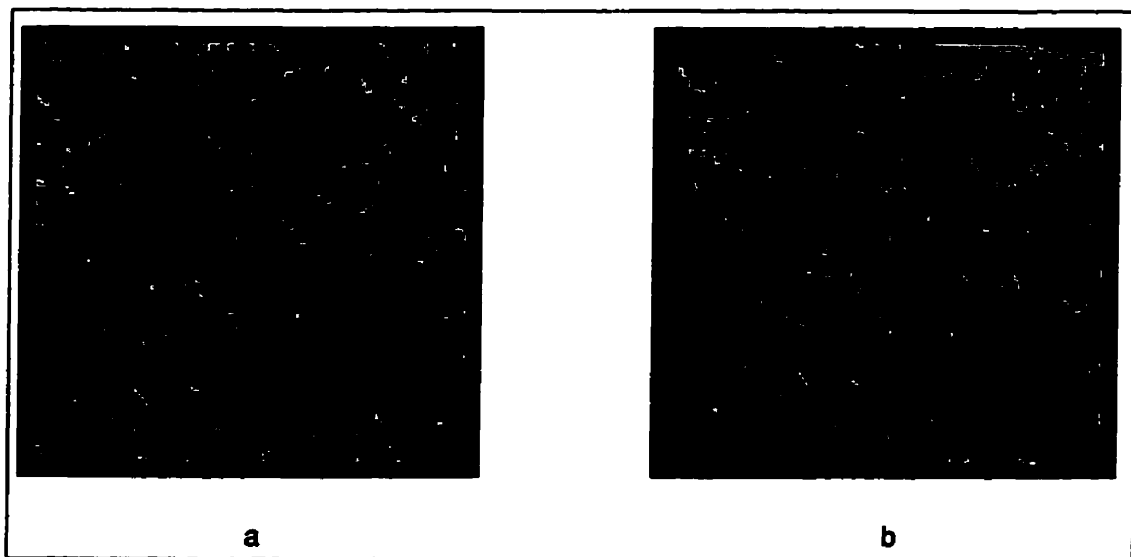


Figure 3.1: *a* and *b*: Starbyte images of Figure 2.2b and Figure 2.3b obtained by applying protocol 3 with a 9×9 neighborhood size.

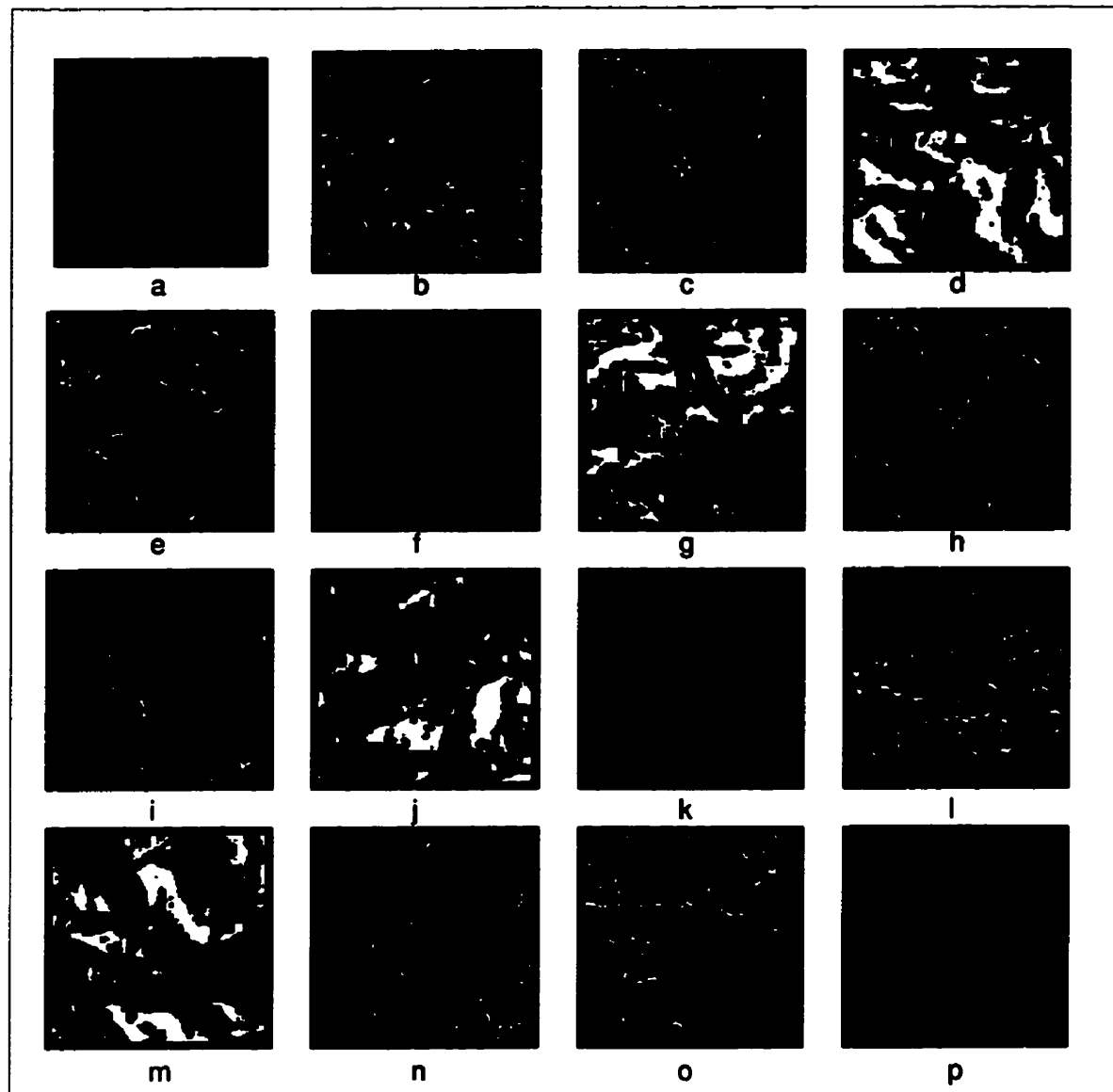


Figure 3.2: *a* to *p*: Figure 3.1a density sliced at the levels from 0 to 15.

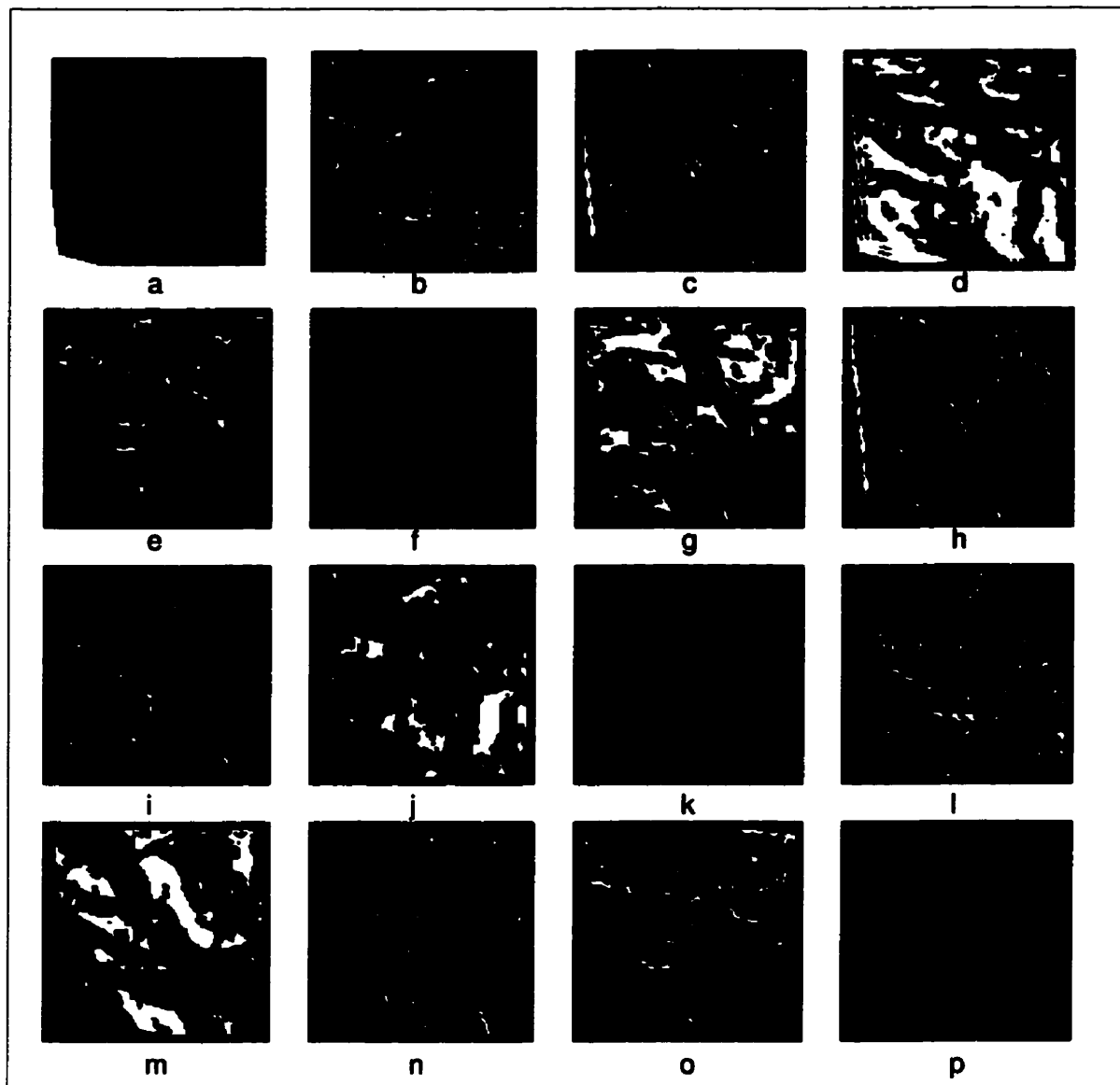


Figure 3.3: *a* to *p*: Figure 3.1b density sliced at the levels from 0 to 15.

The density slices in Figure 3.2 and Figure 3.3 reveal an interesting arrangement of patterns. It is seen that the images along the diagonals have the greatest information. This can be explained in the following way:

Protocol 3 was used to calculate the starbyte transformation for these images. Each bit was generated by comparing the average over the whole neighborhood of a pixel against averages over 4 sectors created by dividing the neighborhood across its diagonals. Since the textural properties of the original image are continuous, if one of these bits evaluate to 1, it is more likely that the bit corresponding to the neighboring sector will also evaluate to 1, rather than that corresponding to a non-adjacent sector. However, because the textural properties do change over the area of the image, it is more improbable for more than 2 bits to be set to 1 for a large number of pixels in the image. Thus it is more probable that those binary slices in the starbyte-transformed image, which have two adjacent bits set to 1, have more information than the other binary slices. An observation of Figure 3.2 and Figure 3.3 shows that significant information is present in the images along the diagonal. These represent the slices 3, 6, 9 and 12, which in hexadecimal format are given by "0011", "0110", "1001" and "1100", all of which correspond to two adjacent bits set to 1. If the bits had been arranged in a different way (for e.g. if adjacent bits did not correspond to adjacent sectors), then the binary slices containing significant information would be ordered differently.

As can be seen, these images are extremely noisy. They contain several isolated pixels as well as groups of noise pixels scattered throughout the image area. Also the boundaries of the large regions (which can be used to obtain control points) are very irregular. These density-sliced images can be cleaned up using an OC operation.

We shall give a brief description of morphological operations below. This description has been adapted from [40].

3.3 Morphological operations

3.3.1 Some basic definitions

Let us first define some basic set operations. Let A and B be sets in Z^2 . The *translation* of A by x , denoted by $(A)_x$ is defined as

$$(A)_x = \{c \mid c = a + x, \text{ for } a \in A\}$$

The *reflection* of B , denoted by \hat{B} , is defined as

$$\hat{B} = \{x \mid x = -b, \text{ for } b \in B\}$$

The *complement* of set A is

$$A^c = \{x \mid x \ni A\}$$

The *difference* of two sets A and B , is given by

$$A - B = \{x \mid x \in A, x \ni B\} = A \cap B^c$$

3.3.2 Dilation

The *dilation* of A by B , denoted by $A \oplus B$, is defined as

$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \emptyset\}$$

Thus the dilation process consists of obtaining the reflection of B about its origin and then shifting this reflection by x . The dilation of A by B then is the set of all x displacements such that \hat{B} and A overlap by at least one non-zero element. Set B is commonly referred to as the *structuring element* in all morphological operations.

3.3.3 Erosion

The *erosion* of A by B , denoted by $A \ominus B$, is defined as

$$A \ominus B = \{x \mid (B)_x \subseteq A\}$$

Thus, the erosion of A by B is the set of all points x such that B , translated by x , is contained in A .

Dilation thus expands an image and erosion shrinks it. Dilation and erosion are duals of each other with respect to set complementation and reflection. That is,

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

3.3.4 Opening and closing

Opening generally smooths the contour of an image, breaks narrow isthmuses, and eliminates thin protrusions. *Closing* also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

The *opening* of set A by B , denoted by $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B$$

i.e., the opening of A by B is simply the erosion of A by B , followed by a dilation of the result by B .

The *closing* of set A by B , denoted by $A \bullet B$, is defined as

$$A \bullet B = (A \oplus B) \ominus B$$

i.e., the closing of A by B is dilation of A by B , followed by an erosion of the result by B .

Opening and closing have a simple geometric interpretation. Suppose, for example, that we view the disk structuring element B as a flat “rolling ball.” The boundary of $A \circ B$ is then given by the points on the boundary of B that reach the *farthest* into the boundary of A as B is rolled around the *inside* of this boundary.

In a similar way, the boundary of $A \bullet B$ is given by the points on the boundary of B that reach the farthest into the boundary of A as B is rolled on the *outside* of this boundary.

These operations are discussed in great detail in [40] and [55]. Other good references for mathematical morphology include [106], [38] and [73].

3.4 Density Slices after OC operation

The mathematical morphology package [81] was used for the OC filter. Figures 3.4a to p show the corresponding figures in Figures 3.2a to p after the open and close operation was applied using [81] with a “plus” as the structuring element (SE). Figures 3.5a to p show the same for Figure 3.3. Figures 3.6a to p show the density slices in Figures 3.2a to p using the open and close operation with a 3×3 SE while Figures 3.7a to p show the same for Figure 3.3. Figures 3.8a to p show the density slices in Figures 3.2a to p using the open and close operation with a 5×5 SE while Figures 3.9 show the same for Figure 3.3. These figures are shown in the next 6 pages. The text continues on page 78.

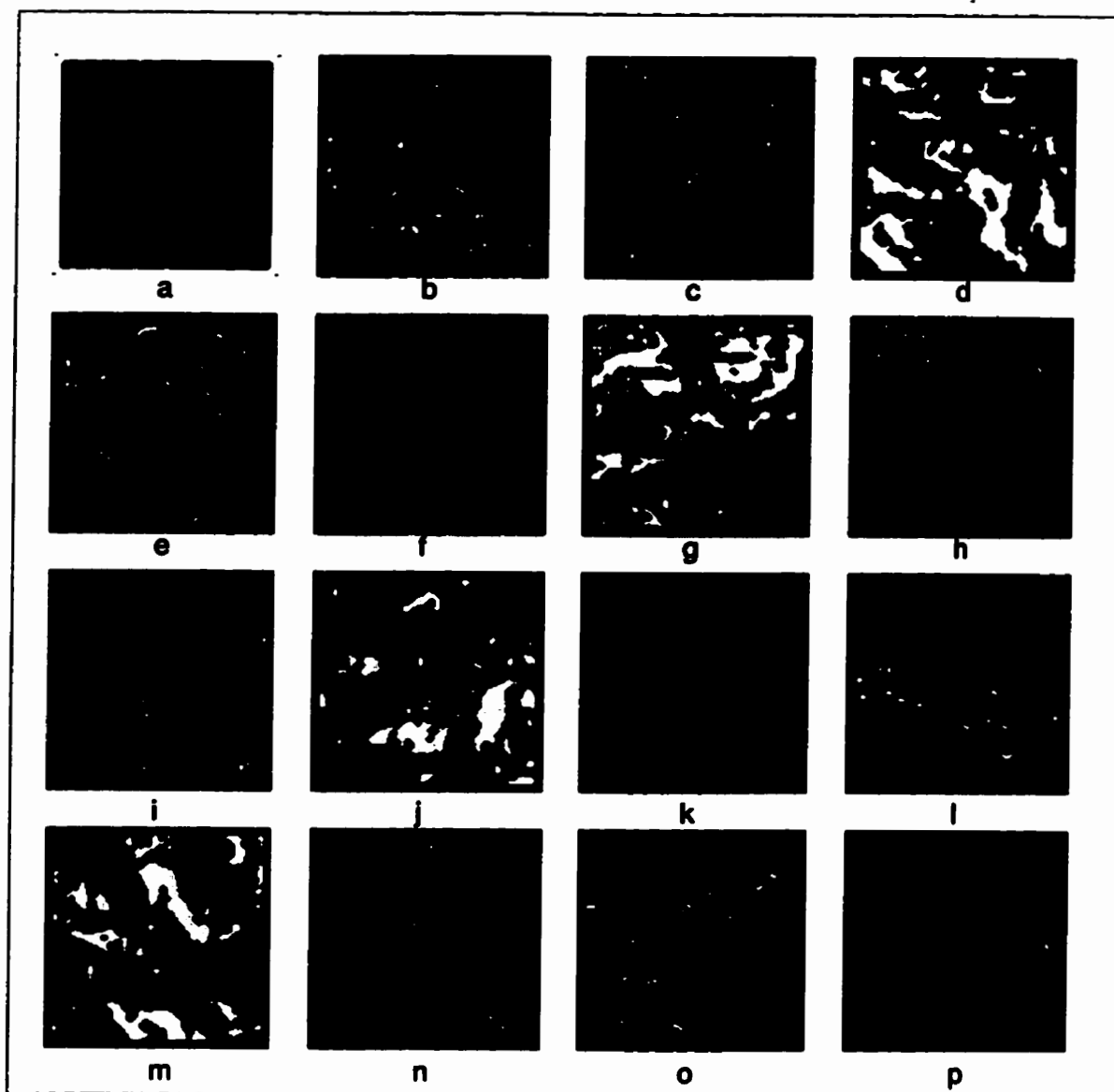


Figure 3.4: *a* to *p*: Density slices in Figure 3.2 after OC operation with a “plus” SE.

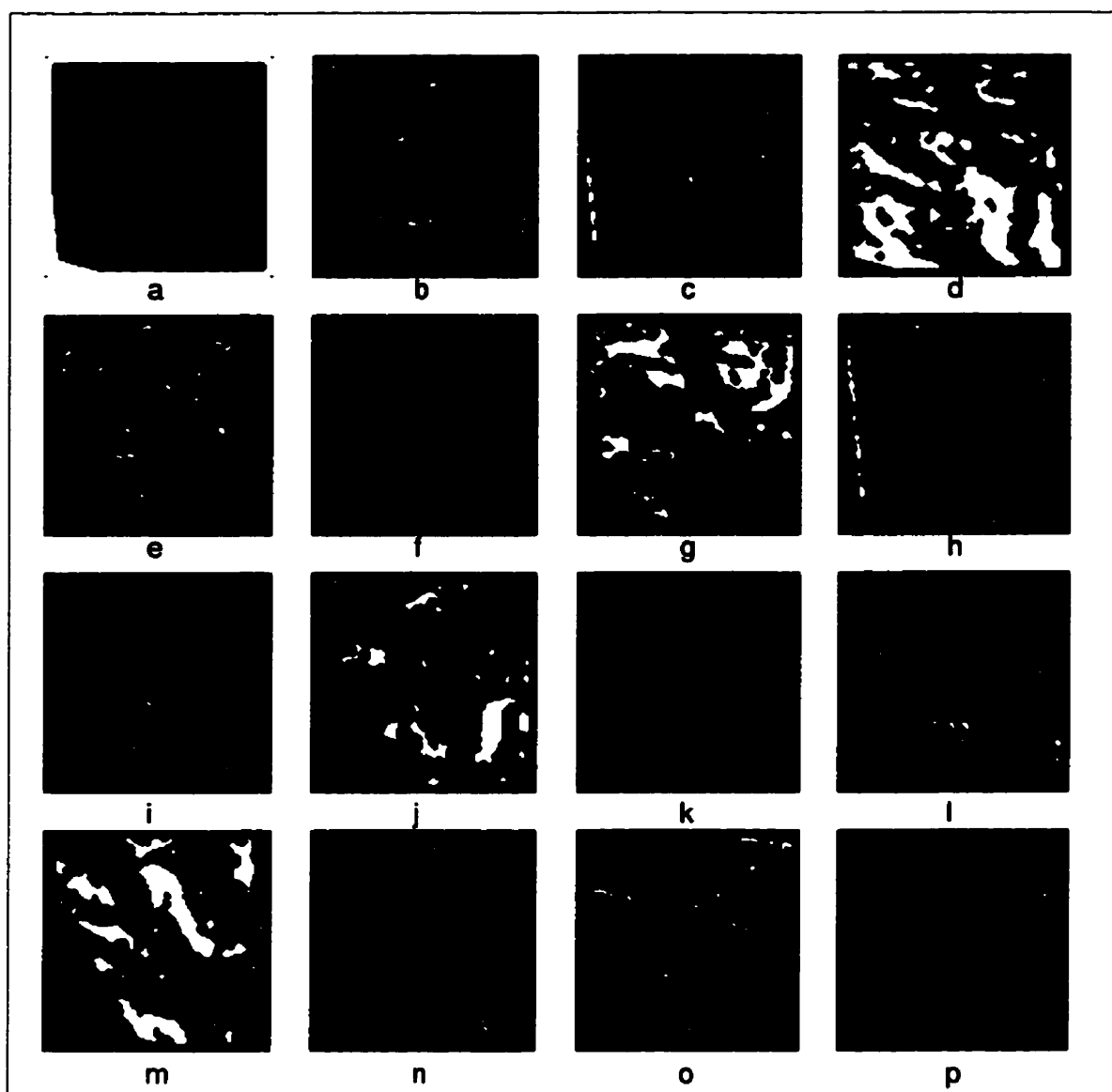


Figure 3.5: *a* to *p*: Density slices in Figure 3.3 after OC operation with a “plus” SE.

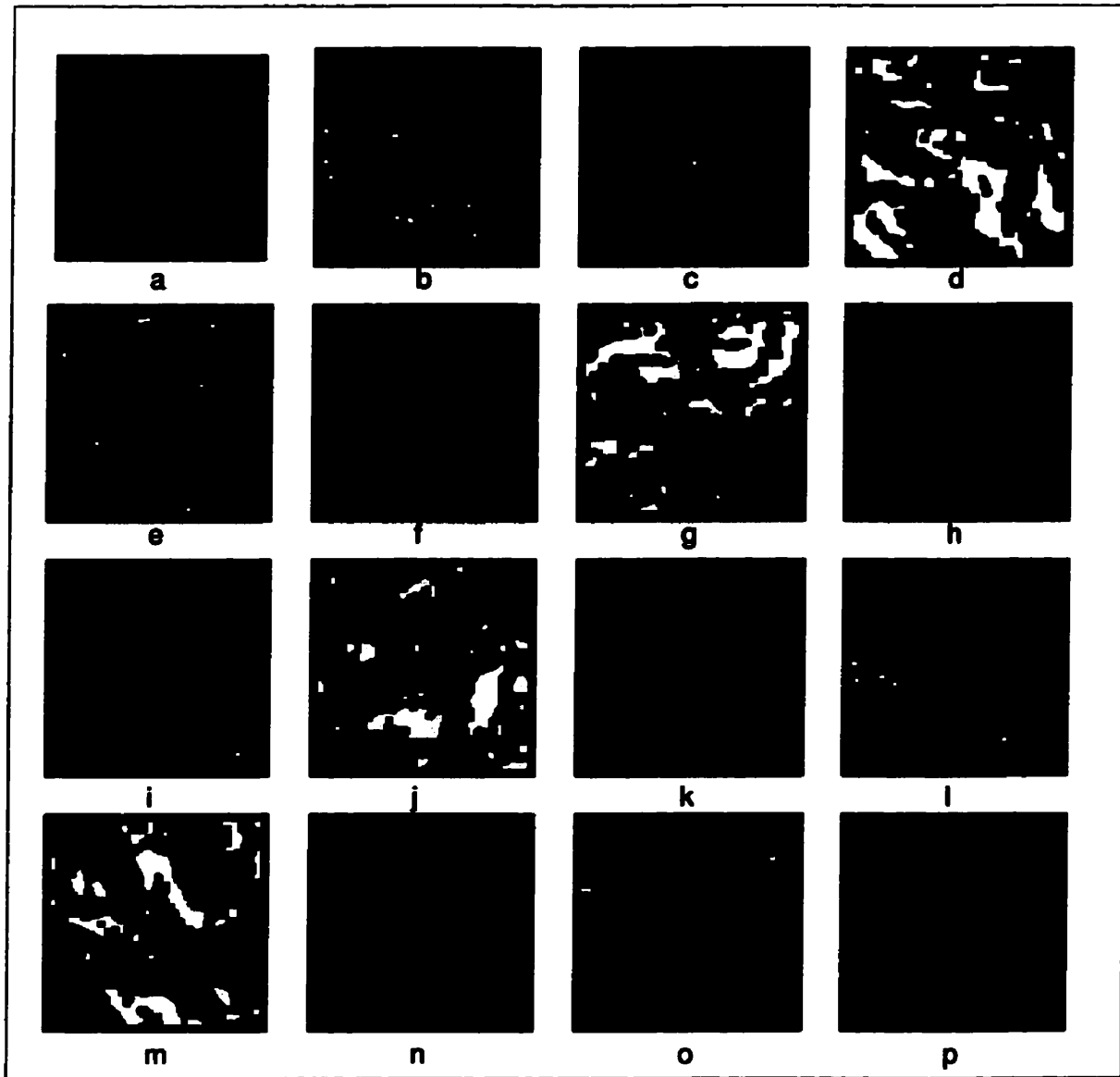


Figure 3.6: *a* to *p*: Density slices in Figure 3.2 after OC operation with a 3×3 SE.

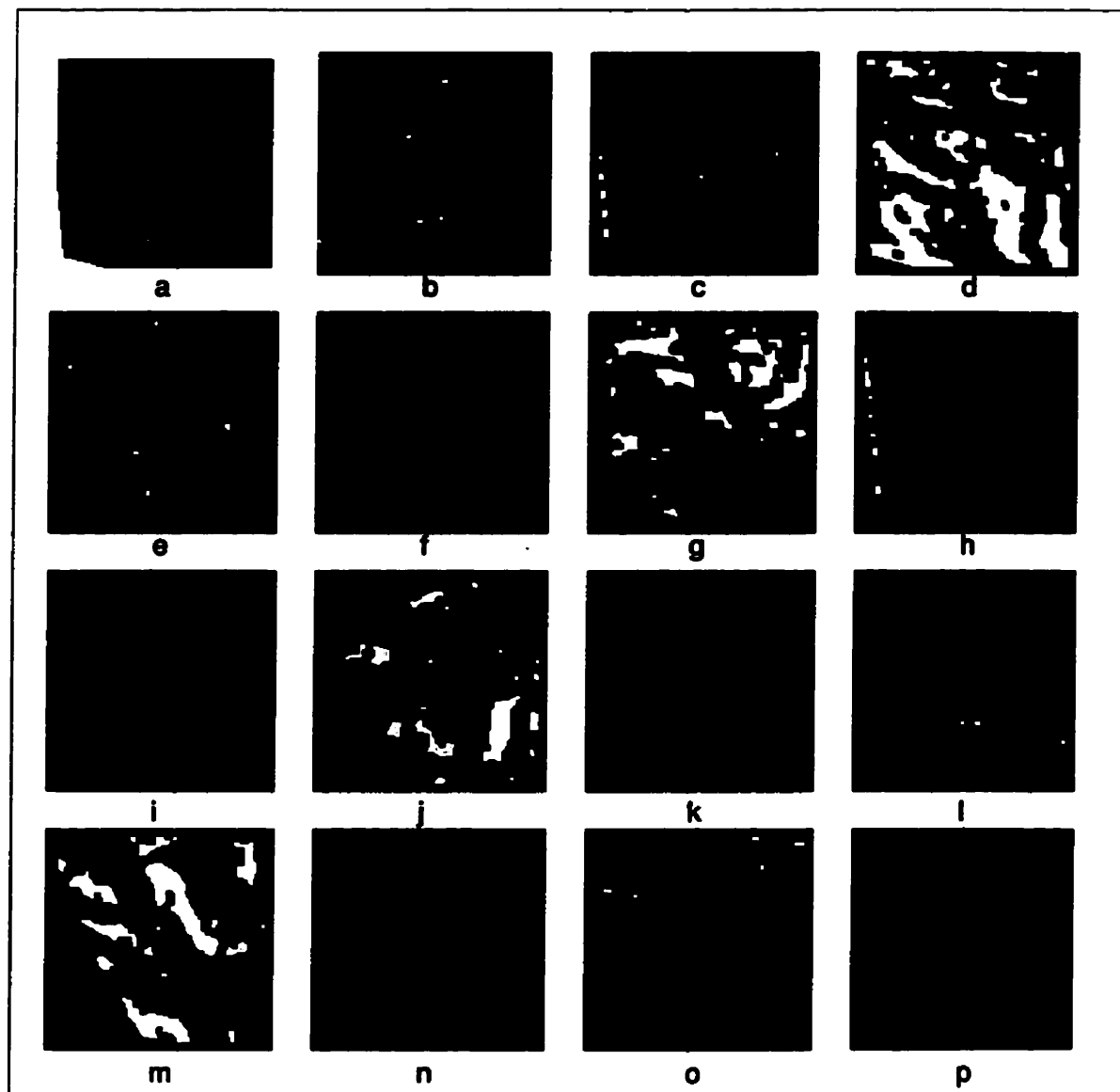


Figure 3.7: *a* to *p*: Density slices in Figure 3.3 after OC operation with a 3×3 SE.

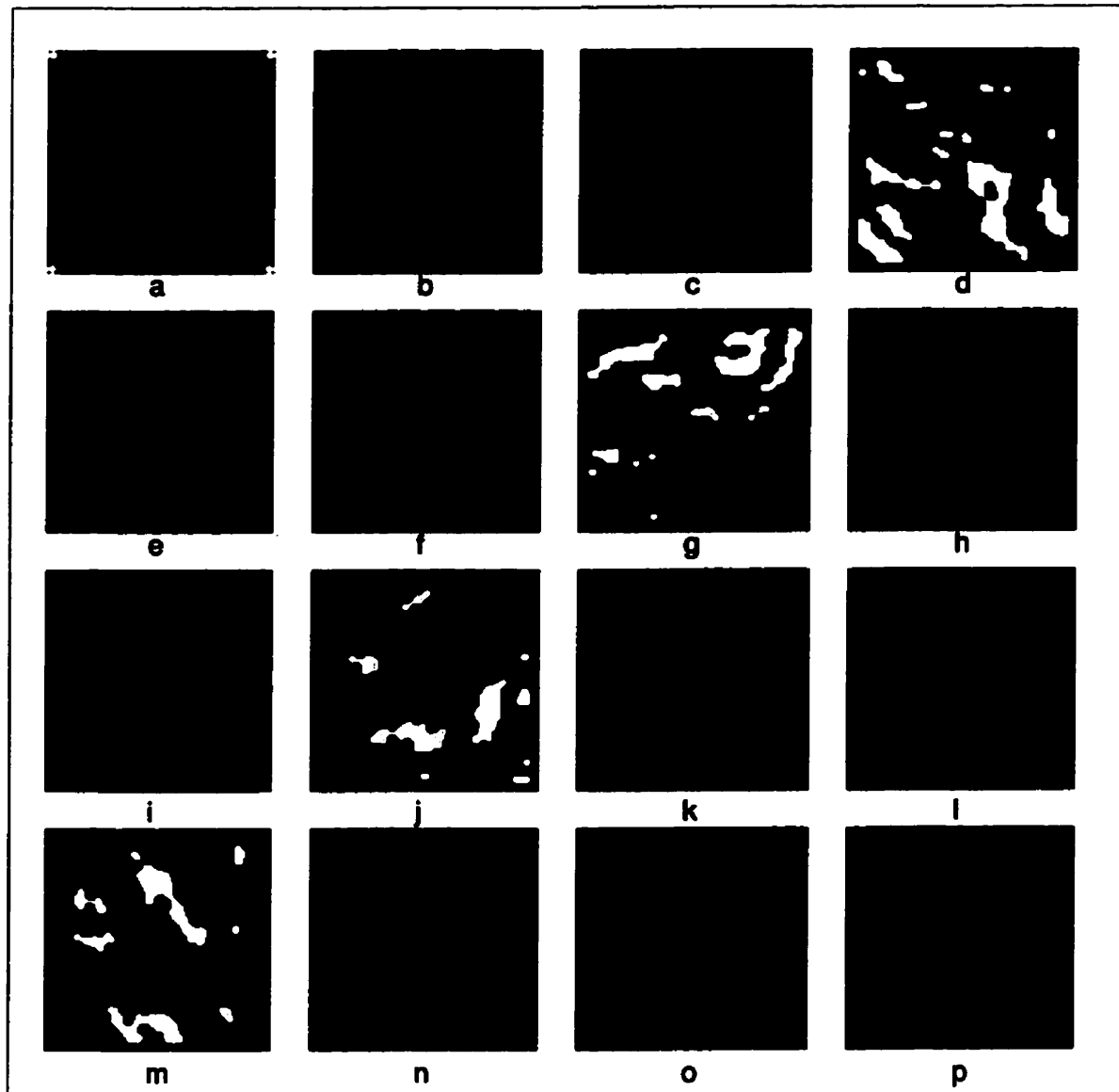


Figure 3.8: *a* to *p*: Density slices in Figure 3.2 after OC operation with a 5×5 SE.

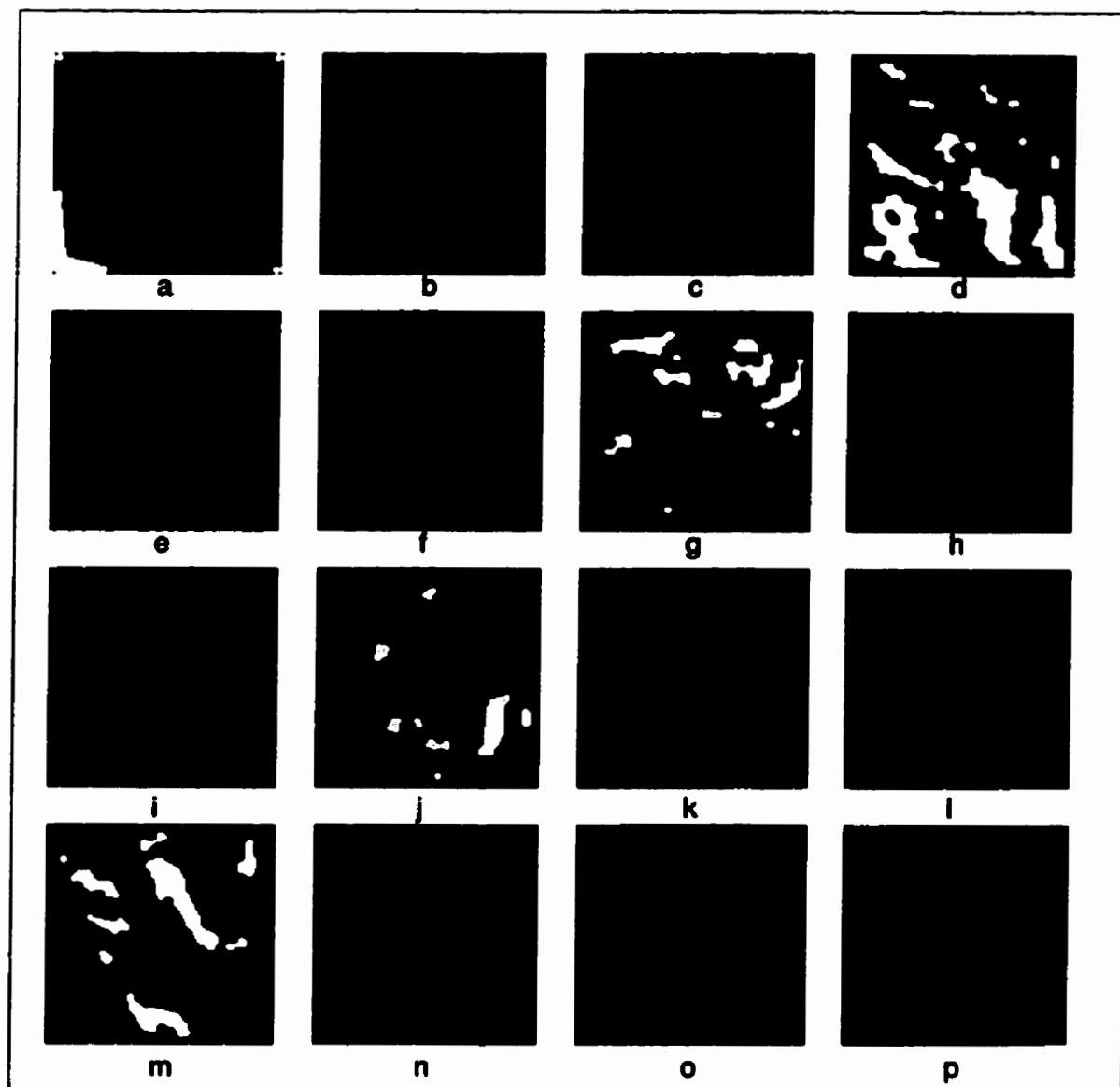


Figure 3.9: *a* to *p*: Density slices in Figure 3.3 after OC operation with a 5×5 SE.

As can be seen, the images in all cases look much cleaner, with most of the isolated and small groups of pixels removed. Also the boundaries of the large regions look smoother. We have used the “plus” SE for all our examples. In the rest of this section we have presented the density slices for many more examples. All these images were discussed in the previous chapter. The distortions used were also discussed in the previous chapter.

Figures 3.10a and b show the starbyte-transformed images of Figure 2.2b and Figure 2.3b obtained using a 9×9 neighborhood size with protocol 1.

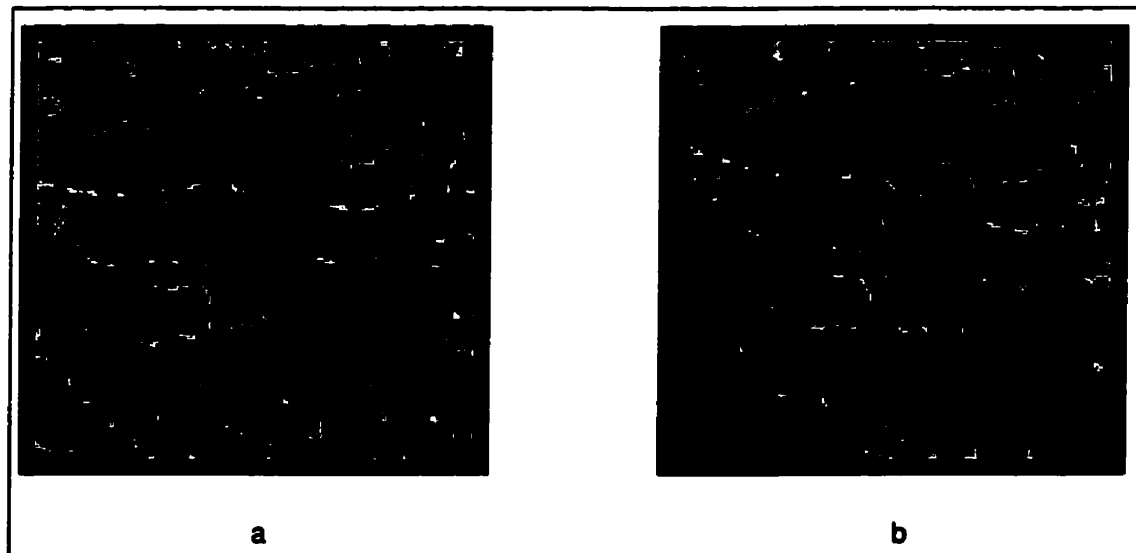


Figure 3.10: *a* and *b*: Starbyte images of Figure 2.2b and Figure 2.3b obtained by applying protocol 1 with a 9×9 neighborhood size.

Figures 3.11a to p show the density slices of Figure 3.10a (after an OC operation with a “plus” SE) while Figures 3.12a to p show the same for Figure 3.10b.

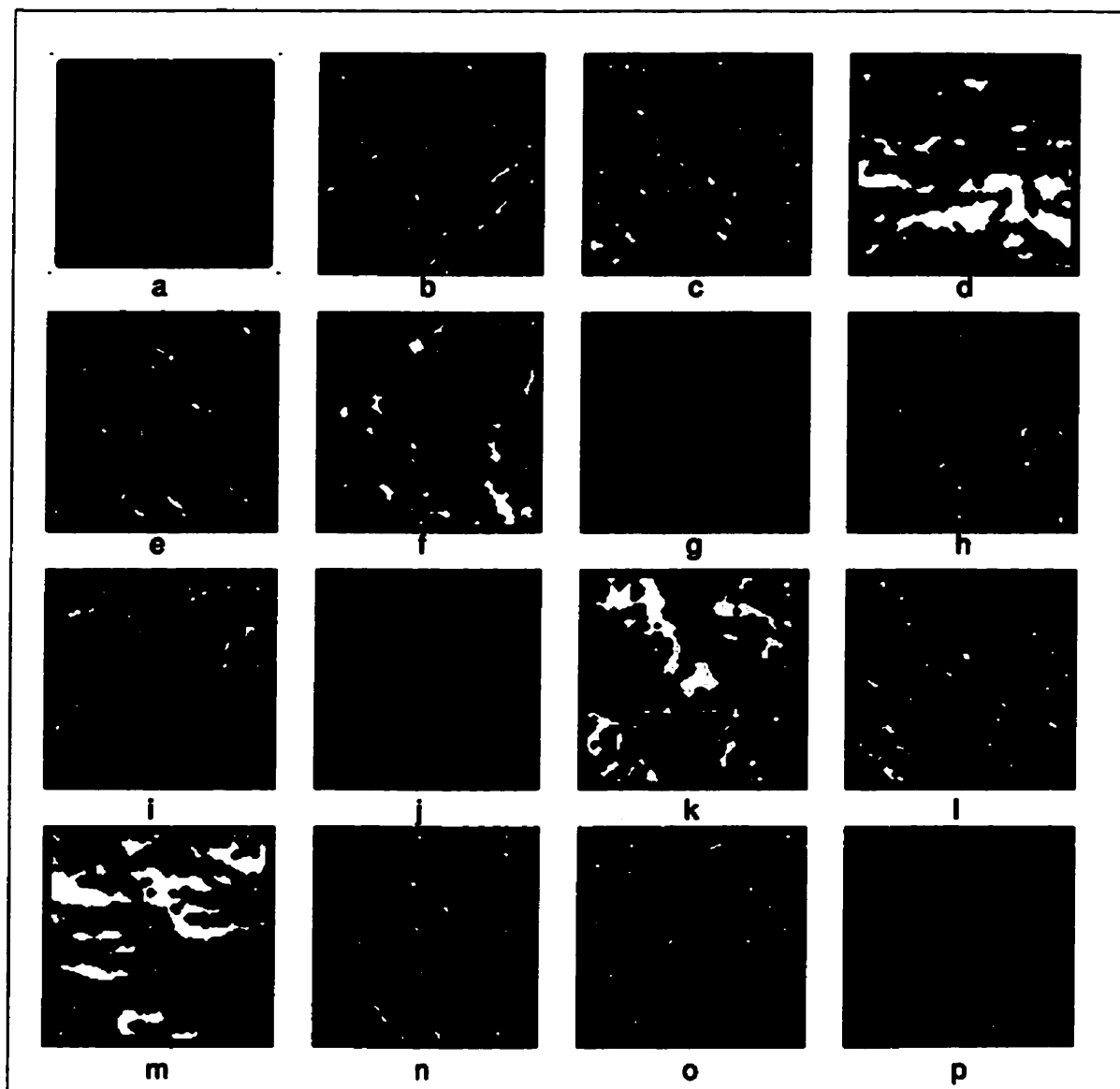


Figure 3.11: *a* to *p*: Density slices in Figure 3.10a after OC operation with a “plus” SE.

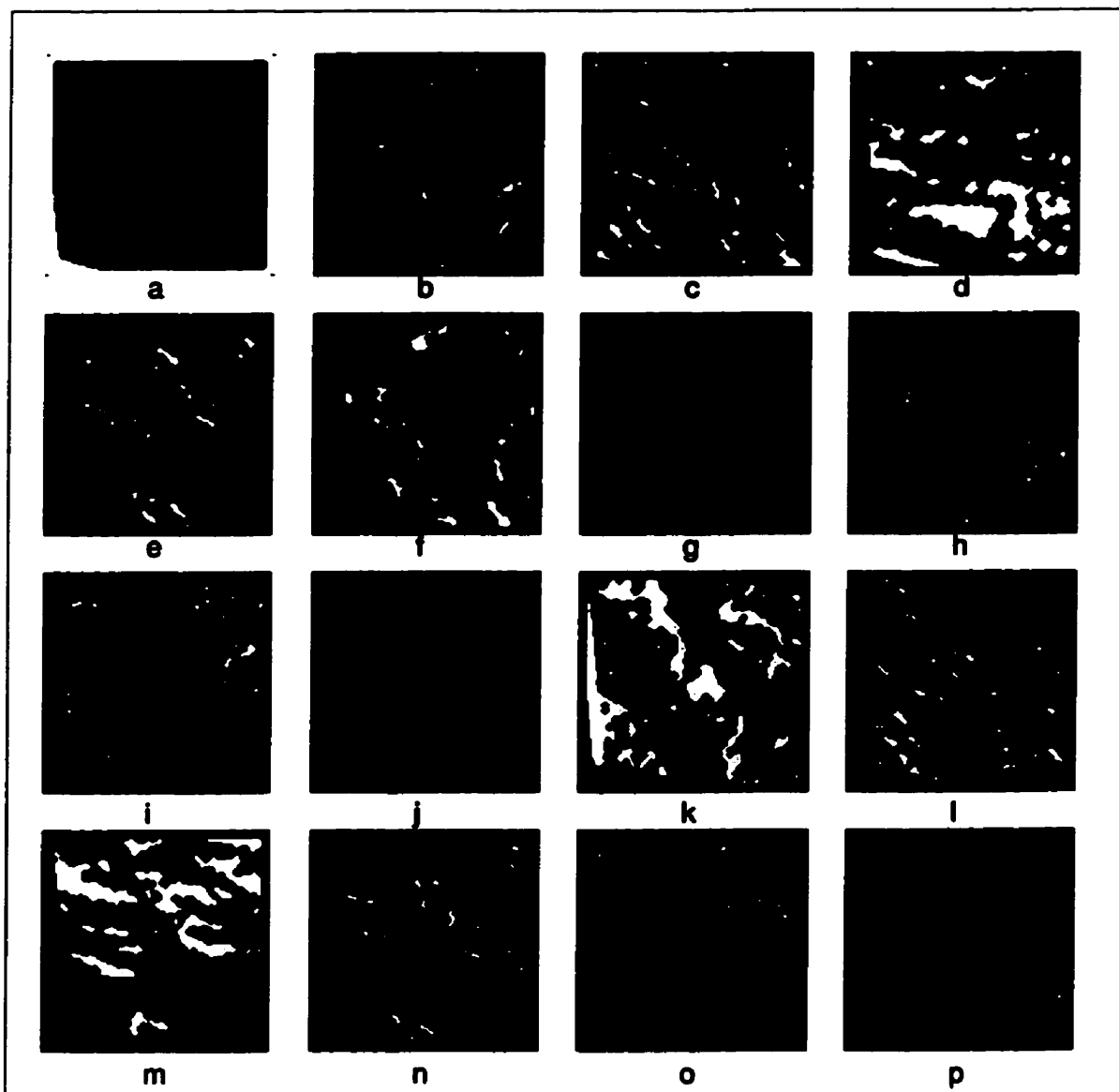


Figure 3.12: *a* to *p*: Density slices in Figure 3.10b after OC operation with a “plus” SE.

In the previous chapter, Figure 2.15a to d showed Figure 2.2b and its translated (8 pixels upwards and 10 pixels leftwards), rotated (10 deg) and scaled (10 %) versions respectively.

Figures 3.13a to d show the starbyte images for the same using the 3rd protocol on a 9 x 9 neighborhood size. Figure 3.13a is the same as Figure 3.1a. Figures 3.14a to p shows the density slices of Figure 3.13b after an open-and-close operation with a "plus" SE) while Figures 3.15a to p and Figures 3.16a to p show the same for Figures 3.13c and d.

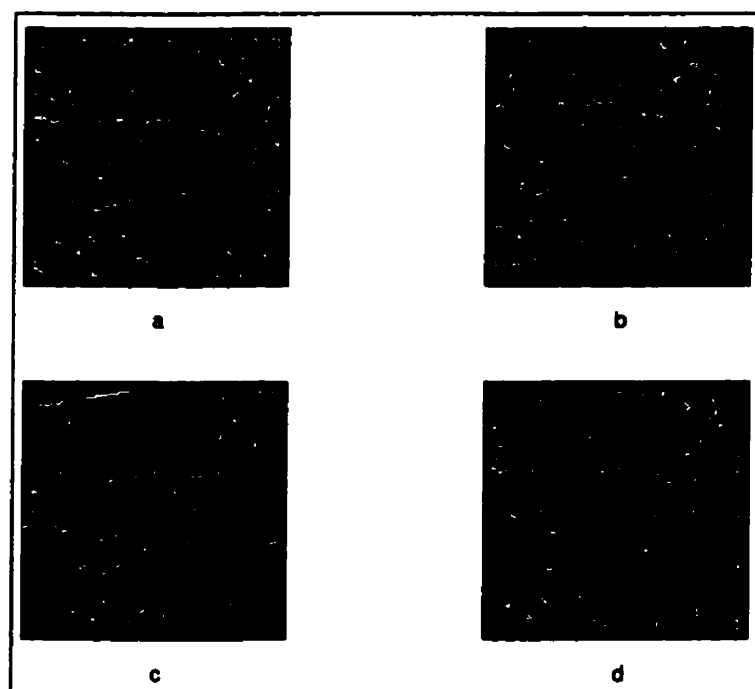


Figure 3.13: *a – d*: Starbyte images of Figures 2.15a to d of the previous chapter using the 3rd protocol on 9 x 9 neighborhoods. Figures 2.15b to d were obtained by translating (8 pixels upwards and 10 pixels leftwards), rotating (10 deg) and scaling (10 %) Figure 2.15a.

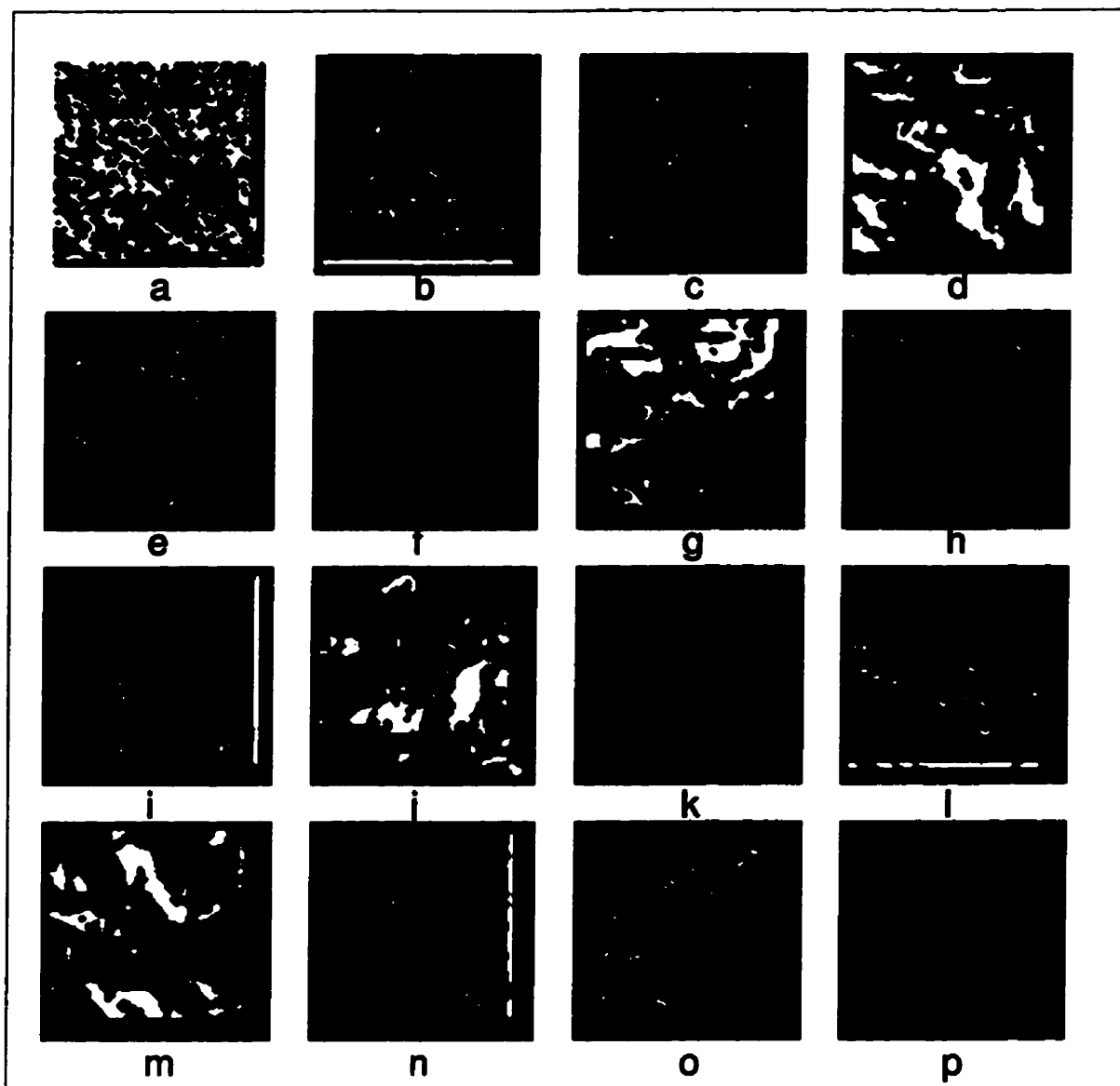


Figure 3.14: *a* to *p*: Density slices of Figure 3.13b after OC operation with a “plus” SE.

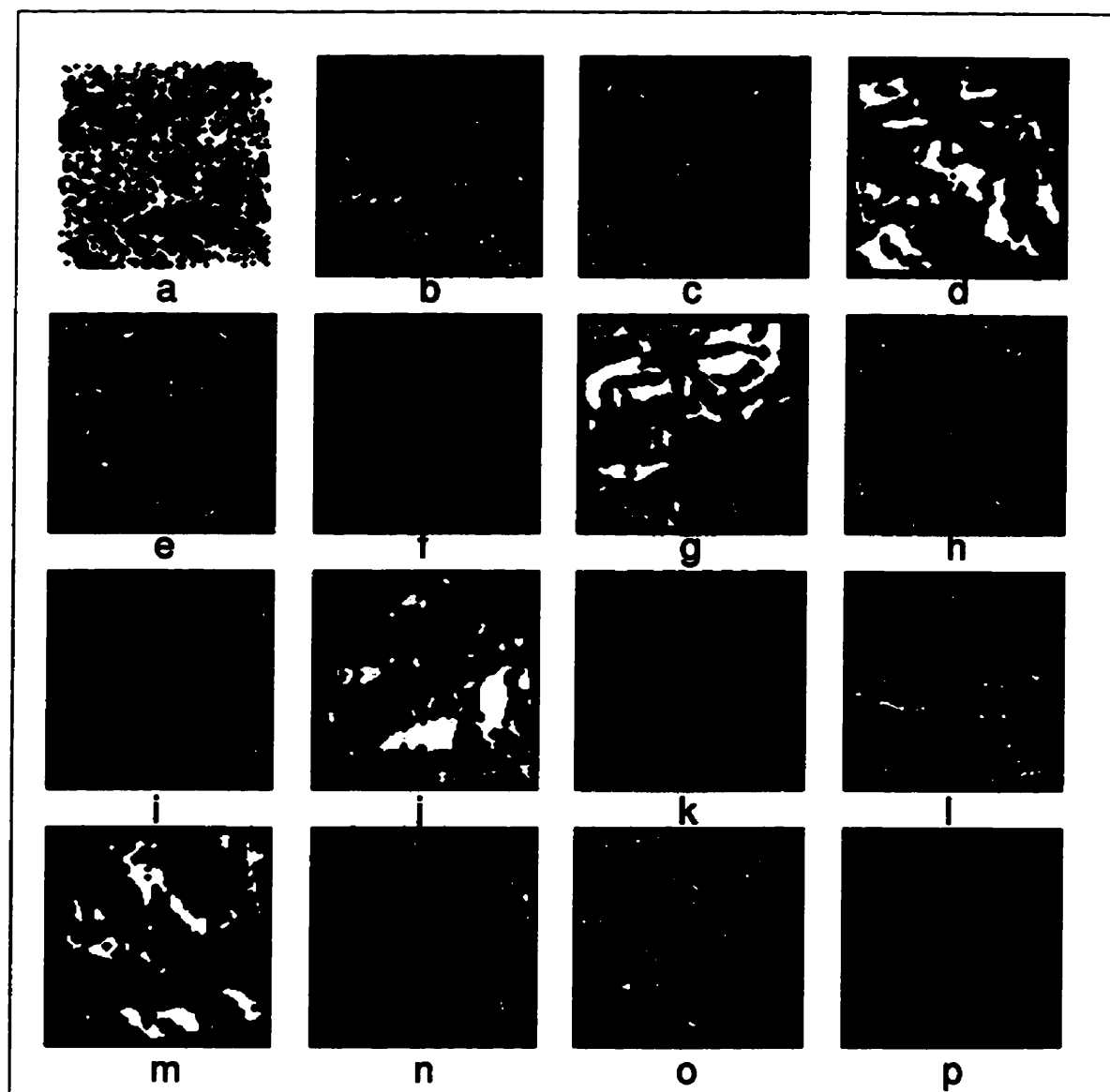


Figure 3.15: *a* to *p*: Density slices of Figure 3.13c after OC operation with a “plus” SE.

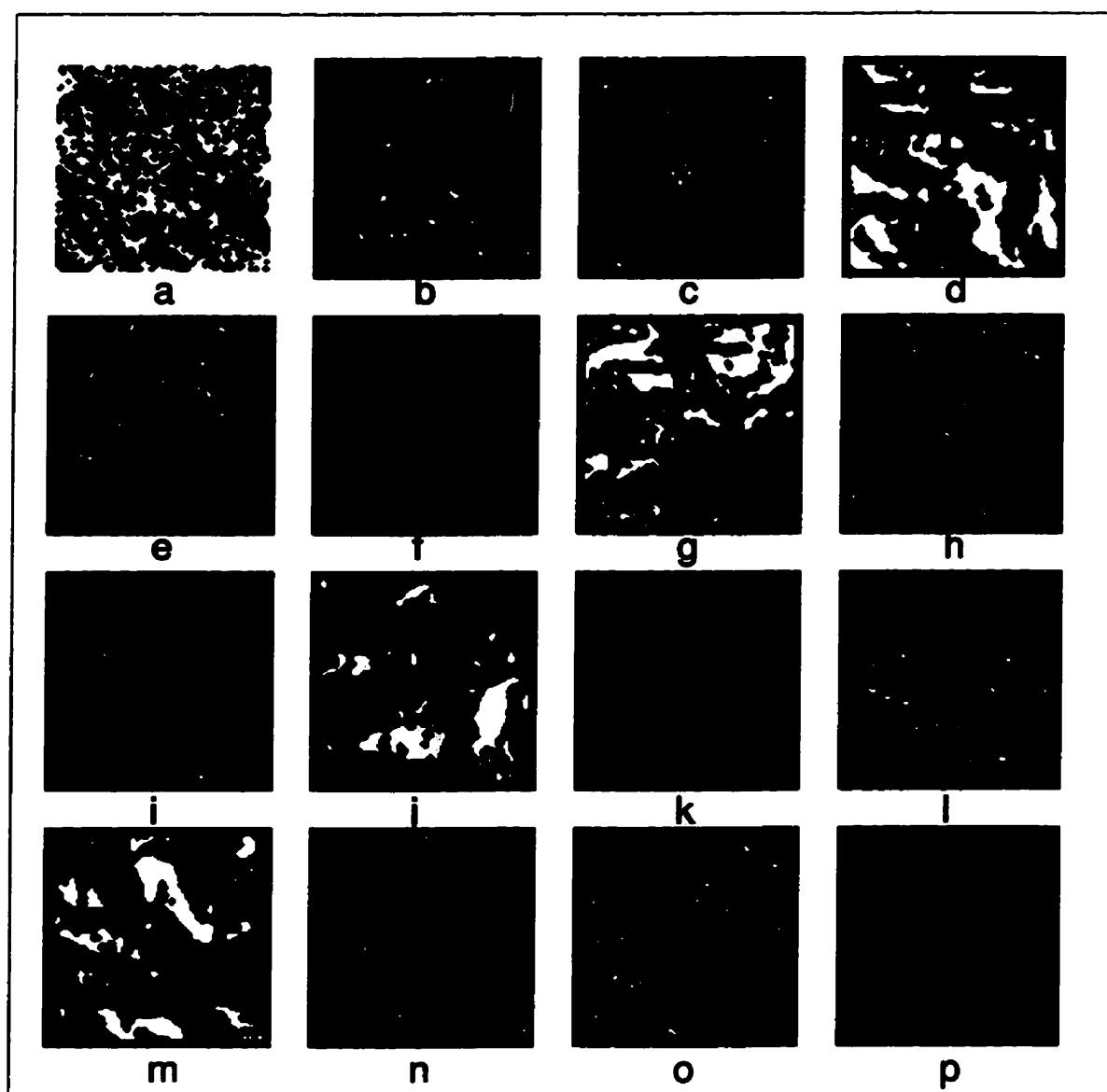


Figure 3.16: *a* to *p*: Density slices of Figure 3.13d after OC operation with a “plus” SE.

Figure 3.17a shows the starbyte image for Figure 2.24a using the 1st protocol on 9×9 neighborhoods, while Figure 3.17b was the result of applying the starbyte transformation (1st protocol on 9×9 neighborhoods) after applying the TPS distortion described in chapter 3 to Figure 2.24a. Figures 3.18a to p shows the density slices of Figure 3.17a after an open-and-close operation with a “plus” SE) while Figures 3.19a to p shows the same for Figure 3.17b.

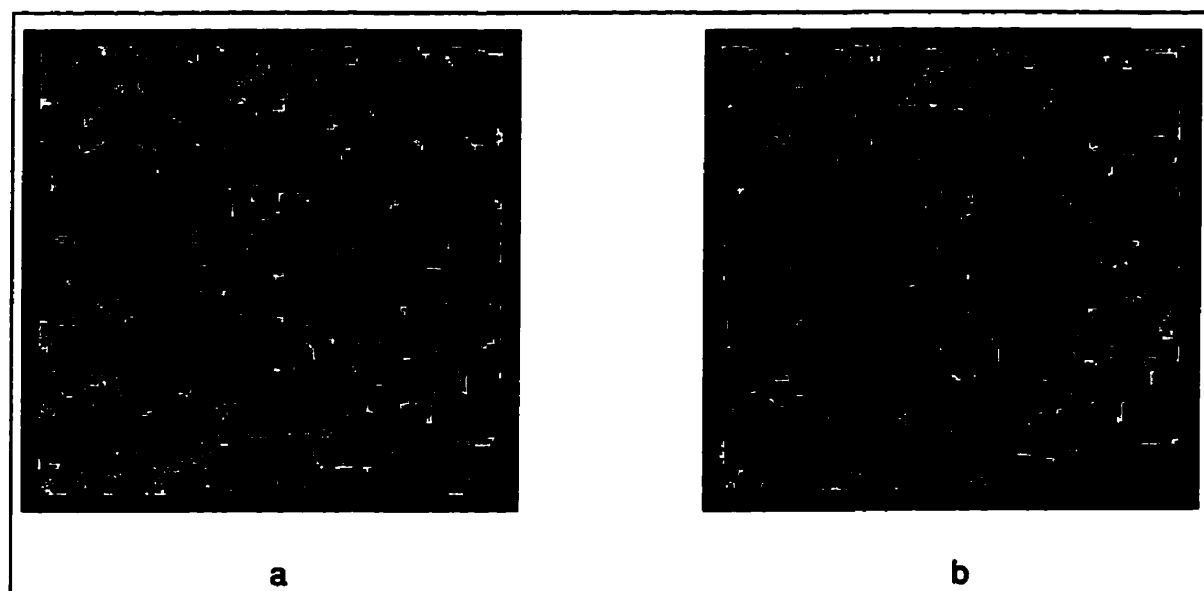


Figure 3.17: *a* and *b*: Starbyte images of Figure 2.24a and its TPS-distorted version obtained by applying protocol 1 with a 9×9 neighborhood size.

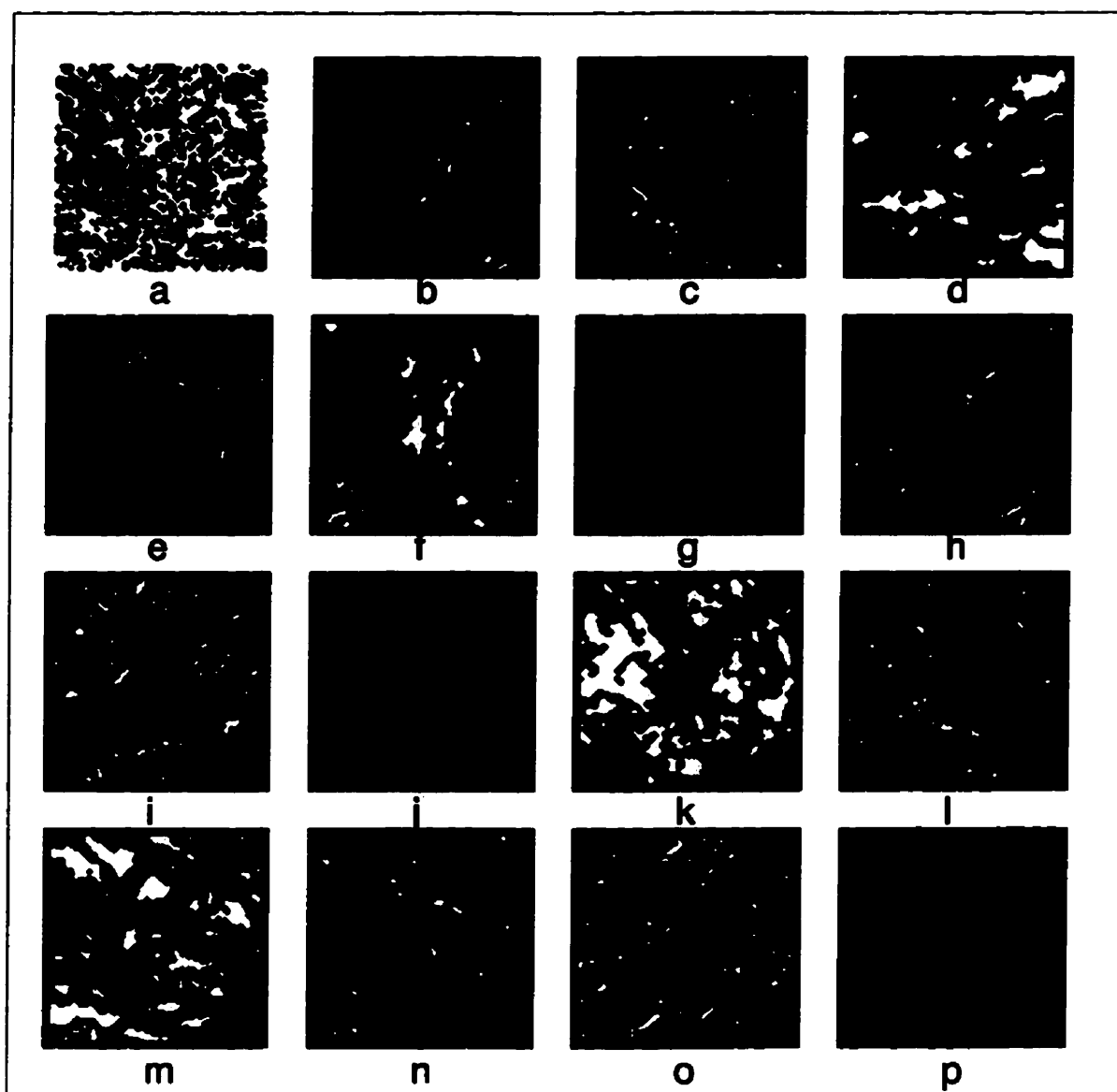


Figure 3.18: *a* to *p*: Density slices of Figure 3.17a after OC operation with a “plus” SE.

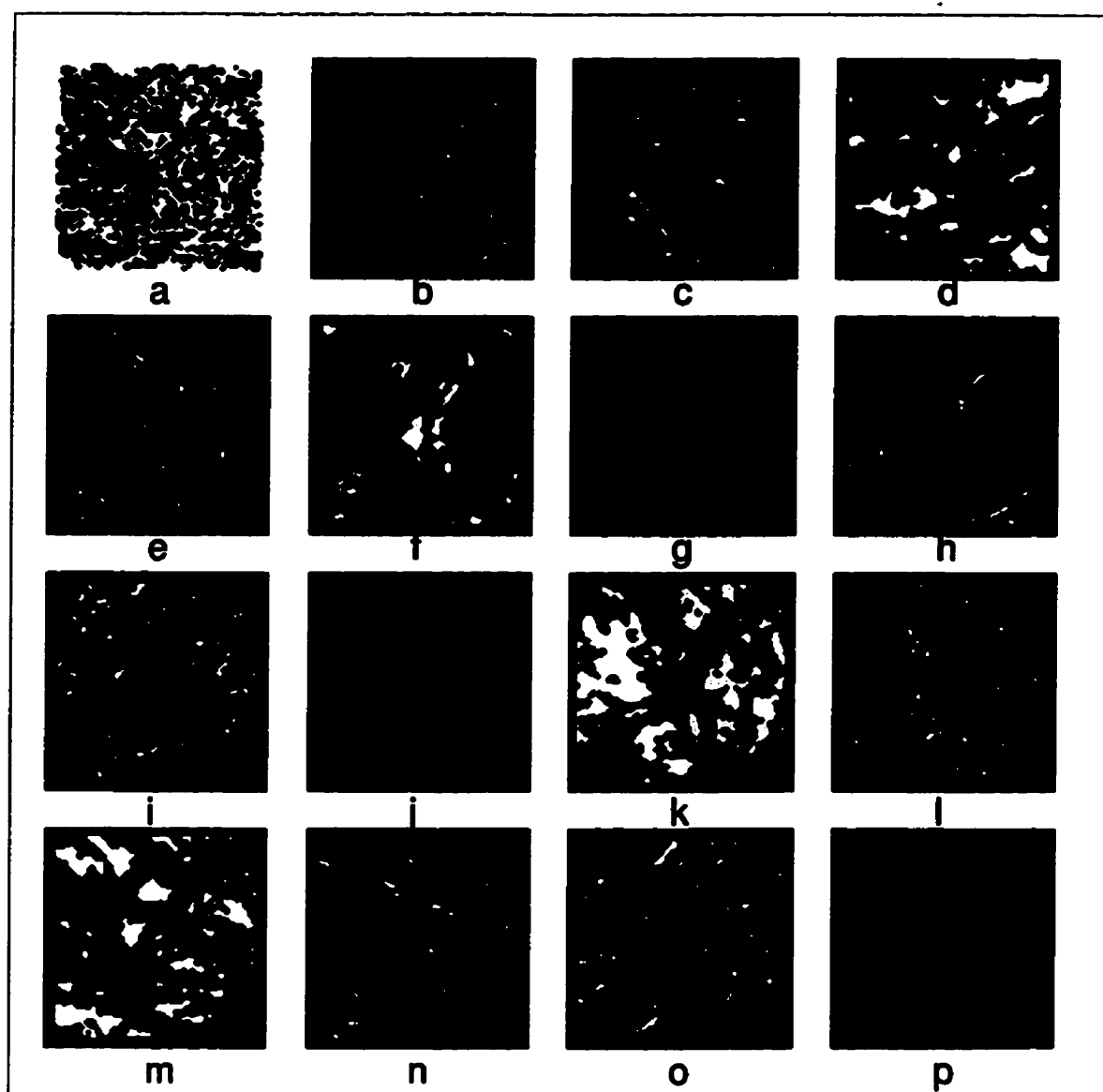


Figure 3.19: *a* to *p*: Density slices of Figure 3.17b after OC operation with a “plus” SE.

Figures 3.20a and b show the starbyte images (1st protocol, 9×9 neighborhoods) for Figure 2.2a and its sinusoidally-distorted counterpart (using the sinusoid distortion given by equation (2.2)). Figures 3.21a to p shows the density slices of Figure 3.20a after an open-and-close operation with a “plus” SE) while Figure 3.22a to p show the same for Figure 3.20b.

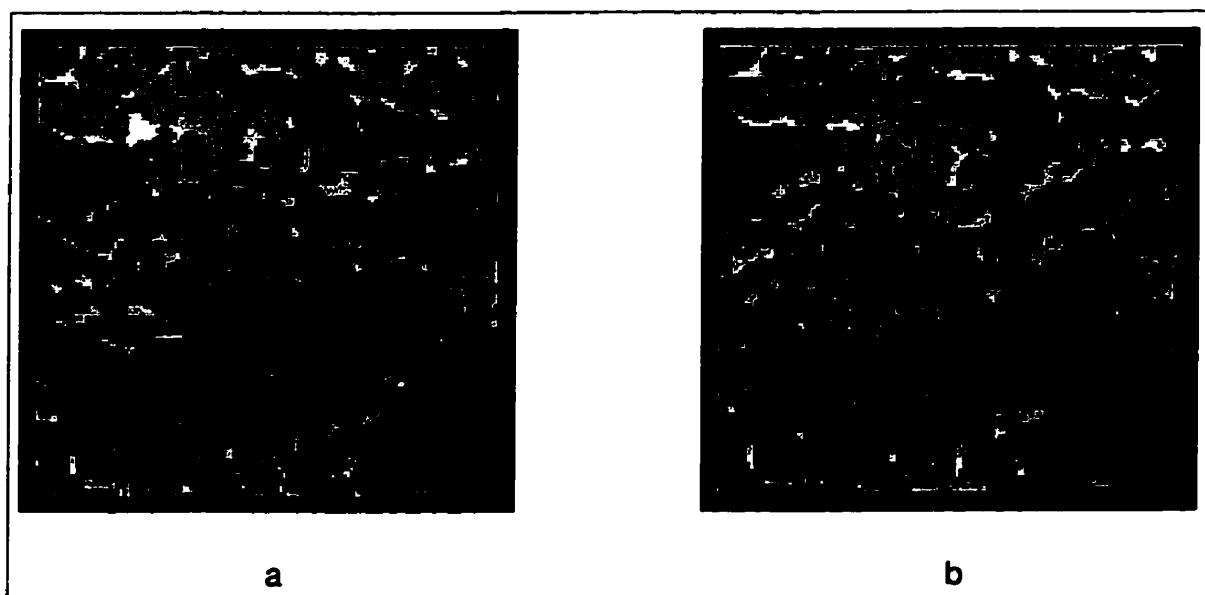


Figure 3.20: *a* and *b*: Starbyte images of Figure 2.2a and its sinusoidally-distorted version (using equation (2.2) obtained by applying protocol 1 with a 9×9 neighborhood size.

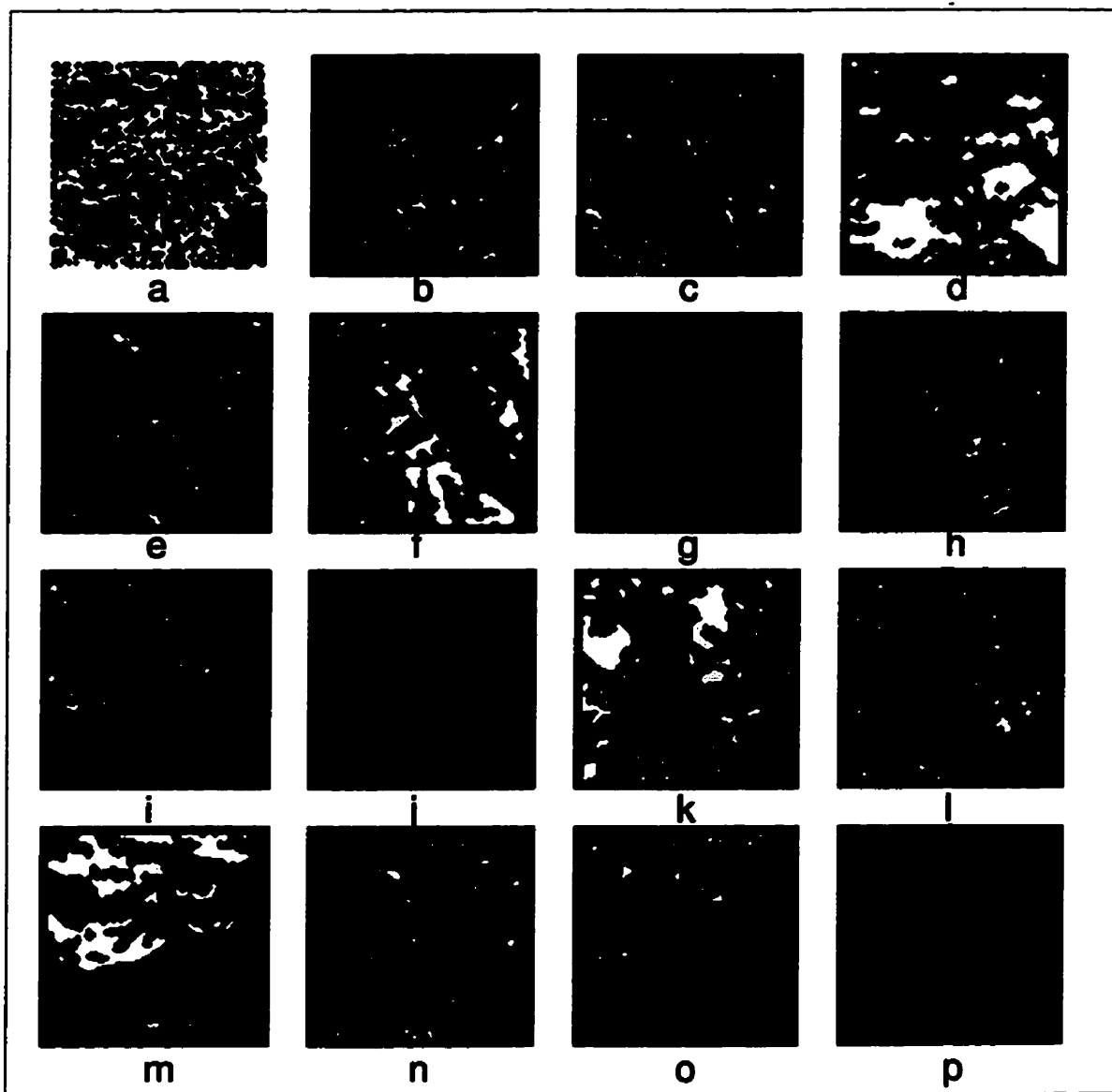


Figure 3.21: *a* to *p*: Density slices of Figure 3.20a after OC operation with a “plus” SE.

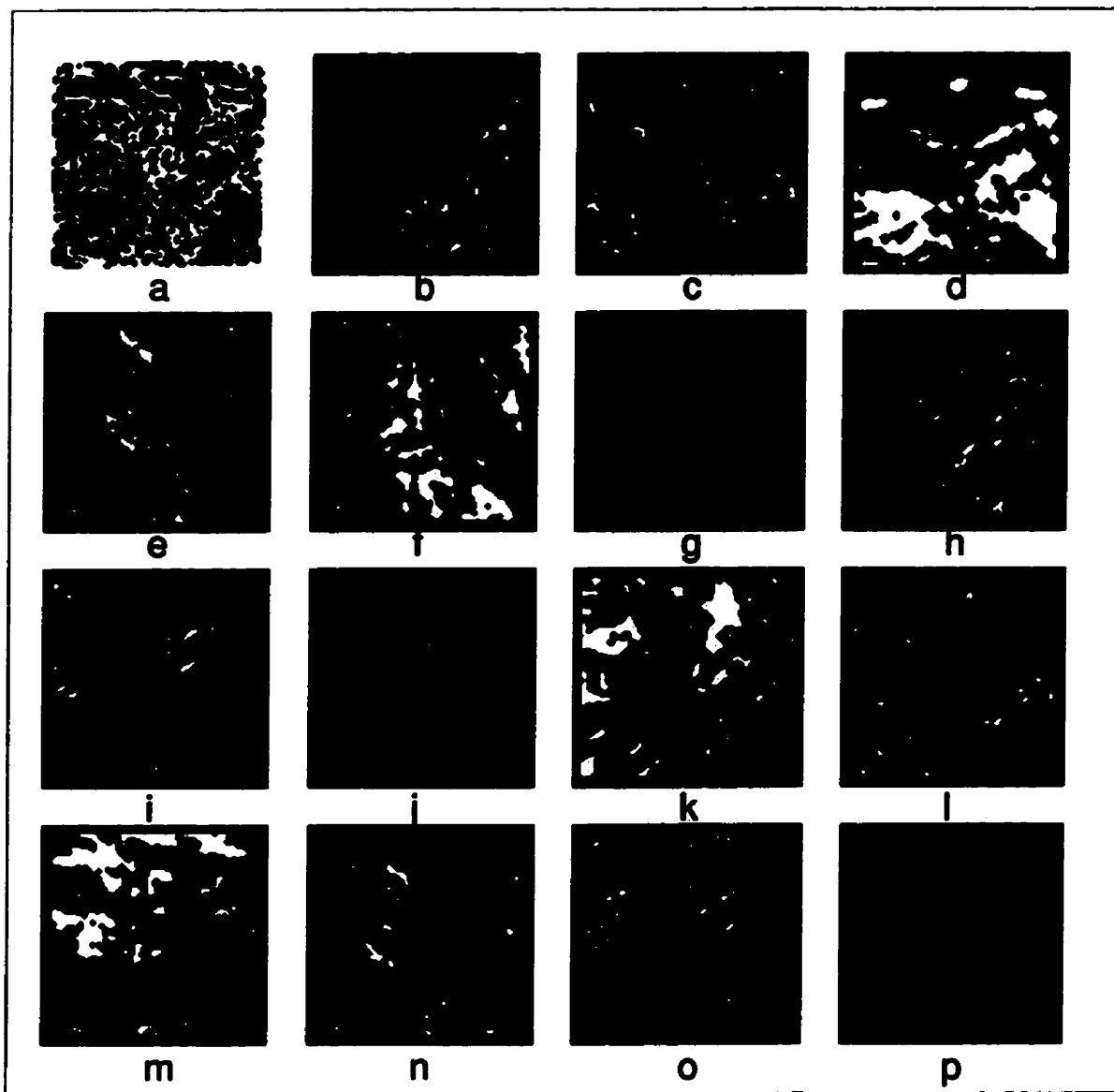


Figure 3.22: *a* to *p*: Density slices of Figure 3.20b after OC operation with a "plus" SE.

From an observation of corresponding slices for a given pair of starbyte images, it is clear that matching regions can be easily identified even by eye. These matching regions can then be used to extract control points for image registration. We shall show numerically in this chapter that the centroids of these regions form good control points. However, before we do that, another important issue needs to be addressed.

If one observes the corresponding slices for the image pairs, it can be seen that regions present in one slice are fragmented in the other, or conversely, separate regions in one slice are concatenated in the other. We recall from the previous chapter that each pixel in the starbyte image is a bit pattern generated by binary comparisons between the average grey level over each sector of its local neighborhood with either the average grey level over the entire neighborhood or the grey level of the pixel under consideration. Now, if even one bit in the pattern is changed, it results in a new density value, and hence when a density slice is taken at a particular level, parts of the region having a different density value will not be included in this slice, and hence the region will appear broken at these parts or will appear fragmented.

The effect of this fragmentation can be studied using the concept of *Hamming distance* and *Hamming neighbors*. This will be discussed in the following section.

3.5 Hamming distance and Hamming neighbors

Hamming distance or *Hamming metric* $d(u, v)$, between two binary words u and v (of the same length) is the number of corresponding bit positions in which the two words have different bit values and is also called the *signal distance* [26], [74]. The Hamming distance is important in the theory of error-correcting codes and error-detecting codes [53], [61].

Using the above definition of Hamming distance, given a bit sequence, we can

identify Hamming neighbors of this sequence at different distances. For e.g., for a 4-bit bit-pattern which is equal to the decimal value "1", the Hamming neighbors at a distance 1, are 0, 3, 5 and 9, while Hamming neighbors at a distance 2 are 2, 4, 7, 8, B and D. Here "B" and "D" are hexadecimal notations. Every number has 4 Hamming neighbors at a distance of 1, 6 neighbors at a distance of 2, 4 neighbors at a distance of 3, and 1 neighbor at a distance of 4. In general for a n-bit number, the number of Hamming neighbors at a distance "d" is given by n_{c_d} , where the "c" denotes combination.

For a 4-bit protocol, there are 16 distinct density levels, which when represented as hexadecimal numbers are 0 to 9, and "A" to "F". Figure 3.23a to d show the Hamming neighbors at distances 1, 2, 3 and 4 respectively for these 16 levels.

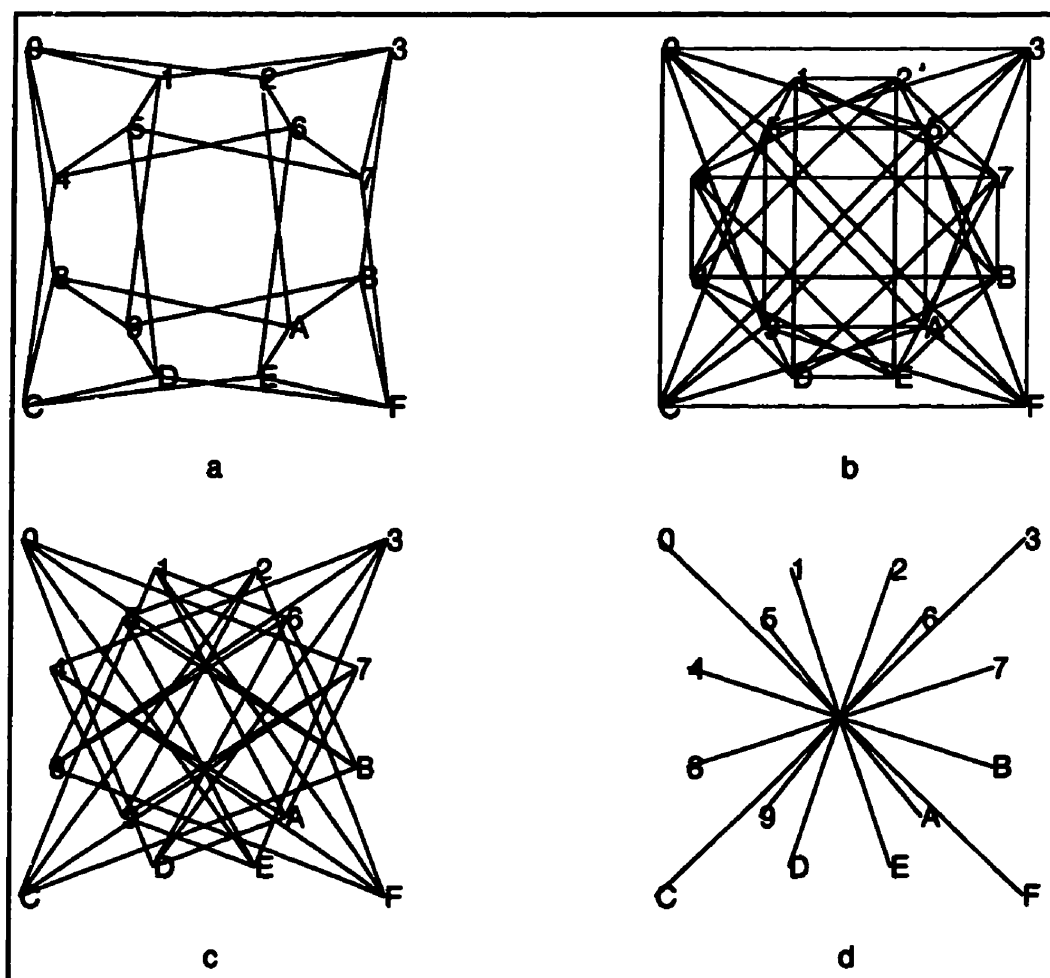


Figure 3.23: *a* to *d*: Hamming neighbors at distances 1, 2, 3 and 4 for the 16 density levels represented as hexadecimal numbers (0 to 9, "A" to "F").

In order to study the fragmentation effect described above, at each density level, instead of considering that density level alone, we will also combine the Hamming neighbors at different distances. In each case, the open-and-close filter will be applied to the slices to clean up the image.

Figures 3.24a to p show the density slices at the 16 levels combined with their Hamming neighbors at distance 1 for Figure 3.1a, while Figures 3.25a to p show the same for Figure 3.1b. Figures 3.26a to p and Figures 3.27 are the same as Figures 3.24a to p and Figures 3.25a to p, after an OC operation with a “plus” SE. These figures are shown in the next 4 pages. The text continues on page 99.

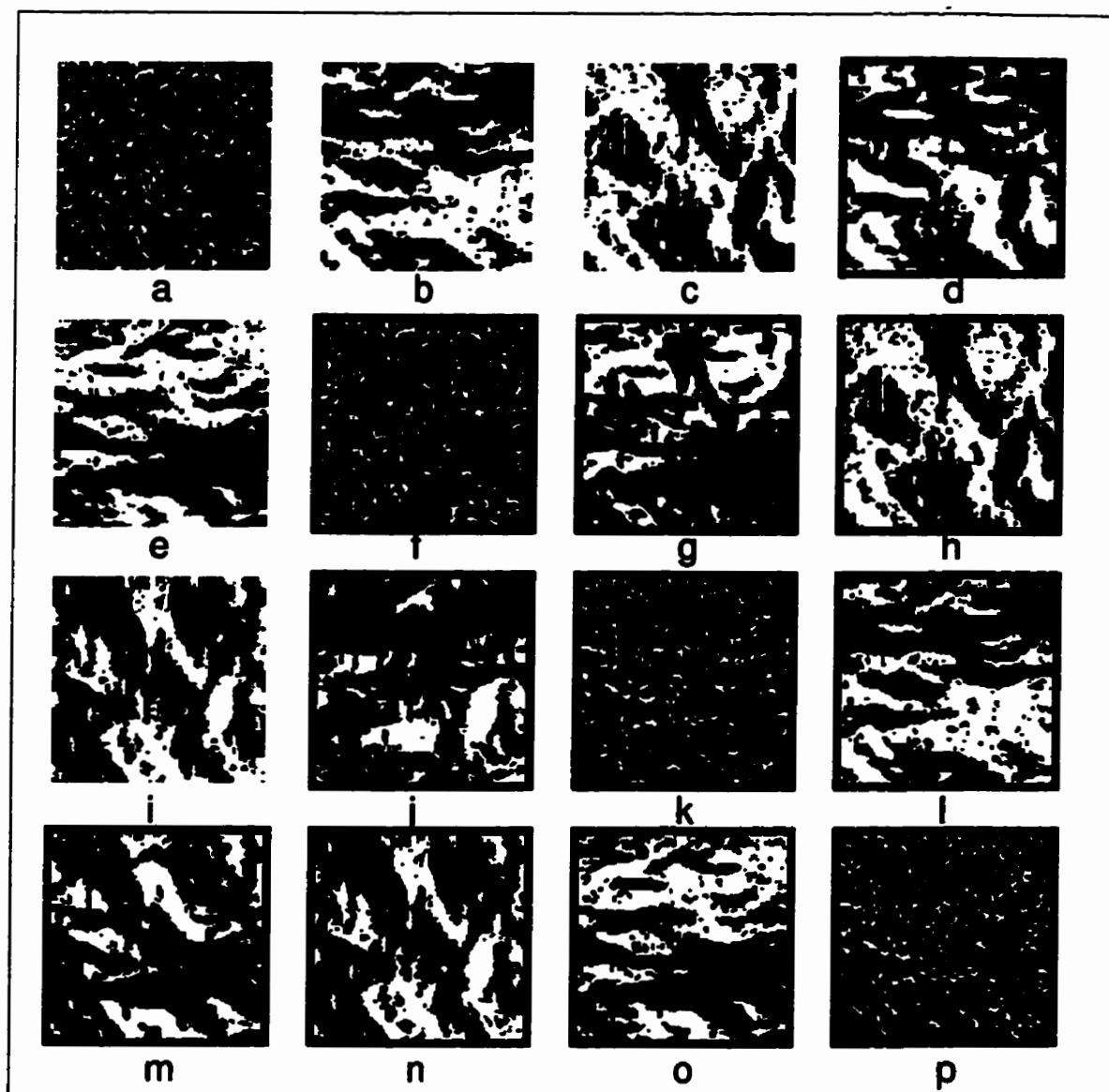


Figure 3.24: *a* to *p*: Density slices of Figure 3.1a combined with their Hamming neighbors at distance 1.

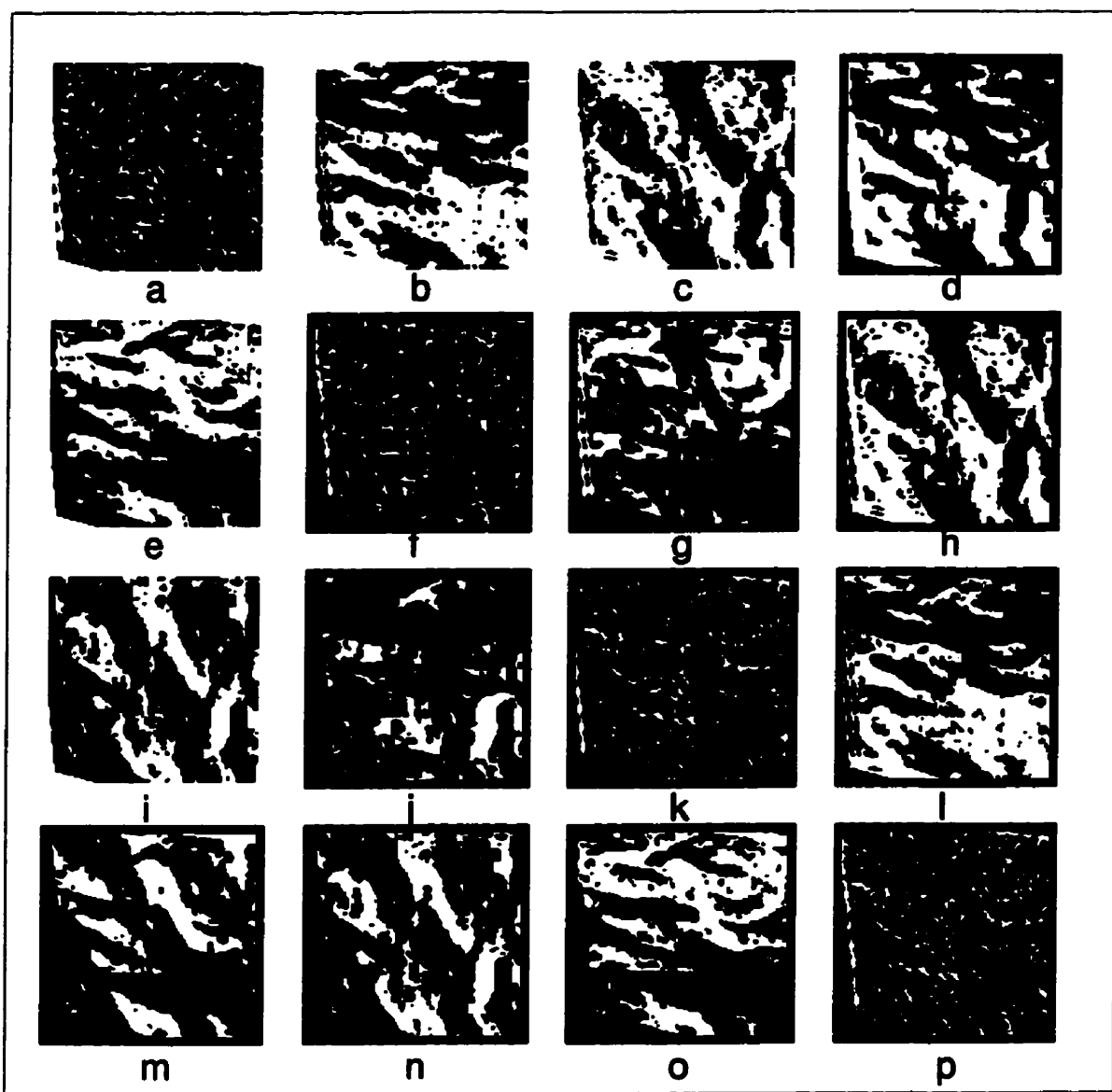


Figure 3.25: *a* to *p*: Density slices of Figure 3.1b combined with their Hamming neighbors at distance 1.

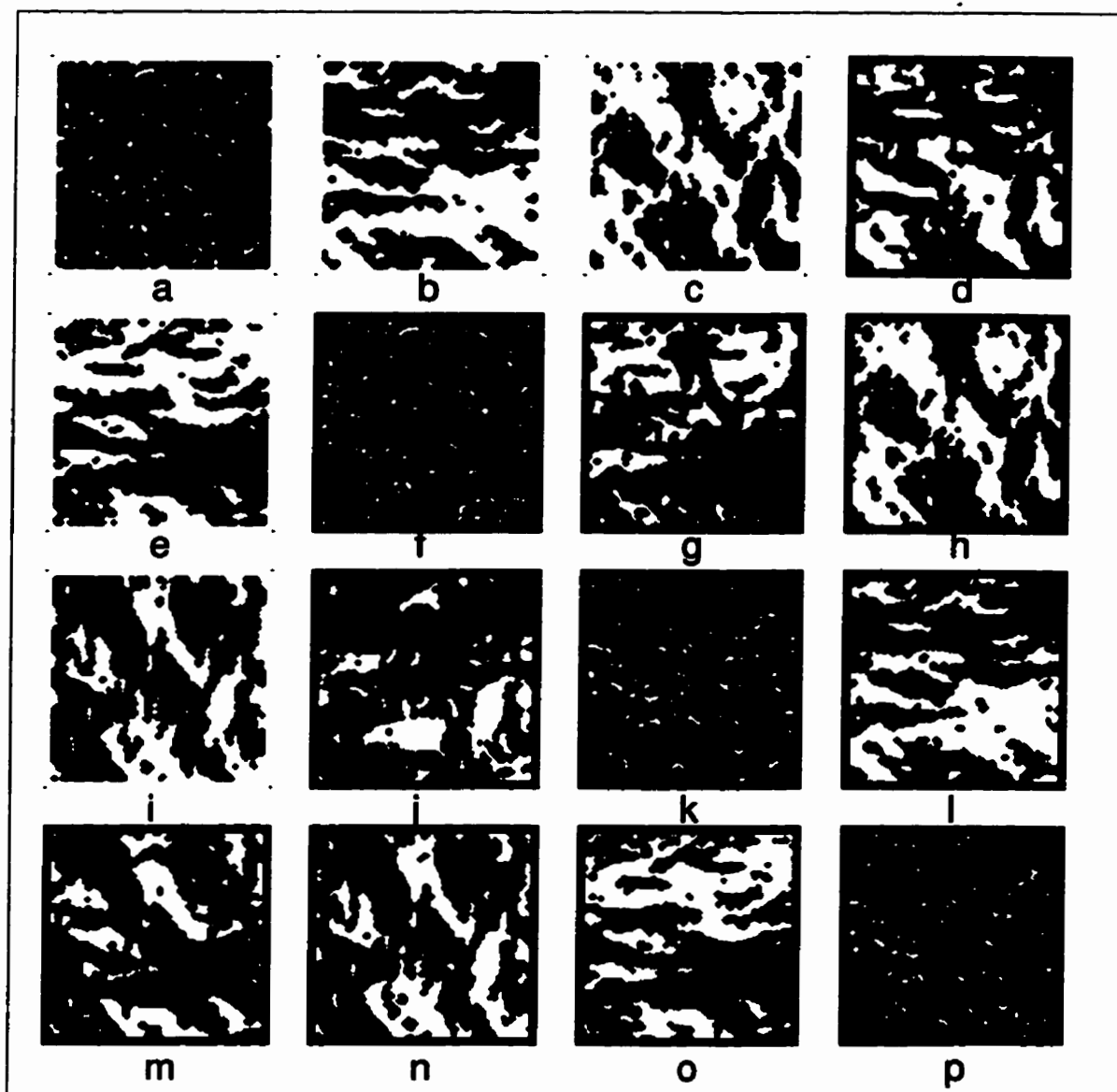


Figure 3.26: *a* to *p*: Density slices of Figure 3.1a combined with their Hamming neighbors at distance 1 after an OC operation with a “plus” SE.

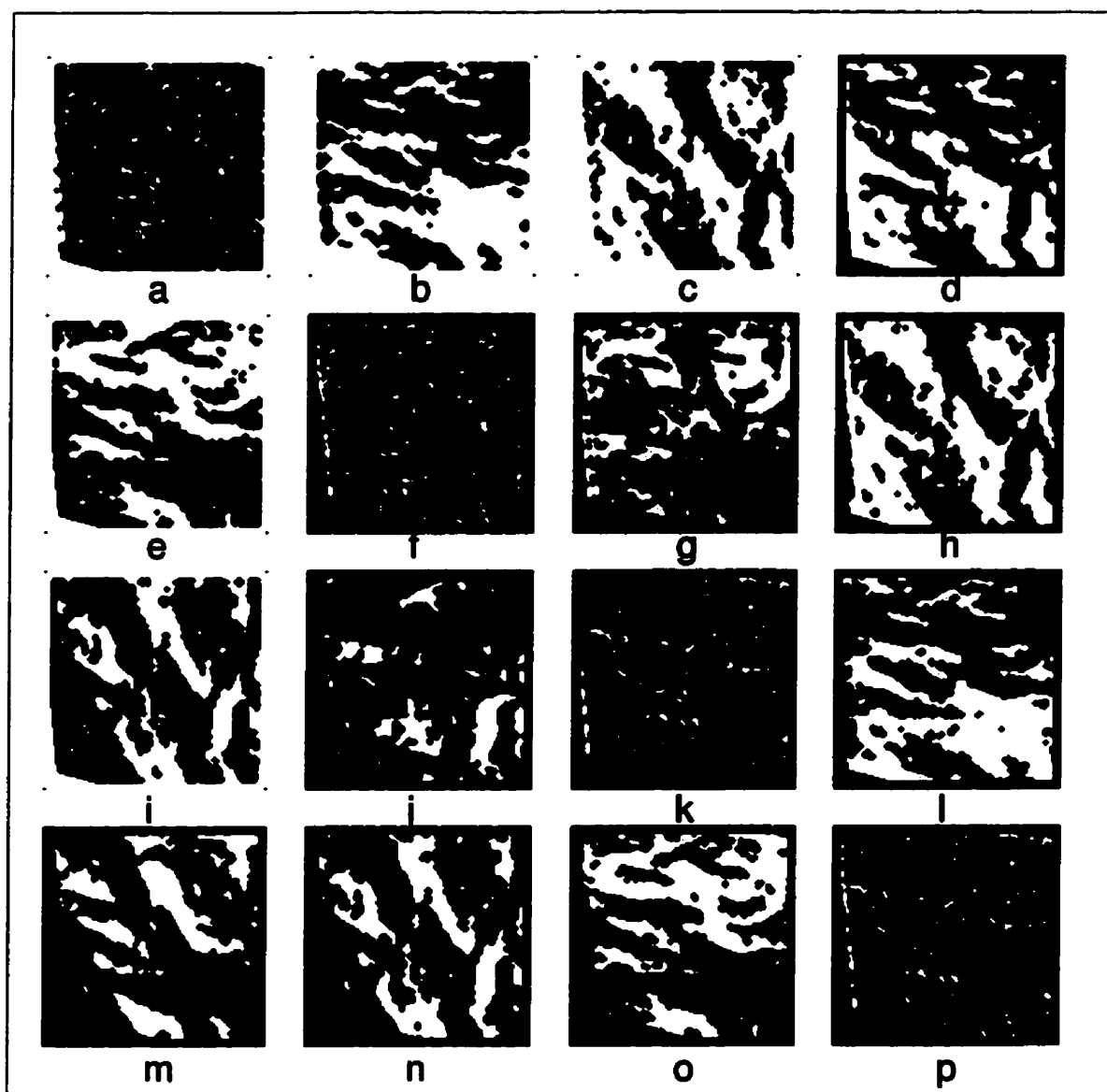


Figure 3.27: *a* to *p*: Density slices of Figure 3.1b combined with their Hamming neighbors at distance 1 after an OC operation with a “plus” SE.

Including the Hamming neighbors at a distance 1 has resulted in “whitening” most of the image area, and region demarcation is lost considerably. This is because 5 of the original slices are present in each new slice. Adding Hamming neighbors of distances 2, 3 etc., will only increase the “whitening” effect. While we have demonstrated this only for a 4-bit protocol, this will happen for other protocols as well, and hence for the subsequent chapters, we will only consider the slice itself in each case, and not any of its Hamming neighbors.

Once the OC morphological filter has been applied on each slice, individual regions have to be extracted from each slice pair. The centroids of these regions will make ideal control points [47]. This is discussed in the next section.

3.6 Region Growing or Pixel aggregation

Extraction of objects or regions from images has been widely used in pattern recognition. Region tracking was first introduced in [51]. Reference [126] contains an extensive review of region growing techniques, including a bibliography.

In our implementation, a combination of sequential tracking techniques as well as omni-directional region aggregation were used to extract each region.

Since each slice was binary, the criterion to decide if a point belonged or did not belong to a region was simple, namely that a region consist only of non-zero pixels. The given slice was raster-scanned sequentially. If a non-zero pixel was encountered, region aggregation was performed, starting with this pixel as seed to extract the complete region. This was done by growing the region in all directions. The explanation below is adapted from [101]: If O was the non-zero pixel encountered, it was used as a seed to aggregate the rest of the pixels in the region containing it. This was done by examining all its neighbors. In our implementation, we considered all 8 neighbors.

However, it is possible to implement the same with a 4-neighbor criterion.

A neighbor (x, y) is accepted as belonging to the region merely if it is non-zero. This simple criterion is only possible because of the binary nature of the images, and by what we deem as a region in our context. For more complex images, more sophisticated acceptance criteria will have to be used. Some of these are outlined in [101]. In these situations, the final region grown will depend on the starting or seed pixel. However, in our case, the same region results irrespective of which pixel is used to grow the region. Thus, when new points are accepted, they are adjoined to O and the process is continued with each of the accepted pixels as seeds and so on till no acceptable neighbors exist.

Once the pixels in a region are aggregated, we calculate the centroid (\bar{x}, \bar{y}) of each region using the following simple formula:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (3.1)$$

$$\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$$

where the summation is carried over all n pixels in the region. It must be noted that normally, given a general image which has an intensity or gray level distribution of $f(x, y)$, the centroid of the image is defined in terms of its moments [40]. However, in our case, since a region has a constant value, the above simple formula can be used.

We will now demonstrate numerically that the centroids of the regions are good control points irrespective of whether the region is fragmented or whole.

Consider Figures 3.1a and b. As has been said before, they are related by the bilinear transformation equation (2.1). We will use only 4 of the 16 possible density slices. These will be slices 3, 6, 9 and 12, which as can be seen from Figures 3.4d, g, j

and m, as well as Figures 3.5d, g, j and m have identifiable regions. The other slices have very small regions.

In the discussion below, and in subsequent chapters, when we show corresponding slices in the two images together, side-by-side, we shall always show the slices of the target image to the left and the slices of the reference image to the right. This is because, as explained in the previous chapter, the transformations were defined by an inverse mapping (e.g. equation (2.1) [122], so that given points in the target image, we can use the equations to find out the corresponding points in the reference image and not necessarily vice versa.

Figures 3.28a and b show Figure 3.5d and Figure 3.4d (density slice 3), with regions numbered. There are 23 clear regions which can be matched up. Very small or ambiguous regions have not been marked. Regions which are fragmented in one image are identified by drawing a boundary around the fragments to show that they represent one region. For e.g. Figure 3.28b has 3 such regions. The centroids of these regions were calculated using equation (3.2). The centroid of fragmented regions was calculated using the same formula.

Table (3.1) shows the positions of the centroids of the control points. Columns 2 and 3 show x and y , the positions of the centroids in the target image, columns 4 and 5 show x^* and y^* , the corresponding location for the entries in columns 2 and 3 as determined by equation (2.1), while columns 6 and 7 show \hat{x} and \hat{y} , the actual location of the centroid of the corresponding region as was marked out by eye. As can be seen, columns 4 and 5 are very close to columns 6 and 7. The actual amount of distortion or the *distortion magnitude* at the region is represented by D , given in column 9. D is the Euclidean distance between (x, y) and (x^*, y^*) . The Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) is given by E (column 9). E represents the *registration error*. From the entries in column 9, it is clear that the centroids of most of the regions are good control points. This shows

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	5.86	6.93	14.05	5.99	13.07	6.34	8.25	1.05
2	9.36	23.43	15.68	21.55	13.16	21.53	6.59	2.52
3	5.00	38.00	9.38	35.88	7.541	36.70	4.86	2.01
4	11.00	41.50	15.04	38.90	14.79	38.79	4.81	0.28
5	19.52	48.87	22.64	45.63	22.00	46.12	4.50	0.81
6	28.19	41.23	32.53	37.67	32.55	36.69	5.61	0.99
7	6.50	80.50	5.67	76.64	5.39	77.72	3.95	1.12
8	22.72	83.12	20.81	79.13	21.40	79.78	4.42	0.87
9	11.15	99.21	7.69	94.72	7.70	94.70	5.68	0.02
10	52.06	53.67	54.40	49.03	55.22	49.71	5.20	1.07
11	51.60	68.81	51.07	64.50	51.37	64.49	4.34	0.30
12	47.78	95.22	42.48	91.40	42.33	89.61	6.53	1.80
13	45.93	106.80	38.66	103.10	39.00	103.00	8.16	0.35
14	60.35	106.80	51.91	103.50	50.81	104.20	9.05	1.33
15	59.31	116.30	49.04	113.30	49.07	113.20	10.70	0.11
16	76.05	121.50	63.02	119.50	62.96	121.00	13.19	1.59
17	103.40	113.60	89.71	112.20	91.64	113.00	13.81	2.12
18	91.31	82.78	86.31	78.70	88.86	80.45	6.50	3.09
19	64.78	26.55	72.75	20.37	73.78	24.61	10.08	4.37
20	76.60	15.53	87.50	7.88	86.00	8.50	13.32	1.63
21	110.40	43.21	115.40	35.23	115.60	36.28	9.45	1.07
22	104.00	27.11	113.30	18.02	106.30	18.24	12.99	7.01
¹ 22	94.78	26.08	104.00	17.60	106.30	18.24	12.51	2.40
23	39.77	14.77	48.85	10.43	48.50	10.50	10.06	0.35

¹ Centroid of 22nd region after removing extraneous portion (Refer Figure 3.29)

Table 3.1: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

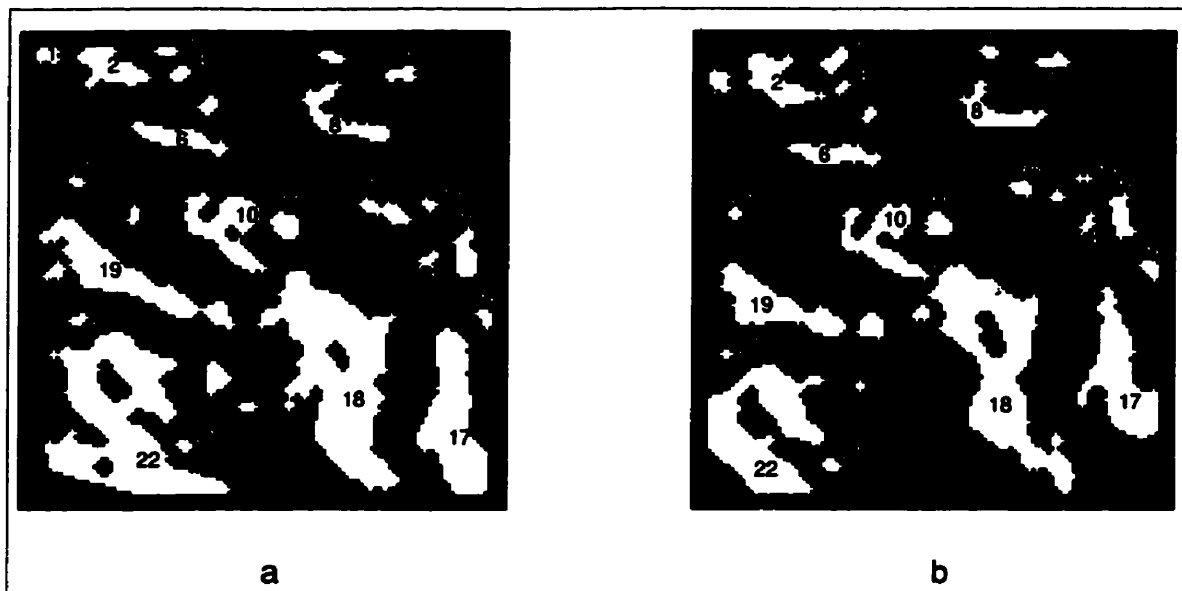


Figure 3.28: *a*: Figure 3.5d shown with 23 prominent regions numbered. *b*: Figure 3.4d with corresponding regions in *a* shown. Fragmented regions are identified by drawing a boundary around the fragments.

that these regions could be easily matched up by eye irrespective of the amount of distortion. E is small for most regions. For region 22 alone the error is extremely large (7 pixels). This is because of the extraneous portion present in the 22nd region in Figure 3.28a which is not present in Figure 3.28b. To prove this point, we have presented Figures 3.29a and b.

Figure 3.29b is the same as Figure 3.28b, while Figure 3.29a was obtained by removing the extraneous portion from region 22. Once this is done and the centroid is calculated again, (last row in table (3.1), we find that E is much smaller, showing that the reason for the earlier large value of E was the extraneous portion in the 22nd region. This extraneous portion is not a peculiarity of the starbyte transformation, but was only created because of the fact that Figure 2.3b was generated from Figure 2.2b using an artificial transformation, which rounded values outside the boundary of the

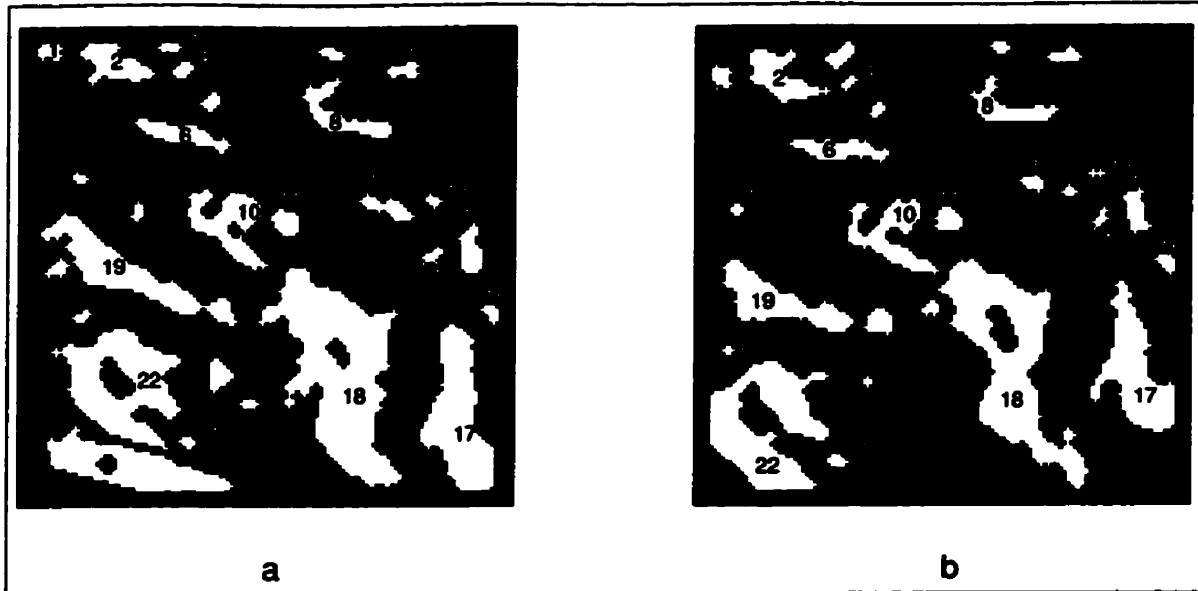


Figure 3.29: *a*: Figure 3.28 shown after extraneous portion in region 22 was removed.
b: Same as Figure 3.28

image to 0. This can be verified by observing that the profile of the extraneous region is exactly the same as the profile or boundary of the bilinear transformation (compare Figure 2.2b and Figure 2.3b). Other regions (for e.g. 18 and 19) also have larger errors (about 3 or 4 pixels) when compared to the other regions and the reason for this is also the presence of extraneous portions present in one region, which are not present in the corresponding region in the other image. We shall discuss in the next chapter, UICP, a user-interactive program for identifying and marking regions which will also allow us to slice out extraneous portions, so that the centroids are more accurate.

Observing columns 8 and 9, it is seen that there is no connection or correlation between D , the distortion magnitude and the E , the registration error. This lack of correlation between D and E is also confirmed by calculating the correlation coefficient between the two vectors. The correlation coefficient r between the two vectors

is given by

$$r = \frac{\sum_{i=0}^{n-1} ((D(i) - \bar{D})(E(i) - \bar{E}))}{\sqrt{\sum_{i=0}^{n-1} (D(i) - \bar{D})^2 \sum_{i=0}^{n-1} (E(i) - \bar{E})^2}} \quad (3.2)$$

In some sense, r indicates the angle between the two vectors. Values of r range from 0 to 1. Vectors with high correlation have values close to 1, while vectors with little correlation have values close to 0. In our case, for the D and E columns in Figure 3.1, we obtained an r value of 0.2902. This shows that there is little correlation between the two columns.

Figures 3.30a and b show Figure 3.5g and Figure 3.4g (density slice 6) with regions marked as before. Here there were 14 clear regions. Table (3.2) gives the details regarding these regions as before.

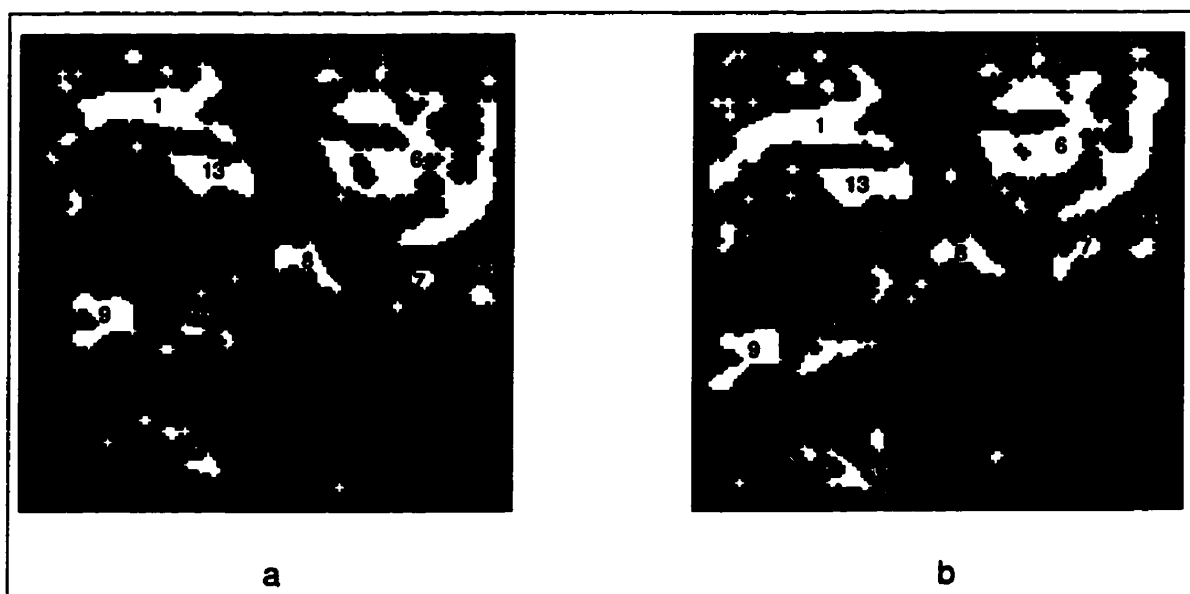


Figure 3.30: *a*: Figure 3.5g shown with 14 prominent regions numbered. *b*: Figure 3.4g with corresponding regions in *a* shown. Fragmented regions are identified by drawing a boundary around the fragments.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	19.29	33.93	24.53	31.03	25.27	28.22	5.99	2.91
2	5.50	28.00	11.10	26.25	11.40	26.40	5.87	0.33
3	10.61	79.43	9.74	75.59	9.63	75.95	3.94	0.38
4	9.77	93.23	7.17	88.91	6.79	88.36	5.04	0.68
5	12.00	121.00	5.62	115.80	5.13	115.50	8.21	0.61
6	33.77	102.20	28.23	98.16	27.12	98.84	6.86	1.31
7	66.80	103.00	58.63	99.80	59.64	98.51	8.79	1.64
8	60.94	73.79	59.16	69.46	59.47	70.16	4.69	0.76
9	75.70	20.55	85.43	13.23	85.70	13.88	12.17	0.70
10	80.91	45.35	85.10	39.06	85.39	35.68	7.55	3.40
11	112.30	44.49	117.00	36.53	116.70	38.07	9.26	1.57
12	44.72	13.24	54.34	8.43	54.03	8.42	10.76	0.31

Table 3.2: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

Figures 3.31a and b show Figure 3.5j and Figure 3.4j (density slice 9) with regions marked as before. In this case there were 14 regions. Details regarding the regions are given in table (3.3).

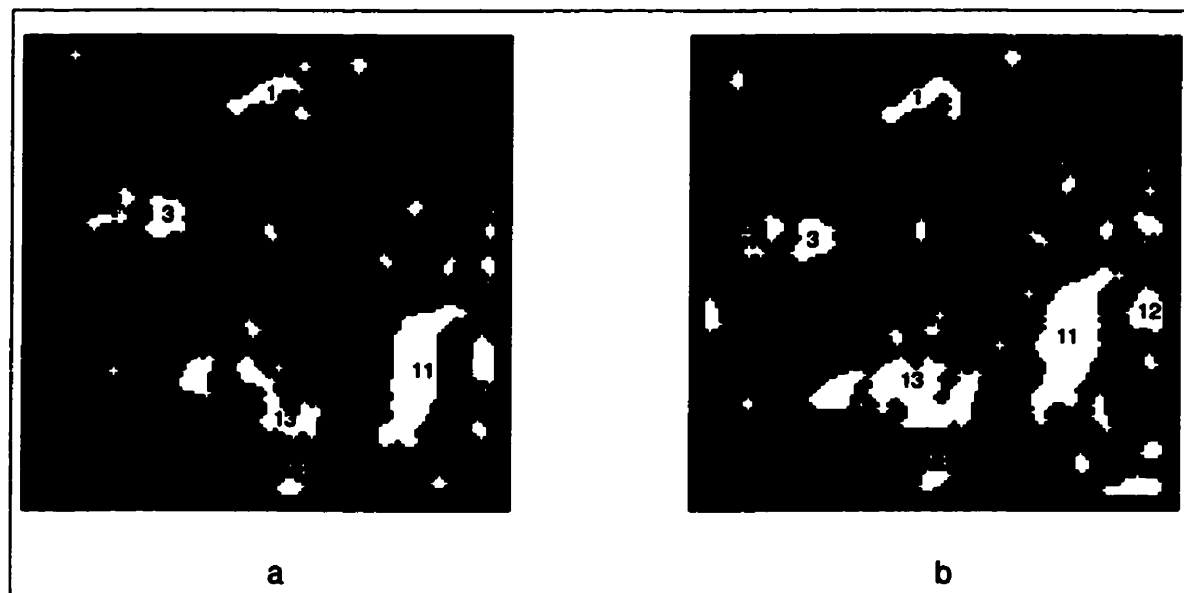


Figure 3.31: *a*: Figure 3.5j shown with 14 prominent regions numbered. *b*: Figure 3.4j with corresponding regions in *a* shown. Fragmented regions are identified by drawing a boundary around the fragments.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	16.00	62.97	17.17	59.53	17.10	60.11	3.64	0.59
2	7.63	87.12	5.92	83.01	5.65	83.00	4.46	0.27
3	48.64	36.15	54.24	31.36	54.42	31.57	7.37	0.28
4	47.08	22.86	55.07	17.97	53.29	19.78	9.37	2.54
5	52.53	63.40	53.01	58.95	52.00	59.00	4.47	1.01
6	45.80	102.20	39.37	98.44	39.50	98.50	7.45	0.15
7	60.00	94.50	54.07	90.83	54.00	90.50	6.98	0.34
8	61.50	111.30	52.04	108.20	51.72	108.60	9.95	0.52
9	52.00	122.00	41.34	118.80	41.00	120.00	11.13	1.25
10	61.28	121.60	49.74	118.90	50.08	119.60	11.86	0.77
11	90.38	102.50	80.56	99.78	81.78	98.86	10.17	1.52
12	86.72	120.50	72.83	118.90	73.57	119.20	13.98	0.80
13	95.56	59.83	96.20	53.94	94.86	55.71	5.93	2.22
14	121.20	69.24	118.80	63.59	119.20	62.69	6.15	1.00

Table 3.3: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

Figures 3.32a and b show Figure 3.5m and Figure 3.4m (density slice 12) with regions marked as before. In this case there were 15 regions. Details regarding the regions are given in table (3.4). Note that in almost all cases, E values are low.

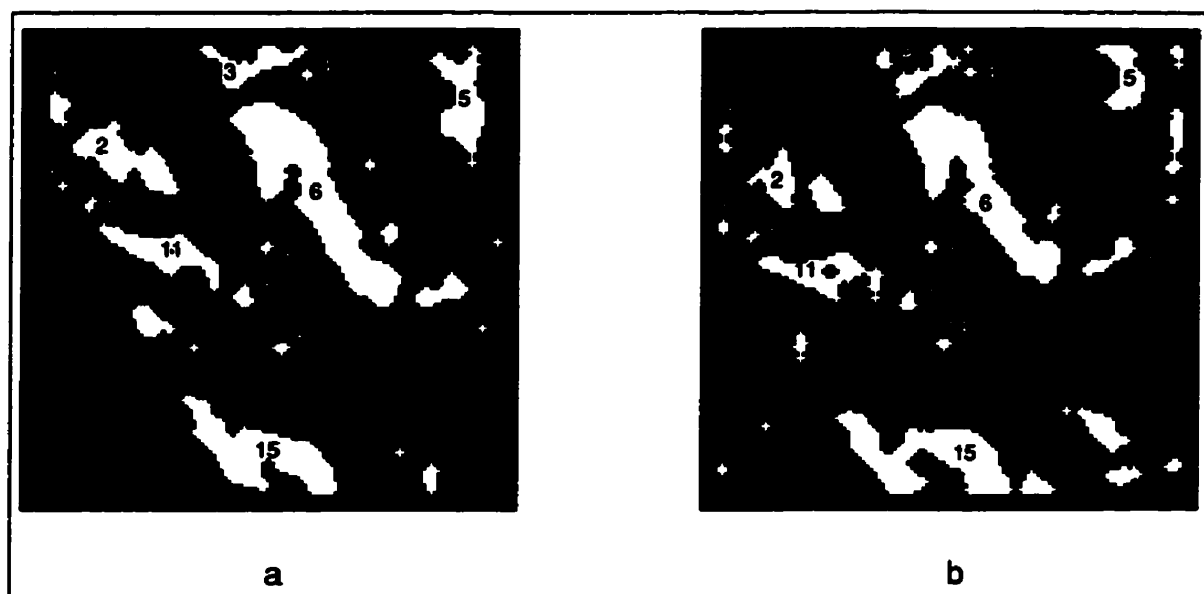


Figure 3.32: *a*: Figure 3.5m shown with 15 prominent regions numbered. *b*: Figure 3.4m with corresponding regions in *a* shown. Fragmented regions are identified by drawing a boundary around the fragments.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	20.44	9.03	29.29	6.56	28.50	5.30	9.19	1.49
2	33.92	26.63	40.74	22.79	41.21	23.50	7.83	0.85
3	8.54	58.99	10.34	55.90	10.32	55.33	3.58	0.57
4	11.50	73.00	11.44	69.35	11.00	68.50	3.65	0.96
5	18.78	112.50	12.91	107.80	12.43	108.00	7.49	0.52
6	43.34	74.89	41.94	70.80	41.36	70.70	4.32	0.59
7	53.75	94.21	48.27	90.45	48.57	90.50	6.65	0.31
8	69.27	107.60	59.94	104.60	60.17	104.80	9.79	0.29
9	41.00	10.00	50.97	5.51	52.00	5.00	10.94	1.15
10	46.00	17.36	54.94	12.49	54.50	12.50	10.19	0.44
11	58.69	37.06	64.34	31.66	64.48	30.88	7.81	0.79
12	57.30	63.30	57.72	58.72	57.50	58.50	4.60	0.31
13	70.44	57.63	71.87	52.44	72.05	52.59	5.39	0.24
14	84.17	67.00	83.24	61.92	83.17	62.00	5.16	0.11
15	112.00	61.67	112.00	55.44	113.30	57.84	6.23	2.72

Table 3.4: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

3.7 Conclusions

The corresponding density slices obtained after applying the starbyte transformation to the two original images (reference and target) can be treated independently of each other. A 4-bit protocol (which produces starbyte images with clearly demarcable boundaries between regions) can be used and results in a manageable set of slices. These slices, after being “cleaned up” with an OC filter, can be used to generate a large number of control points. We have shown numerically that matching regions in corresponding images can be done quite easily visually. Since the starbyte transformation is based on bit patterns, changes in bit values may result in different density levels, which will be manifested as fragmented regions. We have discussed Hamming neighbors and concluded that combining slices with their Hamming neighbors results in whitening and loss of region demarcation. Thus for subsequent chapters, we will only consider the slice itself in each case, and not any of its Hamming neighbors. We have shown numerically that the centroids of the corresponding regions are good control points irrespective of whether the region is fragmented or whole. In the next chapter, we shall discuss a user-interactive program called UICP, by which control points can be detected manually, extraneous portions in regions can be removed and fragmented portions of a region can be marked together as belonging to one region.

Chapter 4

MANUALLY-AIDED DETECTION OF CONTROL POINTS

4.1 Introduction

In the previous chapter, we demonstrated that given a pair of images to register (reference and target), even a simple 4-bit protocol starbyte transformation returned images with clearly demarcable regions. Corresponding density slices of the starbyte images could be matched visually after a simple OC morphological binary filter. We demonstrated that regions present in slices of one starbyte image could be easily matched up to regions in the other starbyte image. We also demonstrated that the centroids of the regions were good control points. These density slices could be treated independently and hence the algorithm is an excellent candidate for parallelization.

In this chapter, we discuss UICP, a simple interactive program which allows a user to mark corresponding regions, separate concatenated regions as well as group fragmented portions of a region. This data can be assembled together, the selected regions can be aggregated, and their centroids calculated to extract control points. We shall discuss the interactive program and demonstrate these different aspects using the same distortions discussed in the previous two chapters. All starbyte transformations considered henceforth are obtained using 9×9 neighborhoods using either protocol 1 or 3 in Chapter 2. Hence when we refer to starbyte images we shall merely refer to the protocol used and not the neighborhood size.

The examples considered here and in subsequent chapters use known transformations. These transformations are expressed as inverse mappings so that given a point in the target image, we can calculate the corresponding point in the reference image. As explained in the previous chapter, we shall use the following notation to refer to points in the target, unwarped and reference image. We shall refer to points in the target image by (x, y) , their *calculated* points in the reference image by (x^*, y^*) , and the *actual* points in the reference image by (\hat{x}, \hat{y}) . Ideally (x^*, y^*) and (\hat{x}, \hat{y}) should be very close to each other. The Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) is denoted by E , the registration error magnitude. The Euclidean distance between (x, y) and (x^*, y^*) is D , the distortion magnitude.

4.2 Description of UICP

Firstly the individual density slices of an ST are cleaned up using the OC operation discussed earlier. Since the individual density values are independent, the cleaned-up slices can be simply added together to form a composite cleaned-up starbyte image, which is ready to be loaded into UICP. Thus individual slice for each of the given pair of STs is cleaned-up and “put back” into the respective image. This image pair is then ready to be loaded into UICP. Since we have used 4-bit protocols throughout this thesis, there will be 16 density slices to be considered (0 to 15). The level 0 is never considered because it was found to be noisy even after the OC operation. Hence only levels 1 to 15 were considered throughout.

UICP, written using Matlab Version 4.2c [76], is a tool to extract identifiable regions from each slice. The centroids of these regions will then form corresponding points which can be used for image registration. Identification of the corresponding or matching regions is done by eye. In some cases, regions separated in one image are

concatenated or joined in the other. Hence, apart from allowing the user to merely select matching regions, UICP also allows users to separate concatenated regions. Since this operation can be done any number of times, in effect, it allows the user to edit regions before they be selected for matching.

The main features of UICP are discussed below followed by a detailed example to illustrate the use of these features.

4.2.1 Separating concatenated regions

UICP first allows the user to load the cleaned-up STs. Corresponding density slices (1 to 15) of the two images are loaded side-by-side, with the slice belonging to the target image loaded on the left, and the one corresponding to the reference on the right. UICP then allows the user to separate concatenated regions. The user can study the two slices and decide which regions are to be separated.

This step is necessary for one of the following reasons:

1. Regions separated in one image are concatenated or joined in the other. In this case, the user has two choices: a. Consider the concatenated region as one consolidated region in one image and consider the separated regions in the other image as *fragments* of a region. Doing this will give us only one control point (namely, the centroid of the consolidated region).
b. Separate the concatenated regions. Then each separated region gives us one control point.
2. A region may be concatenated with an extraneous portion which is not present in the other image. In this case the only option is to slice out the extra portion.

Regions are separated by defining a “line of separation”. This is done by clicking on a start and end point in the area where the regions are to be separated. The

regions are separated or “cut” along this line. Clicking is done using the 1st or 2nd mouse button. For every line of separation, UICP accepts two clicked points. Several lines of separation in different parts of the slice can be specified one after the other. The 3rd mouse button followed by pressing the “RETURN” key is used to terminate the process. Region separation is first done on the left-hand side image and then the right-hand one. If there are no regions to be separated, the user can straightaway press the 3rd mouse button and press RETURN to signal this to UICP.

4.2.2 Identifying matching regions in the two slices

Once the region separation phase is completed for a slice pair, then matching regions can be specified. This is done by observing the two slices, and identifying matching regions by eye. Then the cursor is positioned over any region and the 1st or 2nd mouse button followed by the “RETURN” key is pressed to signal a “click”. The main advantage of using UICP is that the user can click anywhere on a region in order to specify a region. Hence there is no need to exactly position the cursor over any particular portion of a region in order to specify it. This is because, as explained in the previous chapter, since each slice is binary, region aggregation is very simple and the same region is aggregated irrespective of which seed point is specified. Essentially, the “clicked” points specified through UICP form the seed points for region aggregation which will be performed later. Regions can be continuously selected one after the other, until the 3rd mouse button followed by RETURN is pressed. This signals that the user has finished with the current slice pair and that the next one can be loaded.

As before, UICP allows the user to click on points in the left image first, and then the right image afterwards. If a particular slice pair has no distinguishable regions, the user can press the 3rd mouse button and the RETURN key over the left-hand side image to signal this to UICP.

Joining fragments of a region together

UICP also allows the user to join fragments of a region together. This property is essential because regions joined up or concatenated in one slice may be fragmented in the other. While one option is to consider a fragment itself as one region, as was discussed earlier, the other option is to consider all the fragments together as belonging to one region. While this gives us fewer control points as compared to specifying each fragment as a region, the centroid thus obtained will be more accurate because in a sense it will be a "centroid of centroids". Also, the user is saved the trouble of identifying lines of separation. Fragments are specified as such by positioning the cursor over each fragment and performing a "click". The "RETURN" key is pressed only after each fragment is clicked. UICP then records each of these clicked points as belonging to the same region, and hence in the region aggregation stage, they are all accumulated together for centroid calculation.

We shall now use the same example as was discussed in the last part of the previous chapter to illustrate the above two features of UICP. This was the experiment where the distortion relating the two original images was a bilinear distortion given by equation (2.1)). We had considered the cleaned-up slices of Figure 3.1a and b (STs (protocol 3)) of Figure 2.2b and Figure 2.3b). In the previous chapter, only slices 3, 6, 9 and 12 were considered because these contained many more identifiable regions as compared to the other slices. Now we shall also consider slice 4 here. No matching regions were specified in the other slices. A total of 89 control points was obtained using UICP. We shall present the results of using UICP, slice by slice for this example. We shall show the regions marked and numbered for each slice as before. Since the region separation operation alters a slice, regions will be marked and numbered on the altered slice. For this example, we shall also present detailed figures with lines of

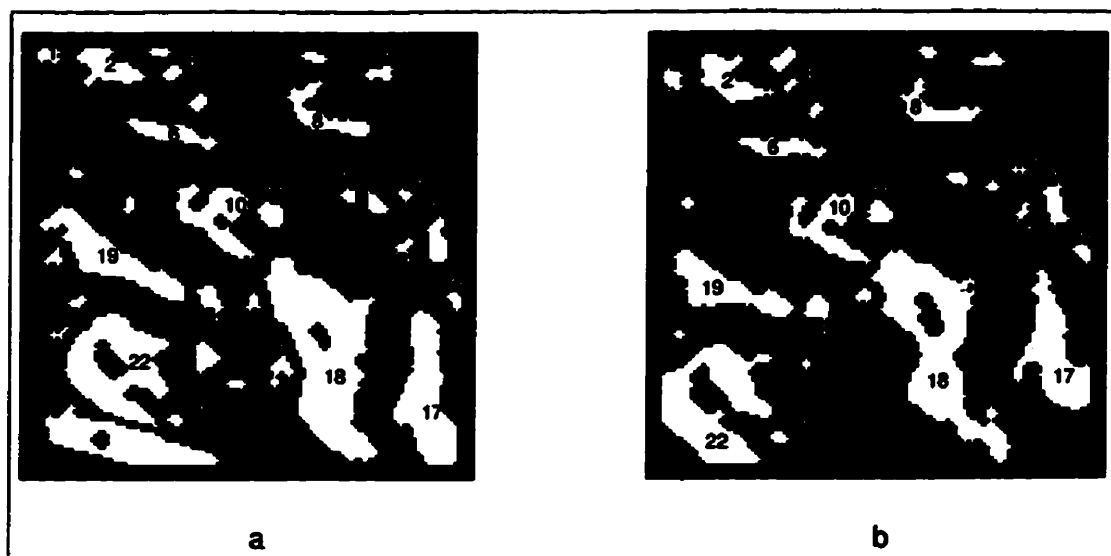


Figure 4.1: *a* and *b*: 26 regions obtained using UICP, shown for slice 3 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. *a* was obtained by modifying Figure 3.28a using UICP, while *b* is the same as Figure 3.28b.

separation drawn, to illustrate how the alteration was done. Following this we shall present the results obtained using UICP for the same pair of original images, but with a startbyte transformation using protocol 1. We shall also present the results for the other five distortions discussed in the previous two chapters. As explained before, left-hand slices are target image slices, while right-hand slices are reference image slices.

4.3 Bilinear distortion, protocol 3

4.3.1 Slice 3

Figure 4.1 shows the marked and numbered regions in the 3rd slice. This time there are 26 control points instead of 23. The extra regions were obtained by separation of concatenated regions and also slicing out extraneous portions.

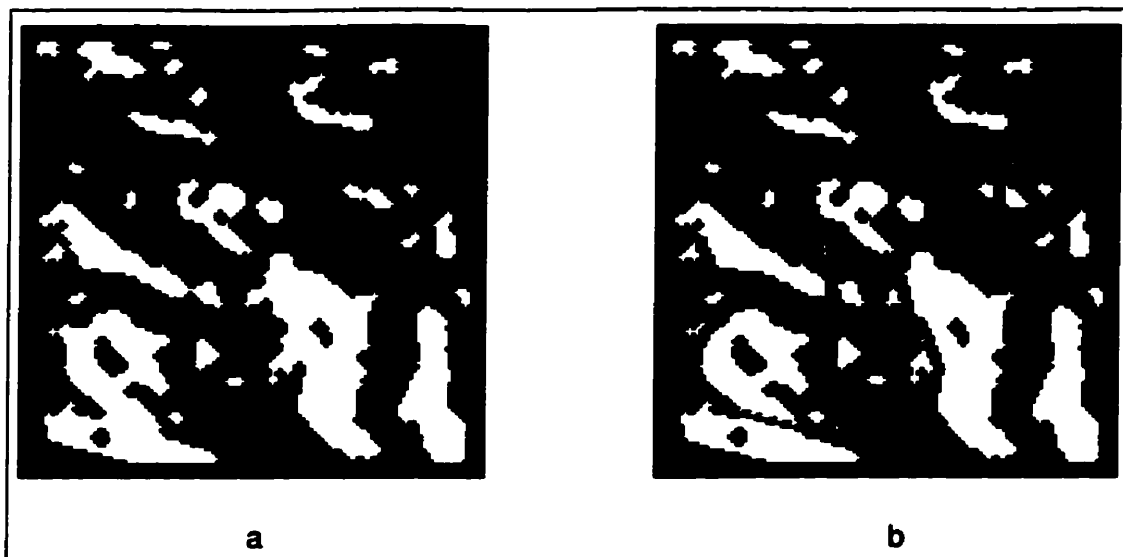


Figure 4.2: *a*: Same as Figure 3.28a. *b*: *a* modified using seven region separation operations using UICP. The seven lines of separations have also been drawn for clarity.

Figure 4.2 shows how concatenated regions were separated for this case. Figure 4.2a is the same as Figure 3.28a of the previous chapter (i.e. merely the cleaned-up slice with no regions separated). We showed in the previous chapter that when the extraneous portion in region 22 is sliced out (Figure 3.29a), then the 22nd control point becomes more accurate, because this extraneous portion is not present in the corresponding slice in the reference image (see Figure 3.28b). Figure 4.2b shows the separation of several other concatenated regions in the slice. The lines of separation have also been drawn for clarity. In some cases (line 1, 2, 3 and 7), this was done to slice out extraneous portions. In the other cases (lines 4, 5 and 6), these were used to generate extra control points. For example, line 4 joined regions 19 and 24, and by separating them, an extra control point was obtained. The same was the case with regions 18 and 25, as well as with 12 and 26.

The control points (centroids of the regions) are tabulated in table (4.1). The

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	5.86	6.93	14.05	5.99	13.07	6.35	8.25	1.04
2	9.36	23.43	15.68	21.55	13.16	21.53	6.59	2.52
3	5.00	38.00	9.38	35.88	7.54	36.70	4.86	2.01
4	11.00	41.50	15.04	38.90	14.79	38.79	4.81	0.28
5	19.52	48.87	22.64	45.63	22.00	46.12	4.50	0.81
6	28.19	41.23	32.53	37.67	32.55	36.69	5.61	0.99
7	6.50	80.50	5.67	76.64	5.39	77.72	3.95	1.12
8	22.72	83.12	20.81	79.13	21.40	79.78	4.42	0.87
9	11.15	99.21	7.69	94.72	7.70	94.70	5.68	0.02
10	52.06	53.67	54.40	49.03	55.22	49.71	5.20	1.07
11	51.60	68.81	51.07	64.50	51.37	64.49	4.34	0.30
*12	46.24	90.41	41.91	86.50	41.58	86.42	5.83	0.34
13	45.93	106.80	38.66	103.10	39.00	103.00	8.16	0.35
14	60.35	106.80	51.91	103.50	50.81	104.20	9.05	1.33
15	59.31	116.30	49.04	113.30	49.07	113.20	10.70	0.11
16	76.05	121.50	63.02	119.50	62.96	121.00	13.19	1.59
17	103.40	113.60	89.71	112.20	91.64	113.00	13.81	2.12
*18	91.61	83.99	86.29	79.99	89.14	80.86	6.65	2.98
*19	63.45	24.36	71.82	18.21	73.02	21.01	10.39	3.05
20	76.60	15.53	87.50	7.88	86.00	8.50	13.32	1.63
21	110.40	43.21	115.40	35.23	115.60	36.28	9.45	1.07
*22	95.20	26.78	104.20	18.32	106.30	18.24	12.38	2.05
23	39.77	14.77	48.85	10.43	48.50	10.50	10.06	0.35
*24	75.42	52.35	77.99	46.71	78.49	46.96	6.20	0.56
*25	76.69	63.94	76.63	58.86	75.06	60.17	5.07	2.04
*26	48.71	98.26	42.79	94.50	43.83	96.00	7.01	1.83

* Regions changed or added because of separation of concatenated regions as compared to table (3.1).

Table 4.1: *Column 1: Region number. Column 2 and 3: x and y coordinates of centroids of the regions in the target image. Columns 4 and 5: x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). Column 6 and 7: x and y coordinates of centroids of the regions in the reference image. Column 8: D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). Column 9: E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).*

symbols have the same meaning as before. Regions which have been changed or added because of the separation of concatenated regions have been marked with an asterisk. As a result of defining lines of separation, there were no fragmented regions for this slice. It can be observed from the table that regions close to the border have larger errors. This is because portions of the target image were outside the border of the original images, which created a “clipping” effect. The clipping effect can be easily seen for regions close to the border in all the slices. This problem is taken care of in the next chapter which describes an automatic program to extract control points. In that case, regions close to the border of the image are not considered.

4.3.2 Slice 4

Figure 4.3 shows the 9 regions identified in slice 4. The centroids of these regions are tabulated in table (4.2).



Figure 4.3: *a* and *b*: 9 regions obtained using UICP, shown for slice 4 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	6.53	57.80	8.50	54.82	9.47	52.33	3.57	2.67
2	21.36	13.00	29.70	10.37	28.77	8.77	8.74	1.85
3	31.00	10.50	40.28	6.98	41.00	8.50	9.92	1.68
4	30.50	31.50	36.42	27.88	37.30	28.30	6.94	0.98
5	41.00	69.50	40.62	65.41	40.50	64.50	4.11	0.91
6	33.50	87.00	30.46	82.99	31.00	84.00	5.03	1.14
7	48.00	85.50	44.48	81.52	45.00	81.00	5.31	0.74
8	51.00	91.00	46.30	87.14	46.50	87.50	6.09	0.41
9	66.32	99.74	58.88	96.37	58.00	95.00	8.17	1.62

Table 4.2: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

4.3.3 Slice 6

Figure 4.4 shows 18 regions marked and numbered in slice 6. Four extra regions were obtained as compared to Figure 3.30 because of the region separation operation. Earlier region 1 on the left-hand side slice had 4 fragments. Now it has only 3 because, one of the fragments has become region 15. In order to generate an extra control point, a part of region 1 was sliced out to create a new region i.e. region 16. In the same way, portions of region 6 earlier were separated out to create regions 17 and 18. Regions 7, 10 and 11 on the left slice continued to have fragmented portions and these fragments have been identified together by drawing a boundary around them. On the right-hand side, Region 6, which had 2 fragments earlier was broken into 3 regions. Region 11 remained unchanged and has 2 fragments.

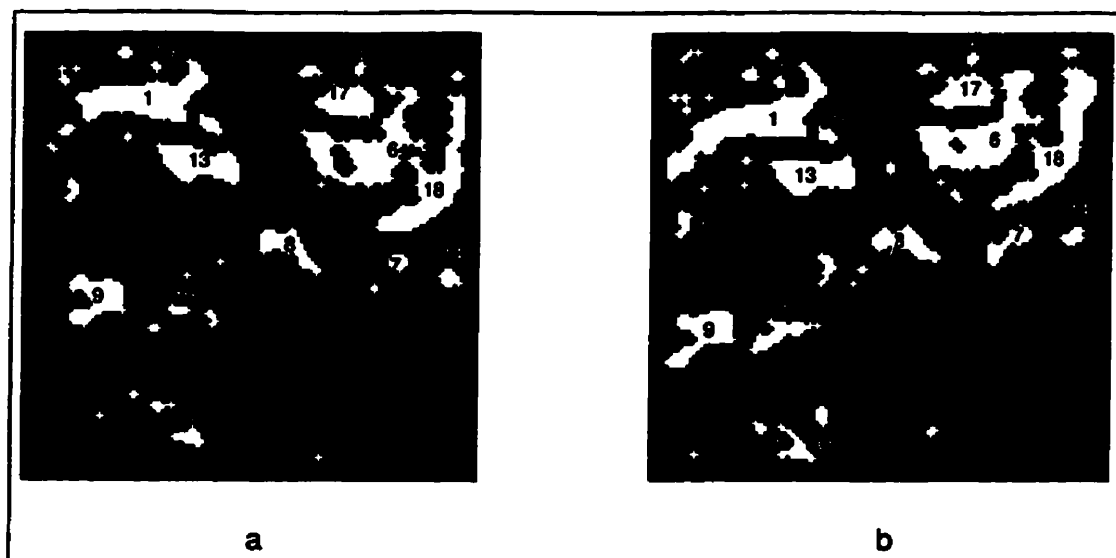


Figure 4.4: *a* and *b*: 18 regions obtained using UICP, shown for slice 6 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. *a* and *b* were obtained by modifying Figures 3.30a and b respectively using UICP.

Thus region separation was performed for both left and right hand slices. Figure 4.5 shows the separation of regions for the left-hand side image. Figure 4.5a is the same as Figure 3.30a of the previous chapter, while Figure 4.5b was obtained by performing 3 region separations. Similarly for the right-hand side image, Figure 4.6b was obtained from Figure 4.6a by 2 region separation operations (Figure 4.6a, being the same as Figure 3.30b).

Table (4.3) shows the centroids and D and E values for the regions. Regions changed or added because of the region separation operation have been marked by an asterisk.

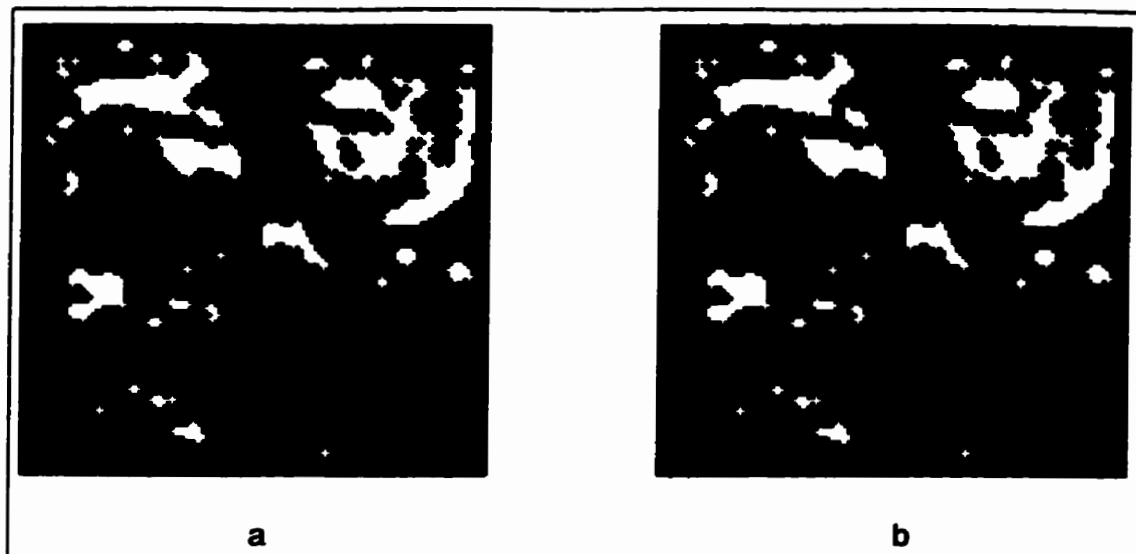


Figure 4.5: *a*: Same as Figure 3.30a. *b*: *a* modified using three region separation operations using UICP. The three lines of separations have also been drawn for clarity.

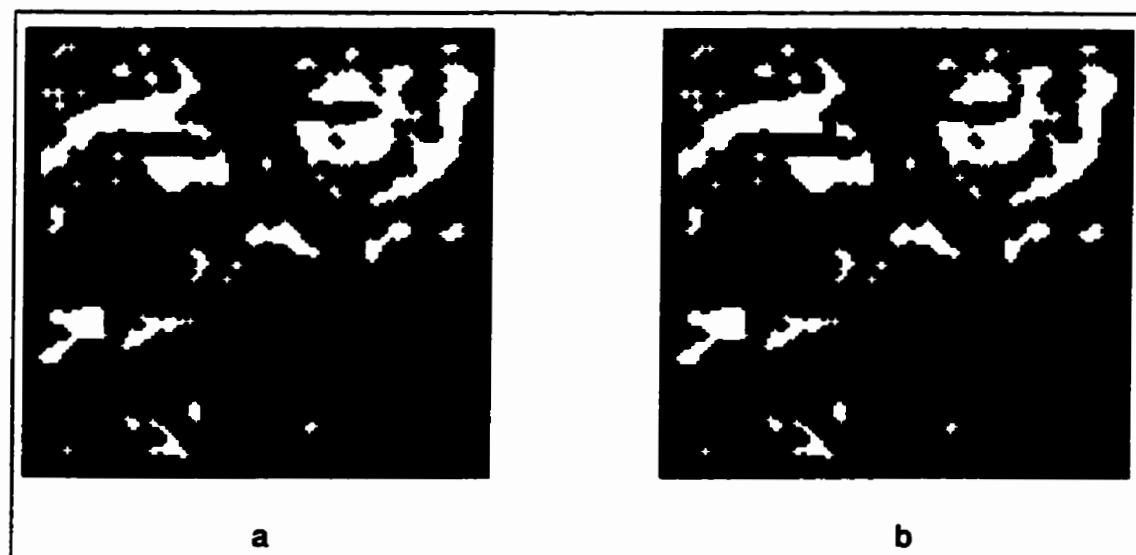


Figure 4.6: *a*: Same as Figure 3.30b. *b*: *a* modified using two region separation operations using UICP. The two lines of separations have also been drawn for clarity.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
*1	18.92	32.15	24.41	29.31	25.39	26.65	6.17	2.84
2	5.50	28.00	11.10	26.25	11.40	26.40	5.87	0.33
3	10.61	79.43	9.74	75.59	9.63	75.95	3.94	0.38
4	9.77	93.23	7.17	88.91	6.79	88.36	5.04	0.68
5	12.00	121.00	5.62	115.80	5.13	115.50	8.21	0.61
*6	32.15	95.75	27.78	91.70	27.66	91.18	5.97	0.53
7	66.80	103.00	58.63	99.80	59.64	98.51	8.79	1.64
8	60.94	73.79	59.16	69.46	59.47	70.16	4.69	0.76
9	75.70	20.55	85.43	13.23	85.70	13.88	12.17	0.70
10	80.91	45.35	85.10	39.06	85.39	35.68	7.55	3.40
11	112.30	44.49	117.00	36.53	116.70	38.07	9.26	1.57
12	44.72	13.24	54.34	8.43	54.03	8.42	10.76	0.31
13	9.30	36.70	13.93	34.36	13.62	34.62	5.19	0.40
14	25.90	51.81	28.60	48.23	28.75	48.21	4.48	0.15
*15	19.15	89.74	16.47	85.61	16.43	86.13	4.92	0.52
*16	41.41	114.30	33.19	110.50	30.51	112.00	9.05	3.04
*17	37.00	49.07	40.16	45.02	40.05	44.75	5.13	0.29
*18	69.66	119.40	57.73	117.00	57.58	116.80	12.17	0.20

* Regions changed or added because of separation of concatenated regions as compared to table (3.2).

Table 4.3: *Column 1: Region number. Column 2 and 3: x and y coordinates of centroids of the regions in the target image. Columns 4 and 5: x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). Column 6 and 7: x and y coordinates of centroids of the regions in the reference image. Column 8: D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). Column 9: E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).*

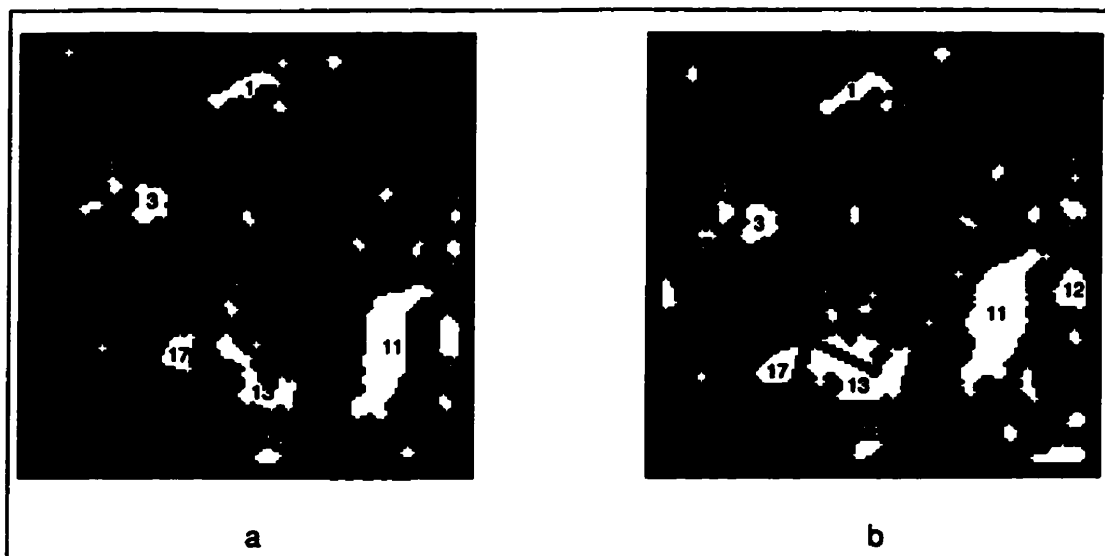


Figure 4.7: *a* and *b*: 18 regions obtained using UICP, shown for slice 9 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. *a* and *b* were obtained by modifying Figures 3.31a and b respectively using UICP.

4.3.4 Slice 9

Figure 4.7 shows 18 regions marked and numbered in slice 9. There are four more regions as compared to Figure 3.31. Region 15 was created by a region separation operation on region 1 in Figure 3.31. Doing this also rendered the earlier control point due to region 1 more accurate. (Compare E values for region 1 in table (3.3) and table (4.4).) Region 16 was earlier a fragment in region 13. Region 17 was obtained by a region separation operation on the right-hand side image. Also region 13 itself was made more accurate by slicing out an unwanted portion not present on the left-hand side image. Region 4 which consisted of one region on the left-hand side and 2 fragments on the right-hand side earlier (Figure 3.31) was divided on the left-hand side by a region separation operation. This gave us region 18.

Thus one region separation operation was performed for the left hand side image

(Refer Figure 4.8) and three region separation operations were performed for the right-hand side image (Refer Figure 4.9). The centroids and D and E values are tabulated in Table (4.4). Regions changed or added because of the region separation operation have been marked by an asterisk in Table (4.4).

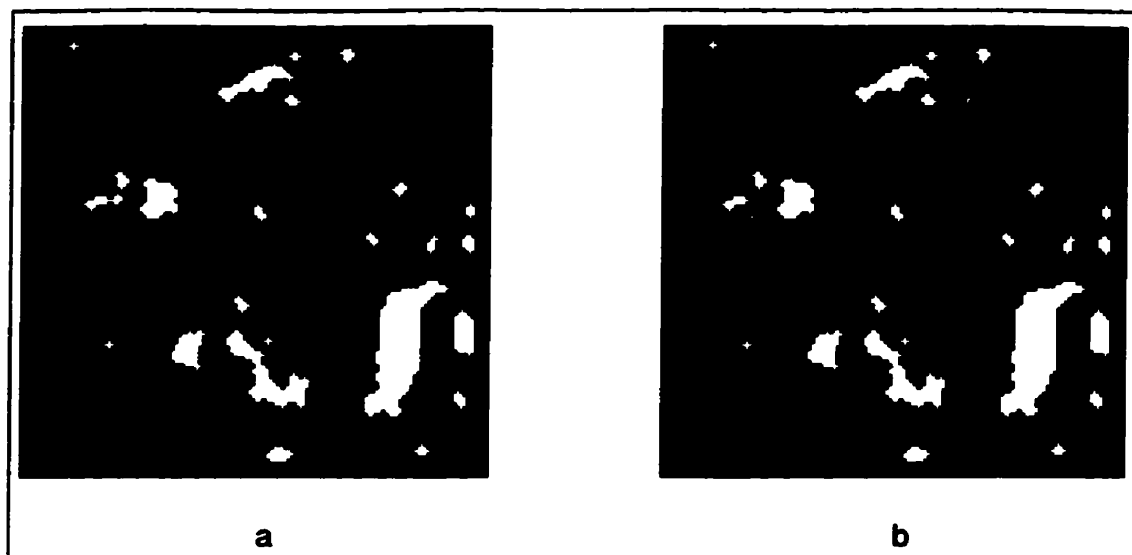


Figure 4.8: *a*: Same as Figure 3.31a. *b*: *a* modified using one region separation operation using UICP. The single line of separation has also been drawn for clarity.

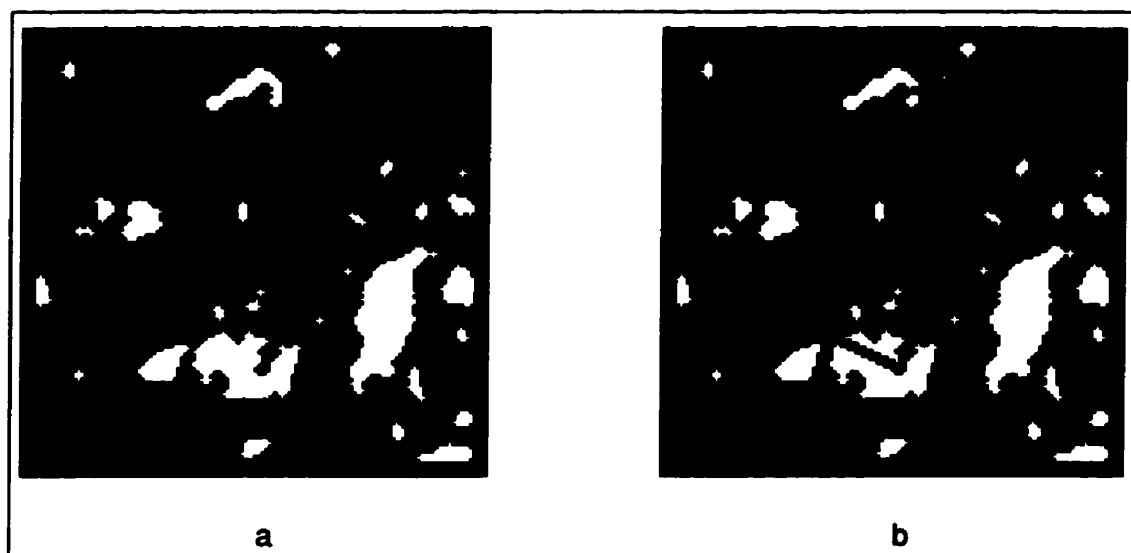


Figure 4.9: *a*: Same as Figure 3.31b. *b*: *a* modified using three region separation operations using UICP. The three lines of separations have also been drawn for clarity.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
*1	15.34	61.73	16.69	58.33	16.68	58.78	3.65	0.44
2	7.63	87.12	5.92	83.01	5.65	83.00	4.46	0.27
3	48.64	36.15	54.24	31.36	54.42	31.57	7.37	0.28
*4	43.28	26.06	50.54	21.50	51.17	21.53	8.57	0.63
5	52.53	63.40	53.01	58.95	52.00	59.00	4.47	1.01
6	45.80	102.20	39.37	98.44	39.50	98.50	7.45	0.15
7	60.00	94.50	54.07	90.83	54.00	90.50	6.98	0.34
8	61.50	111.30	52.04	108.20	51.72	108.60	9.95	0.52
9	52.00	122.00	41.34	118.80	41.00	120.00	11.13	1.25
10	61.28	121.60	49.74	118.90	50.08	119.60	11.86	0.77
11	90.38	102.50	80.56	99.78	81.78	98.86	10.17	1.52
12	86.72	120.50	72.83	118.90	73.57	119.20	13.98	0.80
*13	98.96	66.01	97.96	60.52	98.10	61.07	5.58	0.58
14	121.20	69.24	118.80	63.59	119.20	62.69	6.15	1.00
*15	20.64	71.79	20.47	68.03	20.38	67.62	3.76	0.43
*16	78.50	58.75	79.60	53.34	80.77	52.77	5.52	1.30
*17	91.28	44.54	95.76	37.67	95.61	37.54	8.21	0.20
*18	49.57	19.61	58.27	14.46	57.53	16.27	10.11	1.95

* Regions changed or added because of separation of concatenated regions as compared to table (3.3).

Table 4.4: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

4.3.5 Slice 12

Figure 4.10 shows 18 regions marked and numbered in slice 12. There are three extra regions here as compared to earlier (compare Figure 3.32 and Figure 4.10). Three region separation operations were performed on the left-hand side image (refer Figure 4.11). All the extra regions were created by region separation operations. Region 3 was earlier one component on the left-hand side and three fragments on the right. Two region separations on the left were performed so that each of the fragments on the right could have a matching region on the left. Thus regions 16 and 17 were created. Region 2 had two fragments on the right which became regions 2 and 18, with an appropriate region separation operation on the left-hand side. The centroids and D and E values are tabulated in Table (4.5). As before, regions changed or added because of the region separation operations have been marked by an asterisk in Table (4.5).

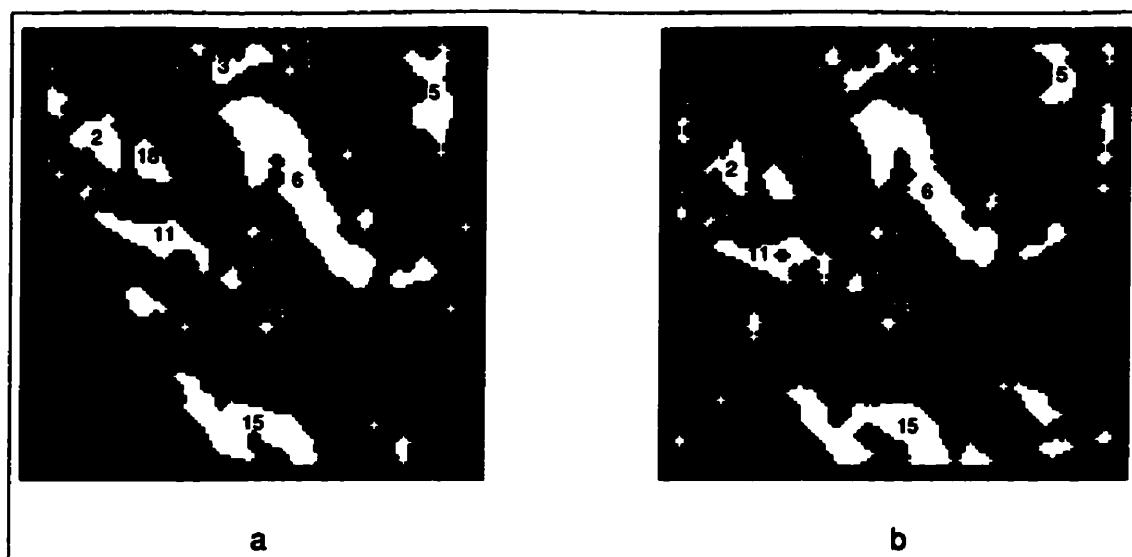


Figure 4.10: *a* and *b*: 18 regions obtained using UICP, shown for slice 12 of the starbyte-transformed images (protocol 3) of Figure 2.2b and Figure 2.3b. *a* was obtained by modifying Figure 3.32a using UICP, while *b* is the same as Figure 3.32b.

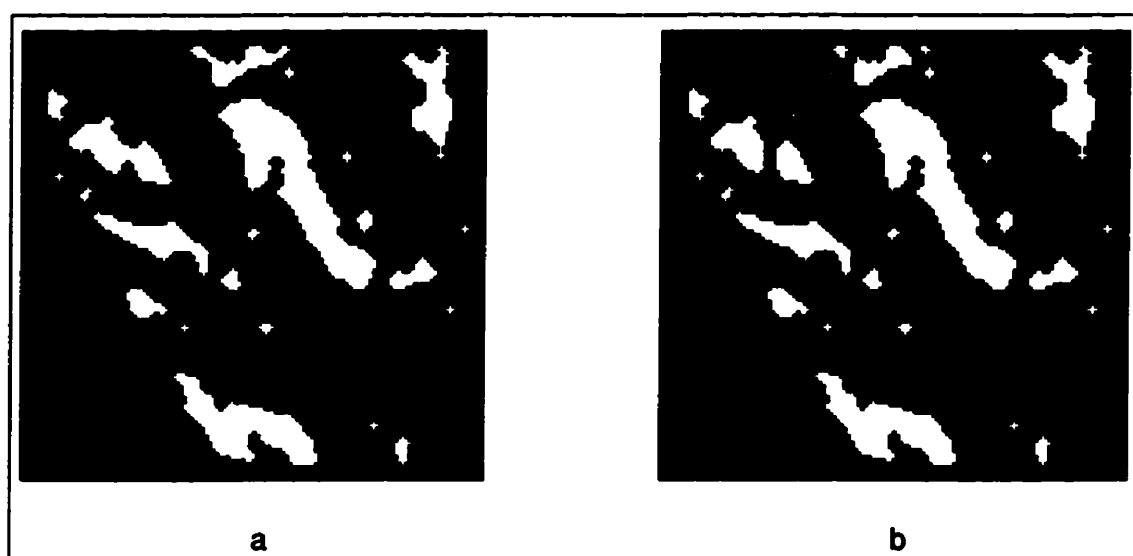


Figure 4.11: *a*: Same as Figure 3.32a. *b*: *a* modified using three region separation operations using UICP. The three lines of separation have also been drawn for clarity.

Region Number	x	y	x^*	y^*	\hat{x}	\hat{y}	D (pixels)	E (pixels)
1	20.44	9.03	29.29	6.56	28.50	5.30	9.19	1.49
*2	31.62	21.32	39.21	17.68	39.63	18.81	8.41	1.21
*3	9.43	59.23	11.18	56.09	11.12	56.68	3.59	0.59
4	11.50	73.00	11.44	69.35	11.00	68.50	3.65	0.96
5	18.78	112.50	12.91	107.80	12.43	11.00	7.49	0.52
6	43.34	74.89	41.94	70.80	41.36	70.70	4.32	0.59
7	53.75	94.21	48.27	90.45	48.57	90.50	6.65	0.31
8	69.27	107.60	59.94	104.60	60.17	104.80	9.79	0.29
9	41.00	10.00	50.97	5.51	52.00	5.00	10.94	1.15
10	46.00	17.36	54.94	12.49	54.50	12.50	10.19	0.44
11	58.69	37.06	64.34	31.66	64.48	30.88	7.81	0.79
12	57.30	63.30	57.72	58.72	57.50	58.50	4.60	0.31
13	70.44	57.63	71.87	52.44	72.05	52.59	5.39	0.24
14	84.17	67.00	83.24	61.92	83.17	62.00	5.16	0.11
15	112.00	61.67	112.00	55.44	113.30	57.84	6.23	2.72
*16	5.31	48.08	8.47	45.53	7.90	45.80	4.06	0.63
*17	4.83	71.83	5.12	68.33	5.00	68.00	3.51	0.35
*18	37.66	35.11	43.18	31.00	43.81	31.19	6.88	0.65

* Regions changed or added because of separation of concatenated regions as compared to table (3.4).

Table 4.5: *Column 1:* Region number. *Column 2 and 3:* x and y coordinates of centroids of the regions in the target image. *Columns 4 and 5:* x and y coordinates of corresponding location for the entries in columns 2 and 3 as determined by equation (2.1). *Column 6 and 7:* x and y coordinates of centroids of the regions in the reference image. *Column 8:* D , the Euclidean distance between (x, y) and (x^*, y^*) (Distortion Magnitude). *Column 9:* E , the Euclidean distance between (x^*, y^*) and (\hat{x}, \hat{y}) (Registration Error).

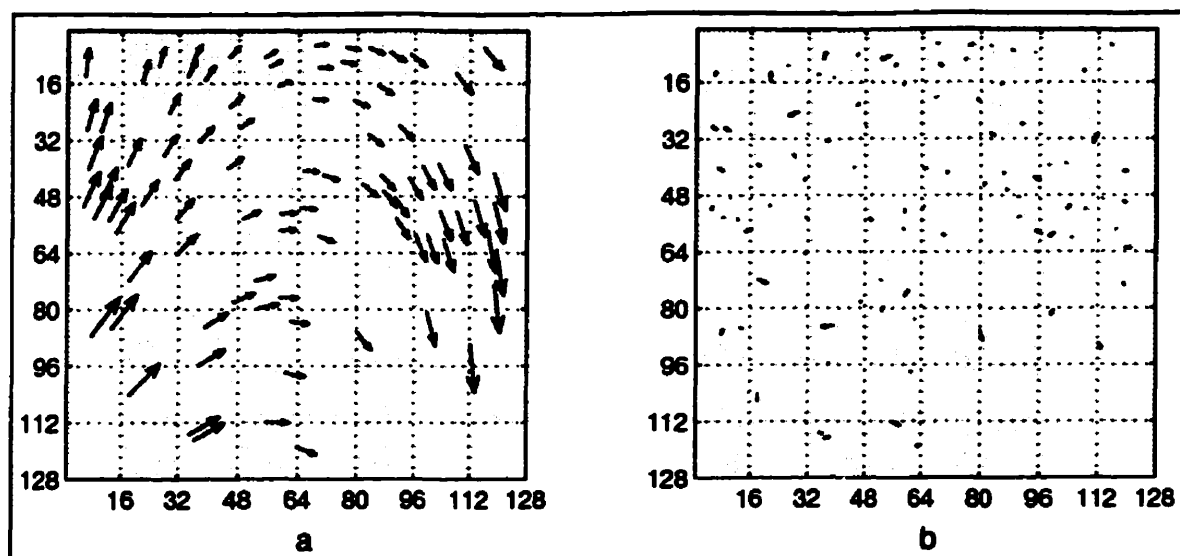


Figure 4.12: *a* and *b*: **D** and **E** for the case where the two original images (Figure 2.2b and Figure 2.3b) are related by equation (2.1), and the starbyte transformation was used with protocol 3.

Figure 4.12 shows the control points from these various slices pooled together as an arrow diagram. The tails of the arrows in both Figure 4.12a and b show (x^*, y^*) , while the heads of the arrows in Figure 4.12a show (x, y) and the heads of the arrows in Figure 4.12b show (\hat{x}, \hat{y}) . Thus Figure 4.12a is a vector diagram of **D** (D is the magnitude of **D**), while Figure 4.12b is a vector diagram of **E** (E is the magnitude of **E**). It is clear from Figure 4.12, that the control points obtained using UICP are accurate and that E is very low. These control points will be used to obtain the unwarped image from the reference image. Numerical and visual comparisons between the obtained unwarped images and the target images for the automatic and manual cases for all distortions will be performed in Chapter 6.

The final registration achieved is shown in Figure 4.13 as a grid diagram. Figure 4.13a shows the actual distortion between the original images, while Figure 4.13b shows the registration achieved after using the control points obtained using UICP.

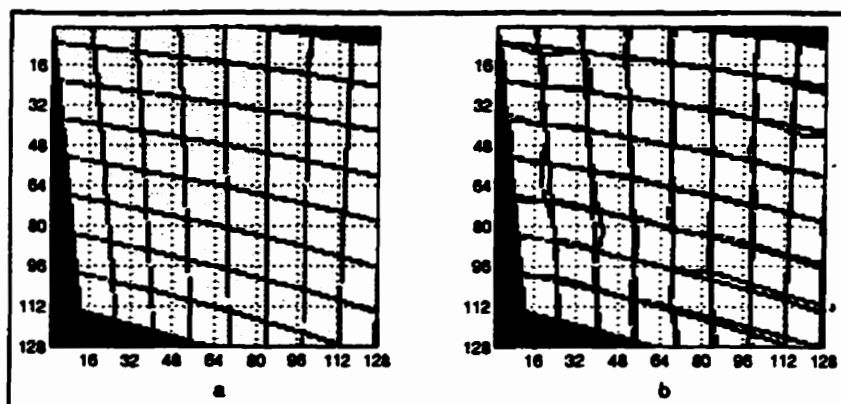


Figure 4.13: *a*: Bilinear distortion shown as a grid diagram (equation (2.1)). *b*: Registration achieved after using UICP.

The final grid images (actual and through UICP) were obtained in the following way: Starting with an image which was a rectangular grid, and using the control points obtained using UICP, and using thin plate splines for unwarping, the final image (which represented the distortion of the grid) was calculated. Since this would be a gray level image, because of interpolation (in spite of the fact that the original (the rectangular grid) was black and white), a threshold was performed at half the density range to obtain a black and white final grid. The grid diagram showing the actual distortion between the original images (Figure 4.13), was obtained in the same way, by actually applying the known transformation (equation (2.1)) to the rectangular grid and performing a thresholding operation to obtain a binary final grid. The grid showing the actual distortion and the grid showing the distortion obtained using the control points from UICP were superimposed and this is what is shown in Figure 4.13b. Grid diagrams will be shown for all the examples, for both manual and automatic cases. Because of the thresholding operation performed, some of the grid lines may appear rugged and this should not be construed as poor reconstruction. The grid diagrams are merely to illustrate the extent of registration,

and the extent to which the obtained grid differs from the actual or desired grid.

4.4 Bilinear distortion, protocol 1

We shall now present the results obtained using UICP for the cleaned-up slices Figures 3.10a and b. These were also the starbyte images of Figure 2.2b and Figure 2.3b. However protocol 1 was used. Hence the original pair of images was the same and the only difference was the protocol used to generate the starbyte images. This experiment was done merely to study the effect of using a different protocol and to see whether control points could be extracted as effectively using UICP as with the earlier case. Here we shall present the marked and numbered regions for the prominent slices namely slices 3, 5, 10 and 12. These were the slices for which more than 10 regions each could be marked. Region separations were performed for slices 3, 5, 10 and 12 on the left-hand side and slices 3, 5, 7, 10 and 12 on the right. Separate figures have not been provided to illustrate the region separations here. However, since the regions are marked and numbered on the altered slices, the region separation operations can be inferred by comparing a slice with its unaltered counterpart in Figure 3.12 and Figure 3.11. A total of 74 control points was obtained in this case as compared to 89 for protocol 3.

Figure 4.14 shows 21 regions marked and numbered in the 3rd slice. Region 5 is the only region with fragments on the right-hand side. Some regions, for e.g. 13 or 16, could have very distinguishing features and it is possible to subdivide them into smaller regions in order to obtain more control points. Figure 4.15 shows 12 regions in the 5th slice. Figure 4.16 shows 24 regions on the 10th slice. As in Figure 4.14, here too there are many regions with clear features (for e.g. 1, 5, 11 etc.) which could have been subdivided further using the region separation feature of UICP to obtain more

control points. Figure 4.17 shows the 17 regions in slice 12. As before regions like 2, 6, 7, 10 and 17 could have been subdivided further using UICP. Figure 4.18a shows **D** and Figure 4.18b shows **E** as before, with all 74 control points pooled together. As before it is clear that the control points obtained using UICP are good because the arrows in the **D** plot have been reduced to points in the **E** plot for all points except one, or in other words, E , the magnitude of **E** is low. An observation of Figure 4.12 and Figure 4.18 shows that the distribution of control points obtained is almost the same for both. This shows that either protocol gives equally good results. The final registration achieved is shown in Figure 4.19 as a grid diagram. Figure 4.19a shows the actual distortion between the original images, while Figure 4.19b shows the registration achieved after using the control points obtained using UICP. Except for one region on the boundary where there is misregistration corresponding to the one large error on the **D-E** plot, there is good registration everywhere else. The six figures showing slices 3, 5, 10 and 12, Figure 4.18 and Figure 4.19 are shown on the next three pages. The text continues after that.

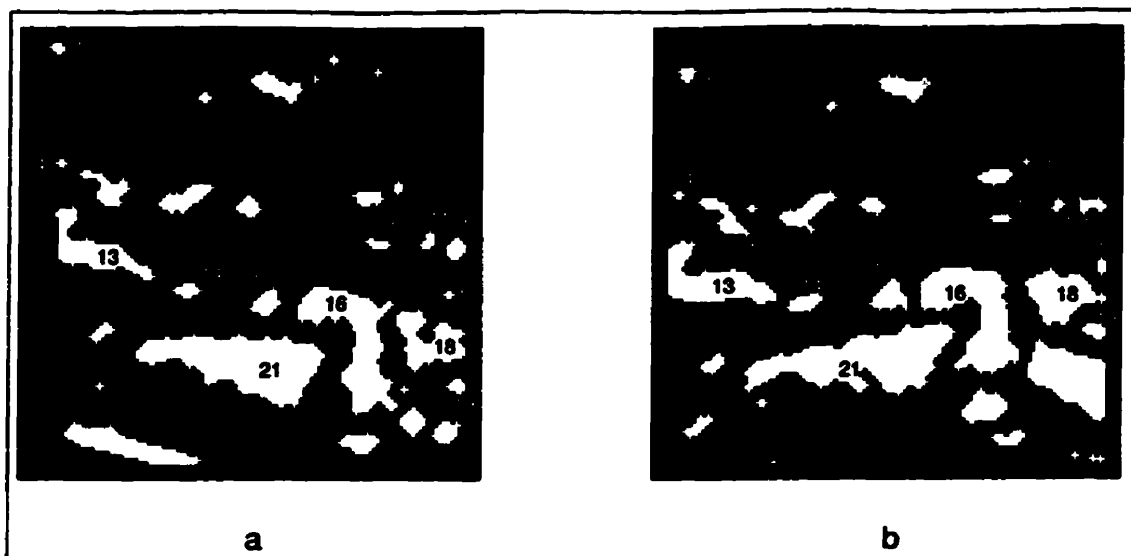


Figure 4.14: *a* and *b*: 21 regions obtained using UICP, shown for slice 3 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12d and Figure 3.11d using UICP.

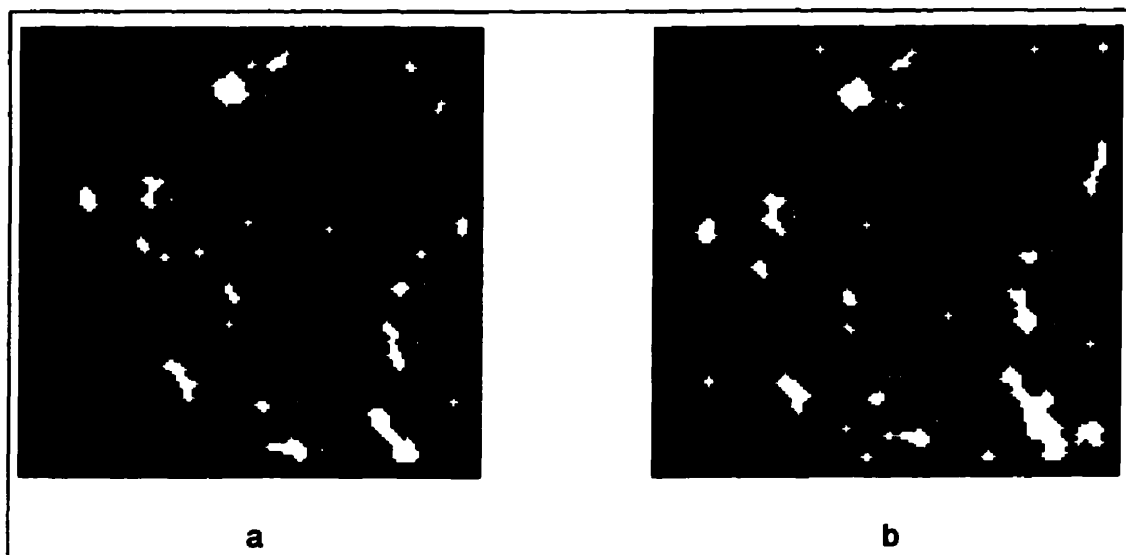


Figure 4.15: *a* and *b*: 12 regions obtained using UICP, shown for slice 5 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12f and Figure 3.11f using UICP.

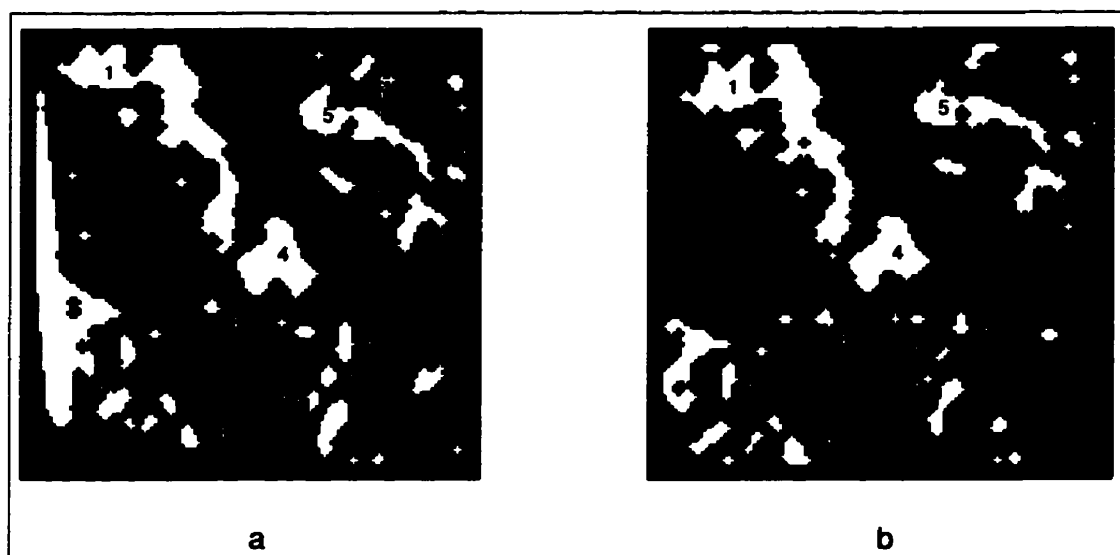


Figure 4.16: *a* and *b*: 24 regions obtained using UICP, shown for slice 10 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12k and Figure 3.11k using UICP.

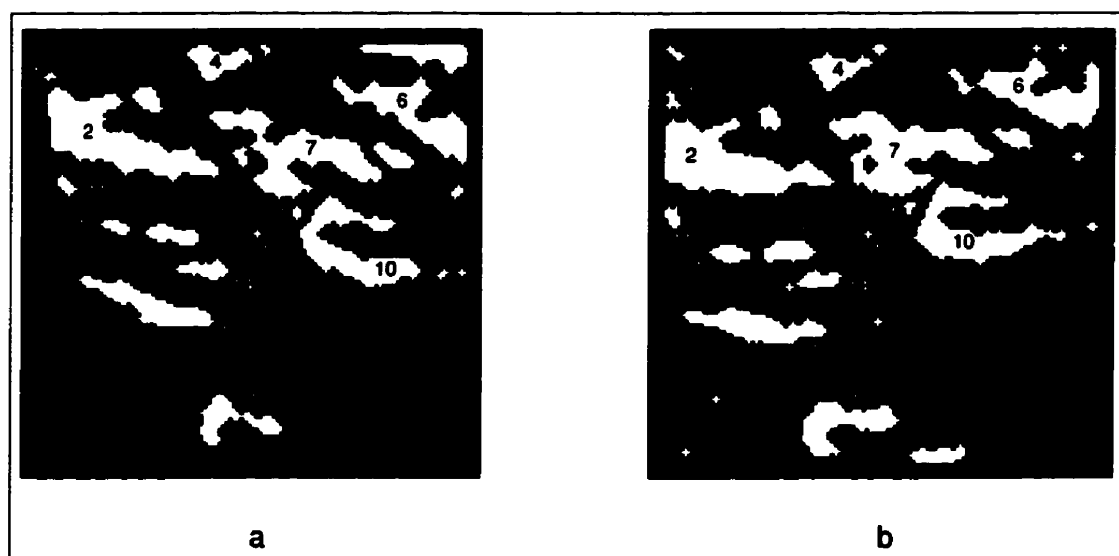


Figure 4.17: *a* and *b*: 17 regions obtained using UICP, shown for slice 10 of the starbyte-transformed images (protocol 1) of Figure 2.2b and Figure 2.3b. Both slices were obtained by modifying Figure 3.12m and Figure 3.11m using UICP.

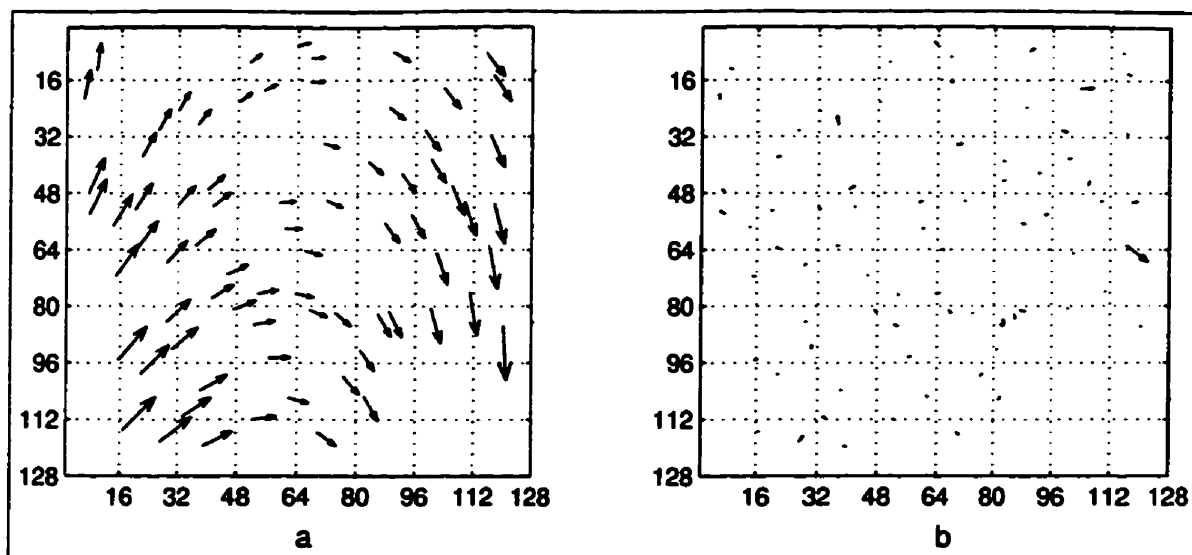


Figure 4.18: *a* and *b*: **D** and **E** for the case where the two original images (Figure 2.2b and Figure 2.3b) are related by equation (2.1), and the starbyte transformation was used with protocol 1.

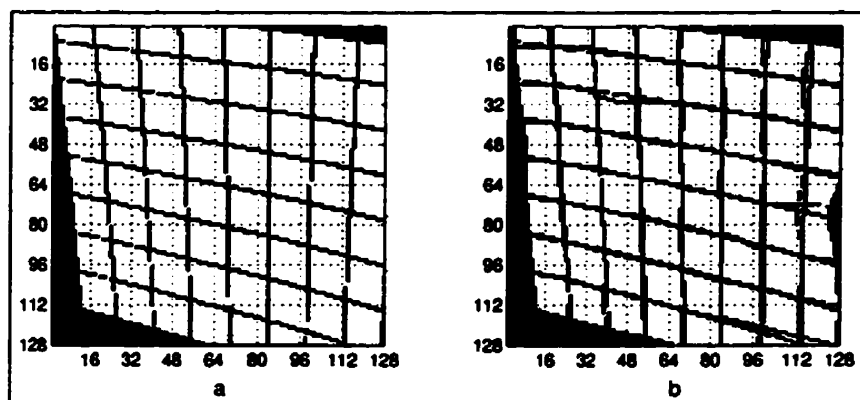


Figure 4.19: *a*: Bilinear distortion shown as a grid diagram (equation (2.1)). *b*: Registration achieved after using UICP with protocol 1.

We shall next discuss the performance of UICP for the other five distortions (translation, rotation, scaling, thin plate spline (TPS) and sinusoidal) discussed in the previous two chapters. Even for these examples, we shall show the marked and numbered regions for the prominent slices (slices where 10 or more identifiable regions were present) and we shall also show vector diagrams of **D** and **E** where all control points obtained using UICP will be pooled together.

4.5 Translation

In this case, the original images were Figures 2.15a and b, where Figure 2.15b was obtained by translating Figure 2.15a by 8 pixels leftwards and 10 pixels upwards. Figure 2.15a was the same as Figure 2.2b. The starbyte images for these were shown in the last chapter (Figures 3.13a and b). The cleaned-up individual density slices using an OC operation for Figures 3.13a and b were also presented in the previous chapter (Figure 3.4 and Figure 3.14). A total of 130 control points was obtained. There were 8 slices with 10 or more matching regions (slices 1, 3, 5, 6, 9, 11, 12 and 14). Only slices 3, 6, 9 and 12 have been presented with regions marked and numbered because these contain the larger regions. Slices containing 10 or more matchable regions contributed totally to 114 control points while the remaining slices had a total of 16 matching regions. No region separation operations had to be performed for any slices. Hence none of the slices were altered. A study of the 4 slices presented shows that almost all the regions were intact on the left-hand side (compared to the right-hand side), except that the translation caused them to be shifted with respect to the untranslated slice on the right.

Figure 4.20 shows 17 regions in slice 3. Figure 4.21 shows 17 regions in slice 6. Figure 4.22 and Figure 4.23 show 20 and 16 regions in slices 9 and 12 respectively. As

before, since many regions in each of these slices have very distinguishing features it would have been possible, using UICP's region separation feature to subdivide many of these regions to obtain more control points. The 130 points have been pooled together to obtain the **D** and **E** vector plots (Refer Figures 4.24a and b). It is clear from Figure 4.24 that E , the magnitude of **E** is almost zero for most points. The final registration achieved is shown in Figure 4.25 as a grid diagram. Figure 4.25a shows the actual distortion between the original images (translation by 8 pixels upwards and 10 pixels leftwards), while Figure 4.25b shows the registration achieved after using the control points obtained using UICP, and TPS-unwarping.

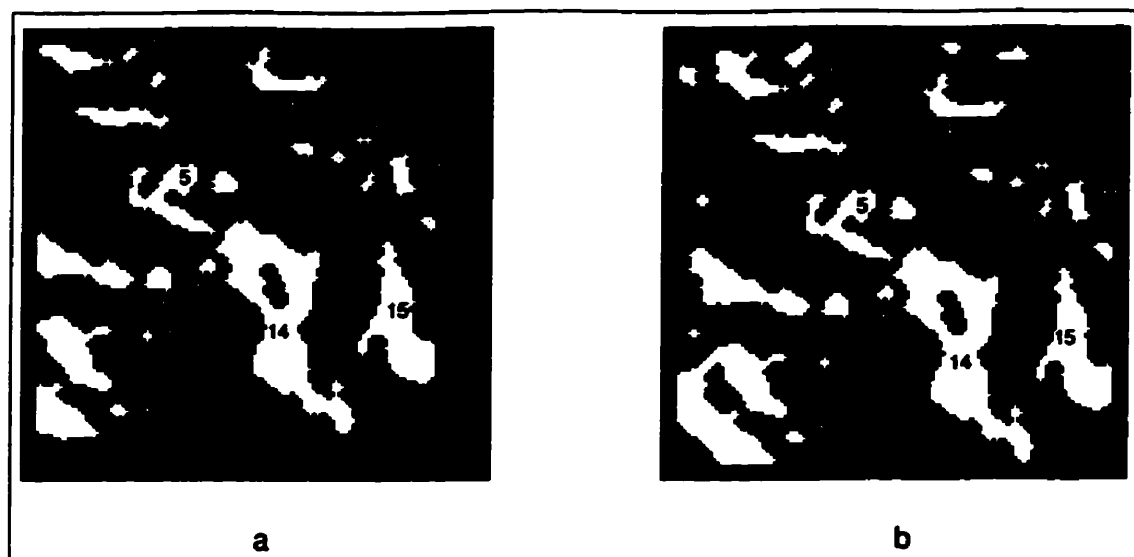


Figure 4.20: *a* and *b*: Figure 3.14d and Figure 3.4d with 17 regions marked.

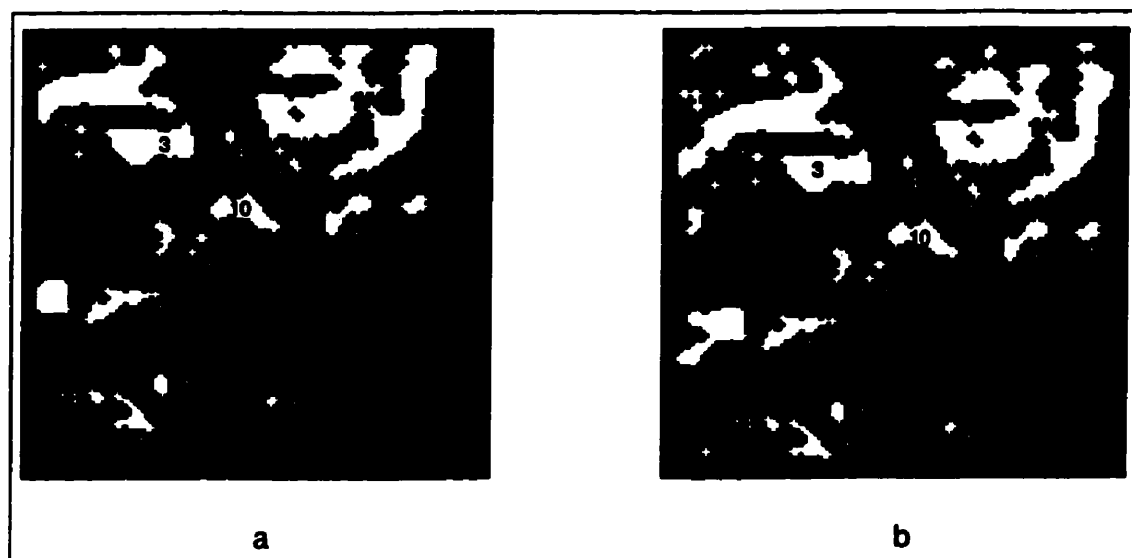


Figure 4.21: *a* and *b*: Figure 3.14g and Figure 3.4g with 17 regions marked.

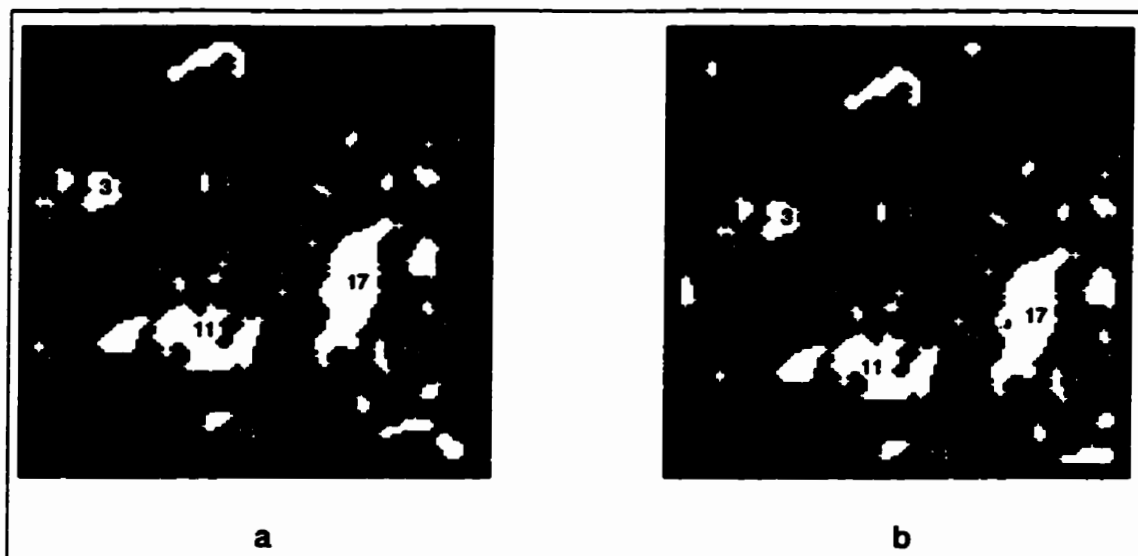


Figure 4.22: *a* and *b*: Figure 3.14j and Figure 3.4j with 20 regions marked.

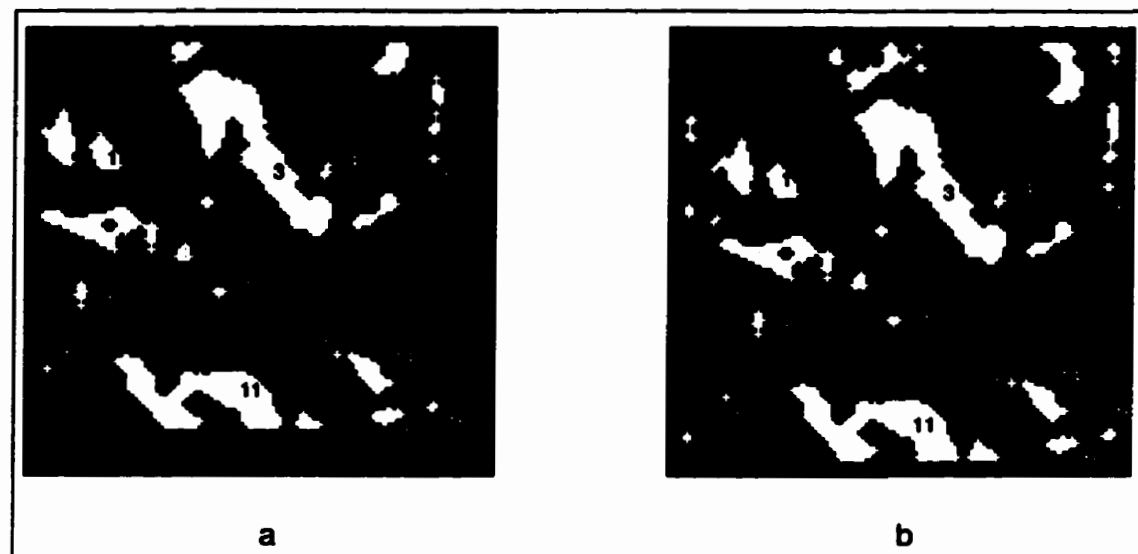


Figure 4.23: *a* and *b*: Figure 3.14m and Figure 3.4m with 16 regions marked.

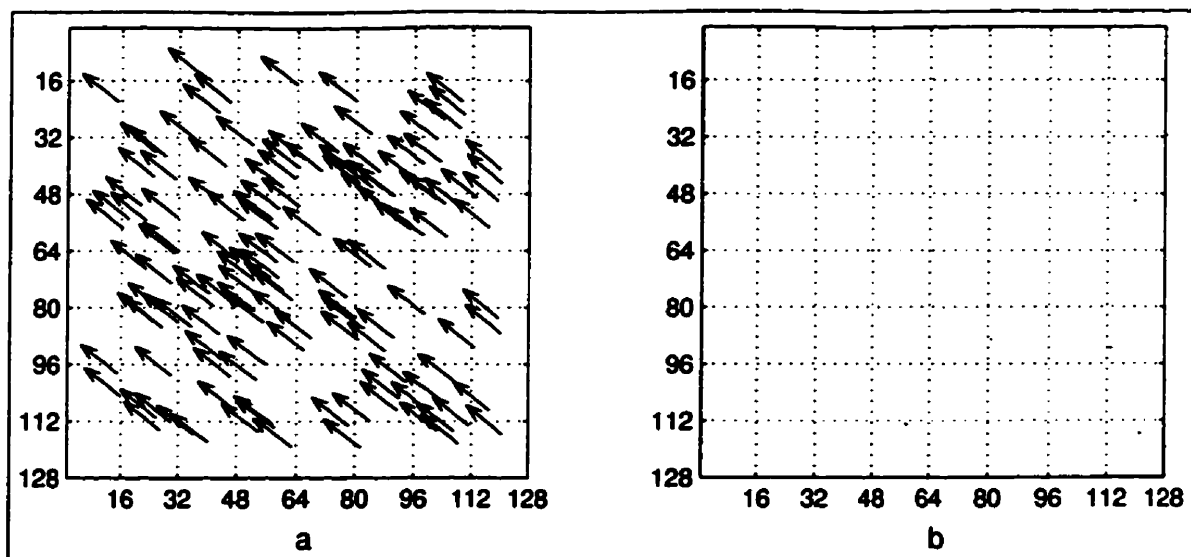


Figure 4.24: *a* and *b*: **D** and **E** for the original images (Figures 2.15a and b) related by translation (8 pixels upwards and 10 pixels leftwards).

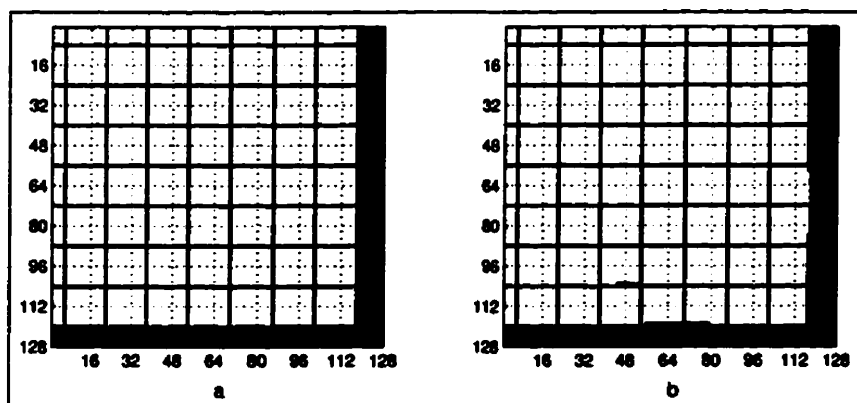


Figure 4.25: *a*: Translation (8 pixels upwards and 10 pixels leftwards) shown as a grid diagram. *b*: Registration achieved after using UICP.

4.6 Rotation

In this case, the original images were Figures 2.15a and c, where Figure 2.15c was obtained by rotating Figure 2.15a by 10 deg counter-clockwise. Figure 2.15a was the same as Figure 2.2b. The starbyte images for these were shown in the last chapter (Figures 3.13a and c). The cleaned-up individual density slices using an OC operation for Figures 3.13a and c were also presented in the previous chapter (Figure 3.4 and Figure 3.15). In this case, a total of 60 points was obtained using UICP. Four slices (slice 3, 6, 9 and 12) had 10 or more matching regions. These slices have been presented below. Region separations had to be performed for both left and right-hand sides for all these slices. The region separation operations can be inferred by comparing the altered slices with their unaltered counterparts in Figure 3.15 and Figure 3.4.

Figure 4.26 shows 12 regions marked in slice 3. While for many of the regions it is easy to identify matches, in some cases, the rotation operation has severely distorted the region. For e.g., regions 10 and 11 on slice 3 in the translation example (refer Figure 4.20), has been severely changed on the left-hand side in Figure 4.26. Also region 2 earlier for the same slice has been changed now. Hence we obtained 3 control points less for slice 3 for rotation as compared to translation. Figure 4.27 shows 12 regions marked on slice 6. Some regions have been fragmented due to rotation (for e.g. regions 11 and 12). Region concatenation has also taken place (region 12 at the bottom). Figure 4.28 shows 20 regions on slice 9. Region 12 on the left-hand side has got fragmented and region 14 consists of two concatenated portions, which are separate on the right-hand side. Deliberate region separation operations were performed to separate regions 9 and 10. Figure 4.29 shows 16 regions on slice 12. Again region fragmentation and region concatenation have taken place due to

rotation. Figures 4.30a and b show D and E as before. Again it is clear that E is low and in most cases, less than a pixel. The final registration achieved is shown in Figure 4.31 as a grid diagram. Figure 4.31a shows the actual distortion between the original images (10 deg counter-clockwise rotation), while Figure 4.31b shows the registration achieved after using the control points obtained using UICP, and TPS-unwarping.

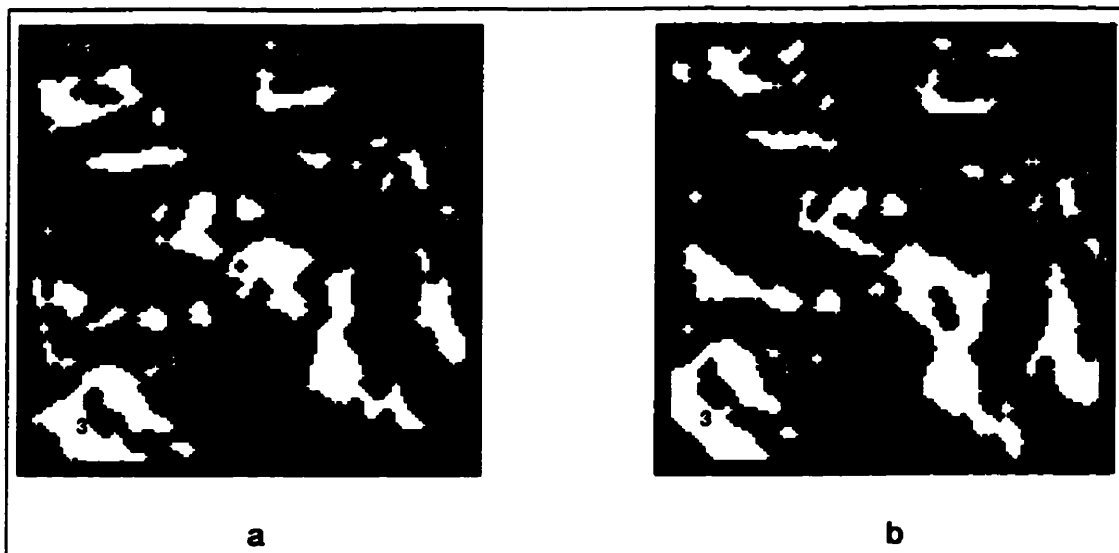


Figure 4.26: *a* and *b*: Figure 3.15d and Figure 3.4d with 12 regions marked after being modified by UICP.

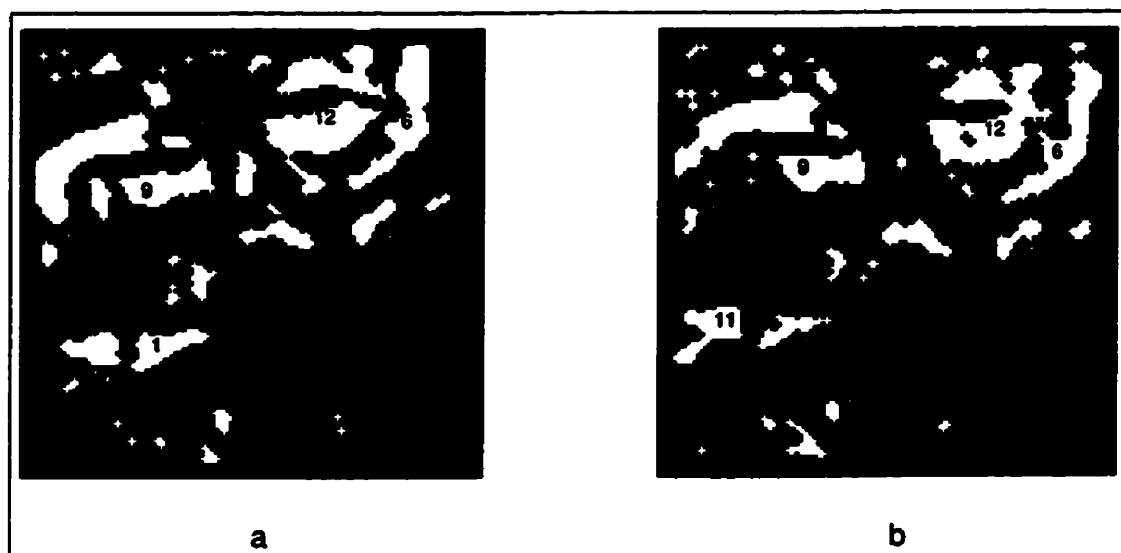


Figure 4.27: *a* and *b*: Figure 3.15g and Figure 3.4g with 12 regions marked after being modified by UICP.

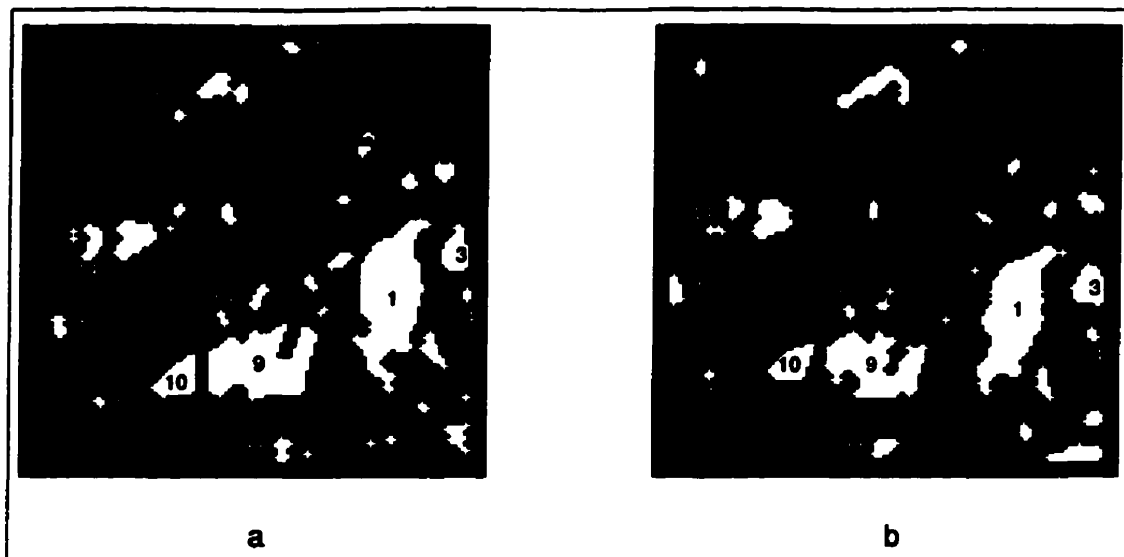


Figure 4.28: *a* and *b*: Figure 3.15j and Figure 3.4j with 20 regions marked after being modified by UICP.

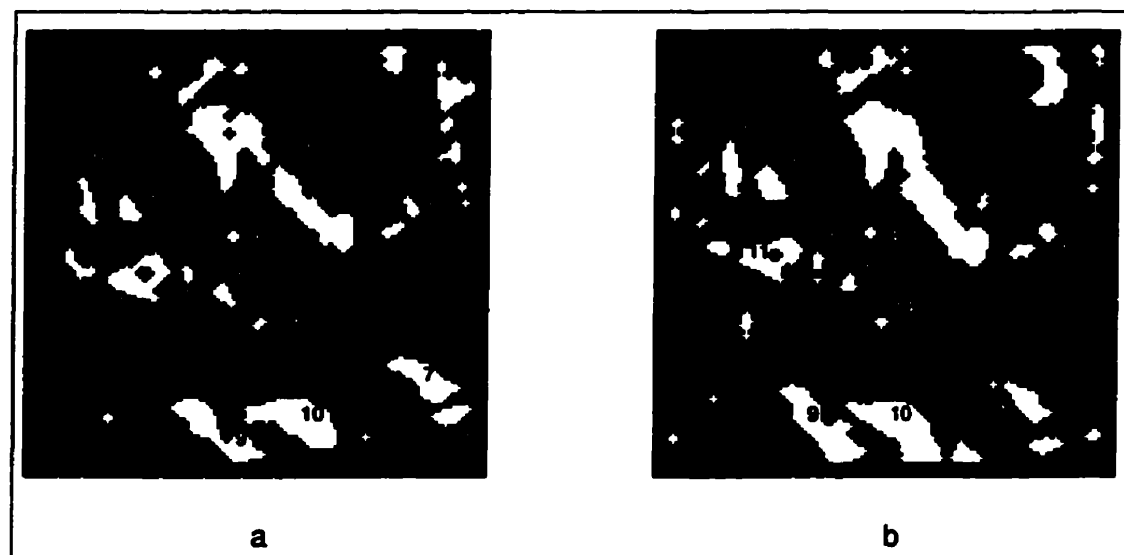


Figure 4.29: *a* and *b*: Figure 3.15m and Figure 3.4m with 16 regions marked after being modified by UICP.

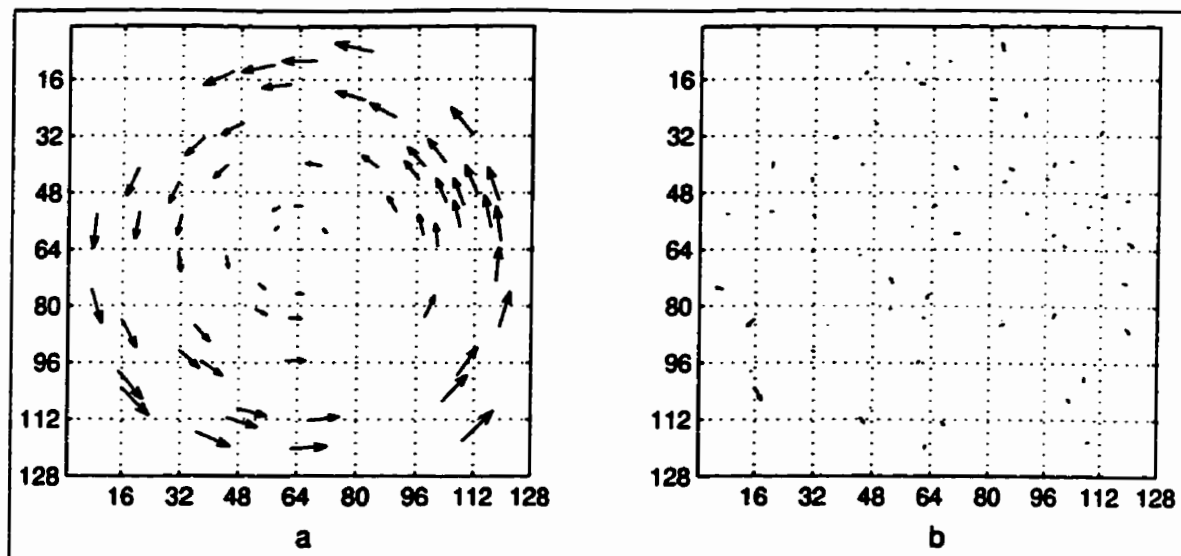


Figure 4.30: *a* and *b*: **D** and **E** for the original images (Figures 2.15a and c) related by rotation (10 deg counter-clockwise).

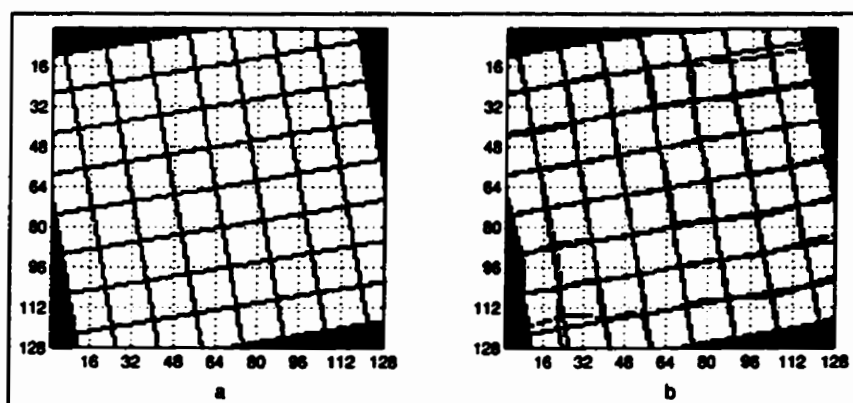


Figure 4.31: *a*: Rotation (10 deg counter-clockwise) shown as a grid diagram. *b*: Registration achieved after using UICP.

4.7 Scaling

In this case, the original images were Figures 2.15a and d, where Figure 2.15d was obtained by enlarging Figure 2.15a by 10 %. Figure 2.15a was the same as Figure 2.2b. The starbyte images for these were shown in the last chapter (Figures 3.13a and d). The cleaned-up individual density slices using an OC operation for Figures 3.13a and d were also presented in the previous chapter (Figure 3.4 and Figure 3.16). Slices 6 on both sides, slice 9 on the right side and slice 12 on the left were altered using UICP.

A total of 95 points was obtained using UICP. In this case there were 5 slices (slices 1, 3, 6, 9, 12) which had 10 or more matchable regions. The other slices contributed 12 points totally. Figure 4.32 shows 10 regions marked on slice 1. We can observe that while regions are still easy to match in both slices, regions have been enlarged on the left side due to scaling. Figure 4.33 shows 22 regions on slice 3. Here also the enlargement effect is obvious. In fact, enlargement has caused parts of some regions to be pushed out of the frame of the image (regions 1, 21 etc). Otherwise, identification of matching regions is straightforward. Figure 4.34 shows 22 regions on slice 6 while Figure 4.35 and Figure 4.36 show 18 and 11 regions each on slices 9 and 12 respectively. Figures 4.37a and b show *D* and *E* as before. *E* is low for all points. The final registration achieved is shown in Figure 4.38 as a grid diagram. Figure 4.38a shows the actual distortion between the original images (10 % enlargement), while Figure 4.38b shows the registration achieved after using the control points obtained using UICP, and TPS-unwarping.

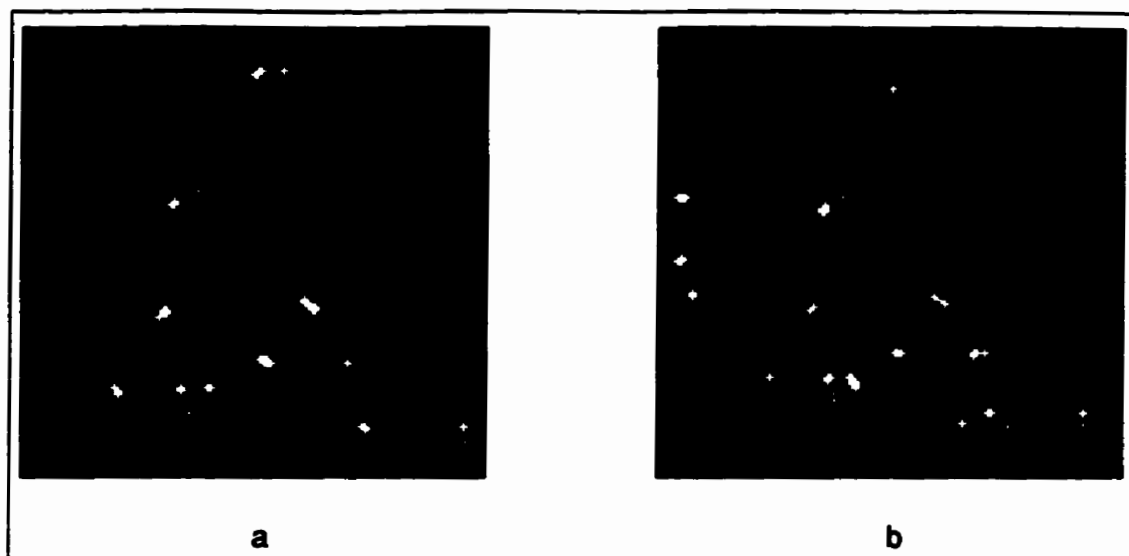


Figure 4.32: *a* and *b*: Figure 3.16b and Figure 3.4b with 10 regions marked.

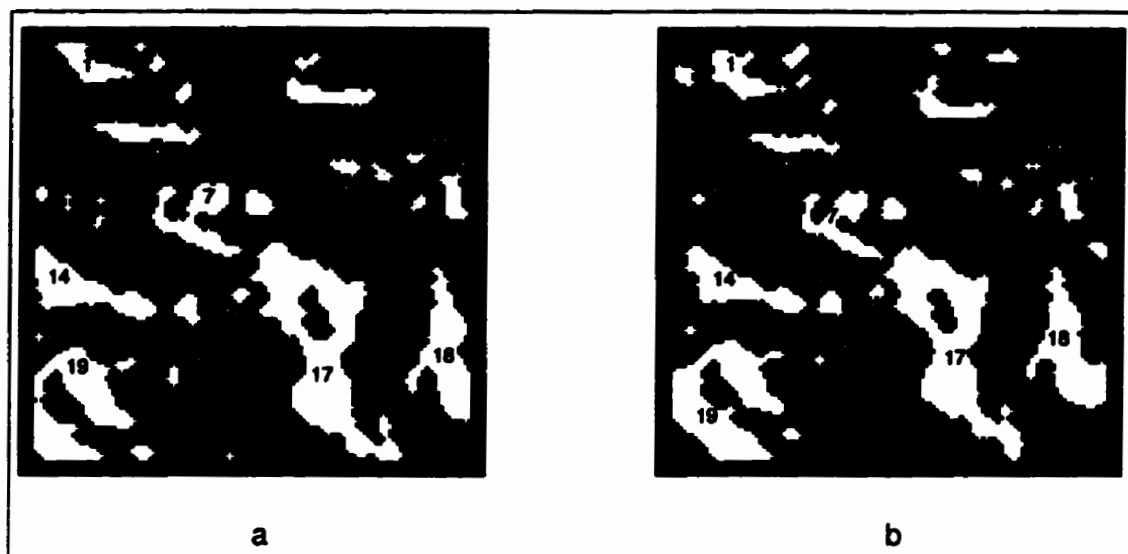


Figure 4.33: *a* and *b*: Figure 3.16d and Figure 3.4d with 24 regions marked.

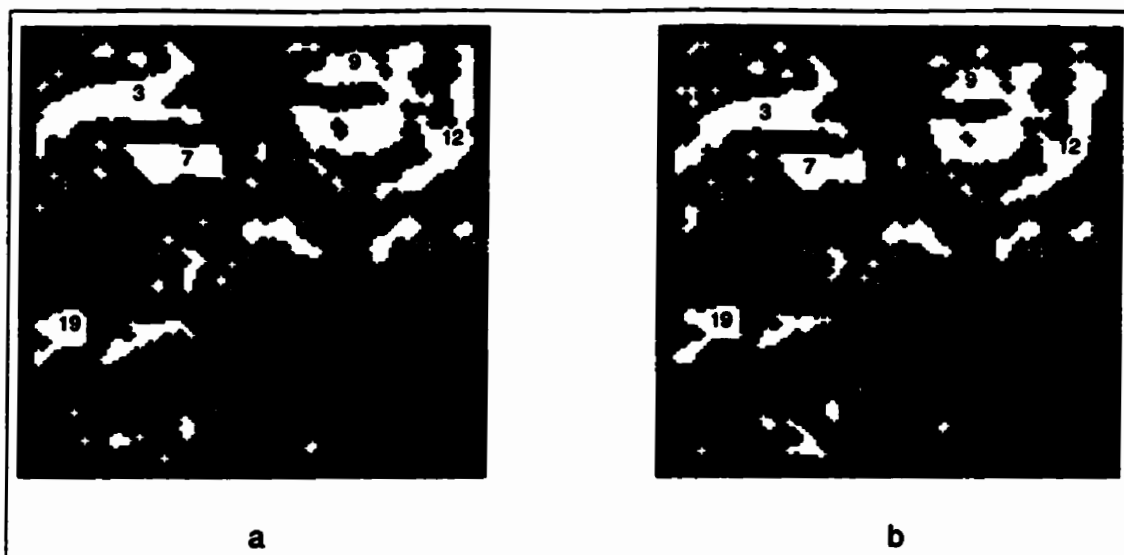


Figure 4.34: *a* and *b*: Figure 3.16g and Figure 3.4g with 22 regions marked after being modified by UICP.

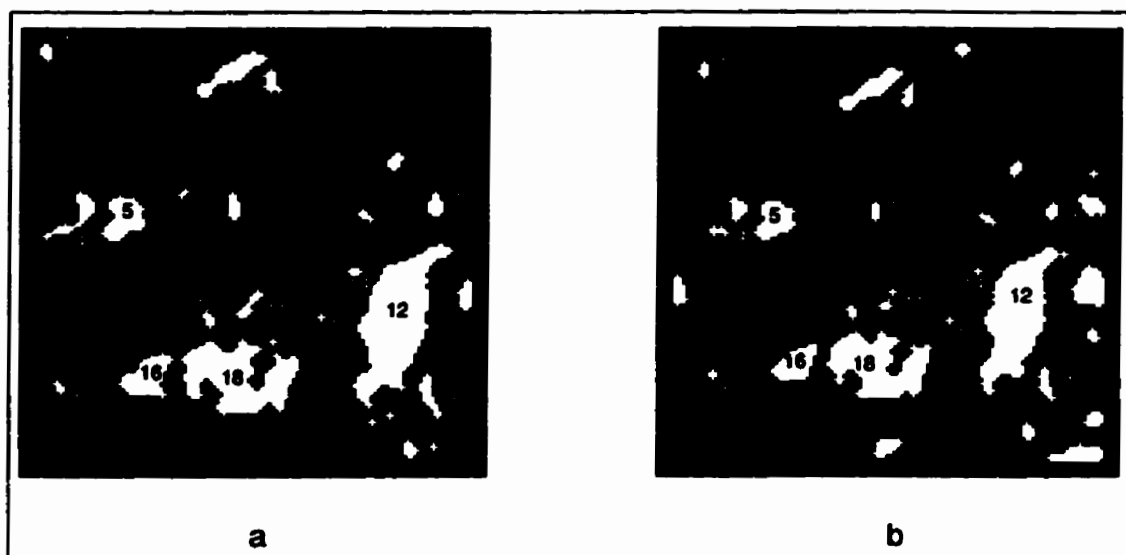


Figure 4.35: *a* and *b*: Figure 3.16j and Figure 3.4j (after modification with UICP) with 18 regions marked.

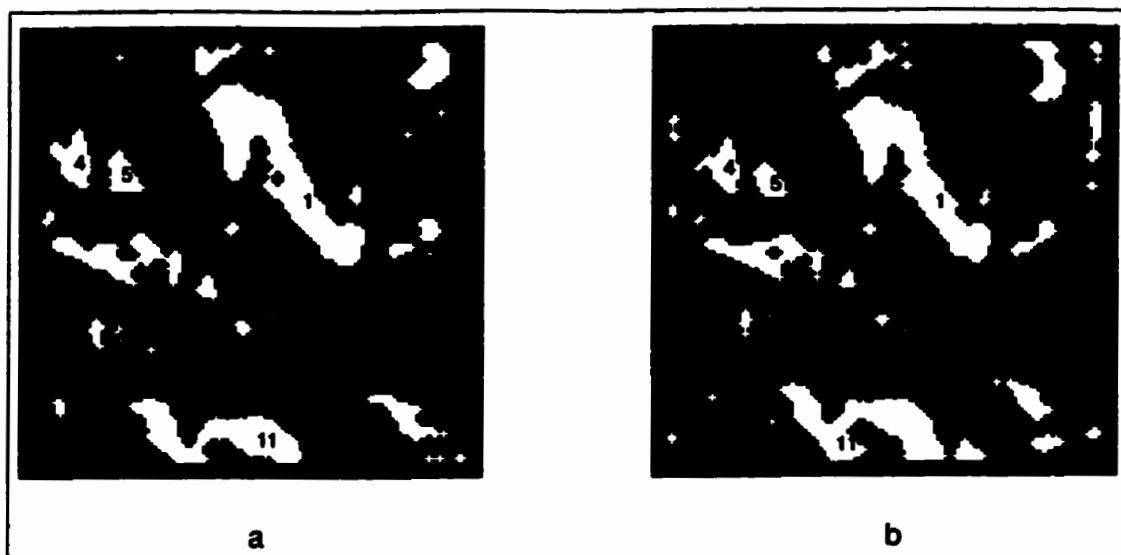


Figure 4.36: *a* and *b*: Figure 3.16m (after modification with UICP) and Figure 3.4m with 11 regions marked.

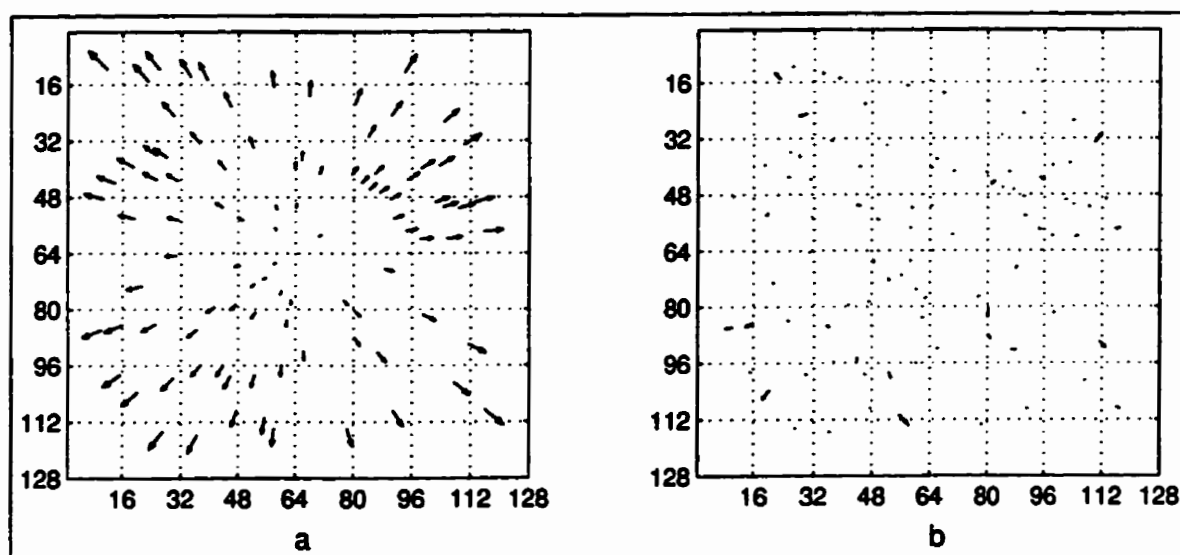


Figure 4.37: *a* and *b*: *D* and *E* for the original images (Figures 2.15a and c) related by a 10 % enlargement.

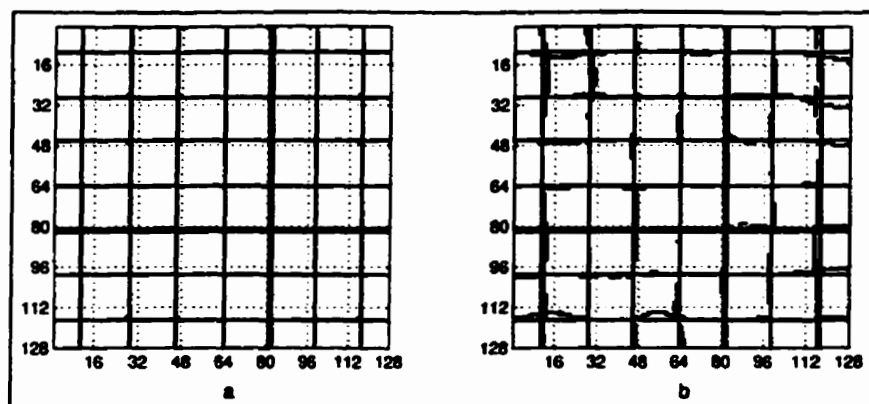


Figure 4.38: *a*: Scaling (10 % enlargement) shown as a grid diagram. *b*: Registration achieved after using UICP.

4.8 TPS distortion

Here, the original images were Figure 2.24a and its TPS-distorted counterpart using the TPS distortion shown in Figure 2.23e. The starbyte images obtained using protocol 1 are shown in the previous chapter (Figure 3.17a and b) and the cleaned-up density slices using an OC operation are also shown (Figure 3.18 and Figure 3.19). In this case, slices 5 and 10 were altered on both sides using UICP while slice 8 was altered on the right and slice 12 on the right side. There were 7 slices with 10 or more identifiable regions (slices 2, 3, 5, 8, 10, 12 and 13). A total of 143 control points was obtained using UICP. The main slices contributed to 120 points while the remaining contributed to 23 points. Figures 4.39, 4.40, 4.41, 4.42, 4.43, 4.44 and 4.45 show 10, 21, 10, 14, 31, 24 and 10 numbered regions in slices 2, 3, 5, 8, 10, 12 and 13 respectively. Fragmented regions are indicated as usual by drawing a boundary enclosing all the fragments. Figure 4.46a and b show **D** and **E** respectively, with all 143 points pooled together. While *D*, the magnitude of **D** is not large in this case, this example is important because the stretch created by **D** is in all directions since

it was obtained by manually selecting the corresponding points to define the TPS, and care was taken to see that the corresponding points were specified arbitrarily so that their positions did not follow any regular pattern. (refer to Figure 2.20 and Figure 2.23e). Hence the arrows in the **D** diagram (Figure 4.46a) are in all directions, and do not follow any particular pattern. Hence it is important to see that even in this case E was less than a pixel in most cases and that UICP was able to give a large number of points.

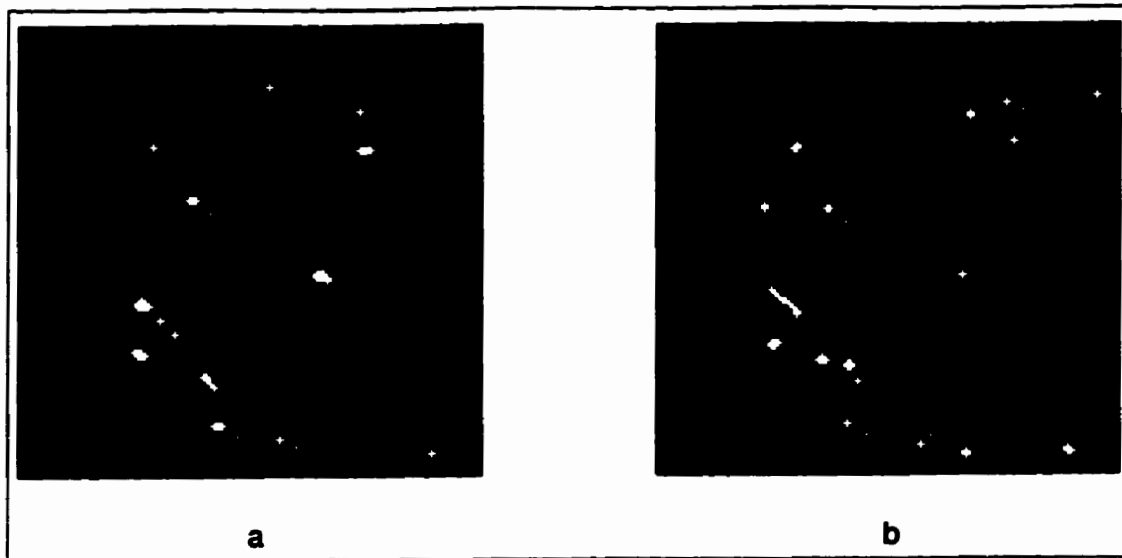


Figure 4.39: *a* and *b*: Figure 3.19c and Figure 3.18c with 10 regions marked.

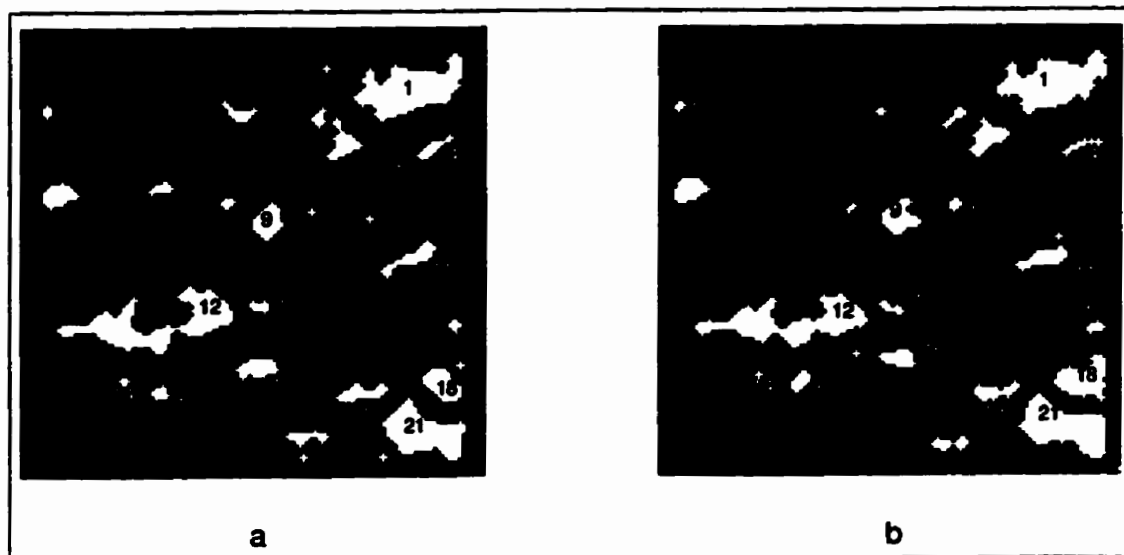


Figure 4.40: *a* and *b*: Figure 3.19d and Figure 3.18d with 21 regions marked.

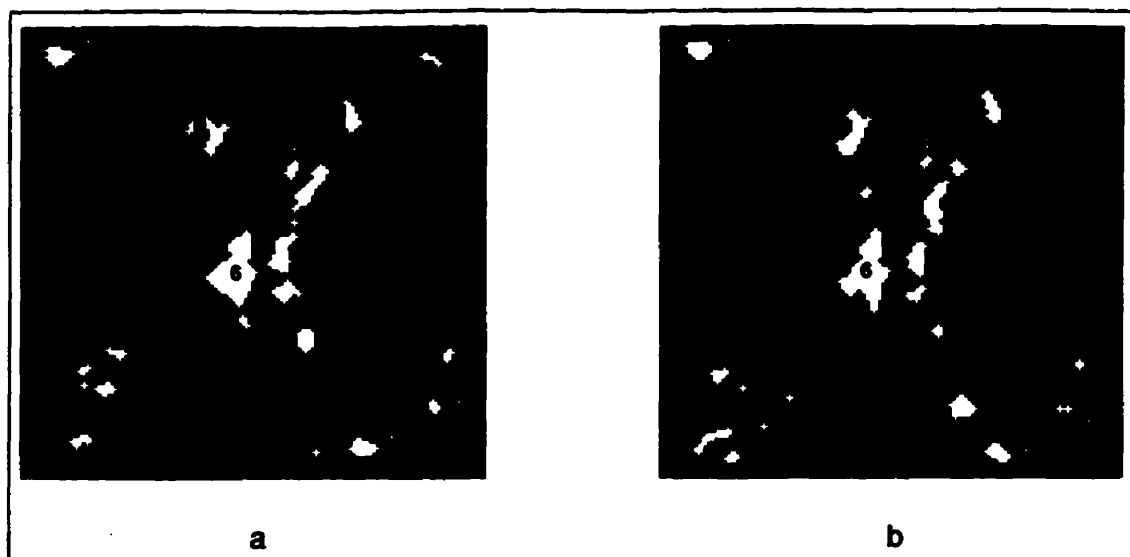


Figure 4.41: *a* and *b*: Figure 3.19f and Figure 3.18f with 10 regions marked after being modified by UICP.



Figure 4.42: *a* and *b*: Figure 3.19i and Figure 3.18i (after modification with UICP) with 14 regions marked.

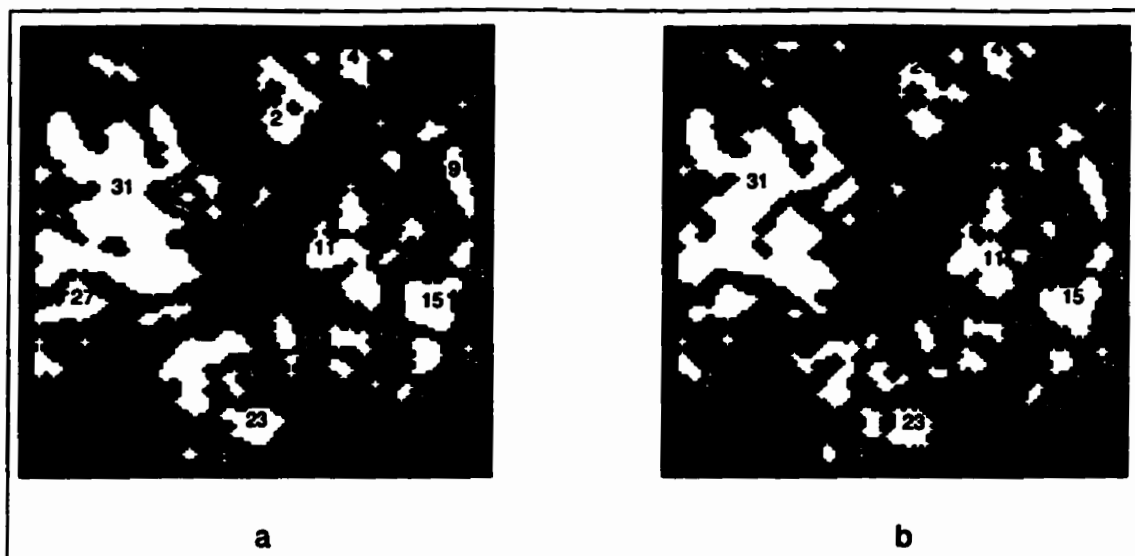


Figure 4.43: *a* and *b*: Figure 3.19k and Figure 3.18k with 31 regions marked after being modified by UICP.

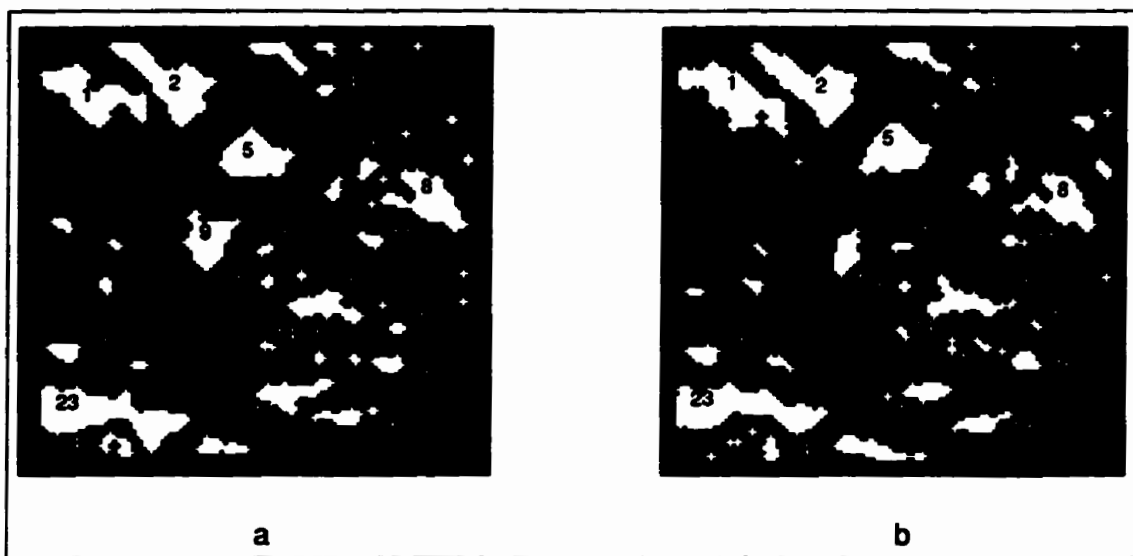


Figure 4.44: *a* and *b*: Figure 3.19m (after modification with UICP) and Figure 3.18m with 24 regions marked.



Figure 4.45: *a* and *b*: Figure 3.19n and Figure 3.18n with 10 regions marked.

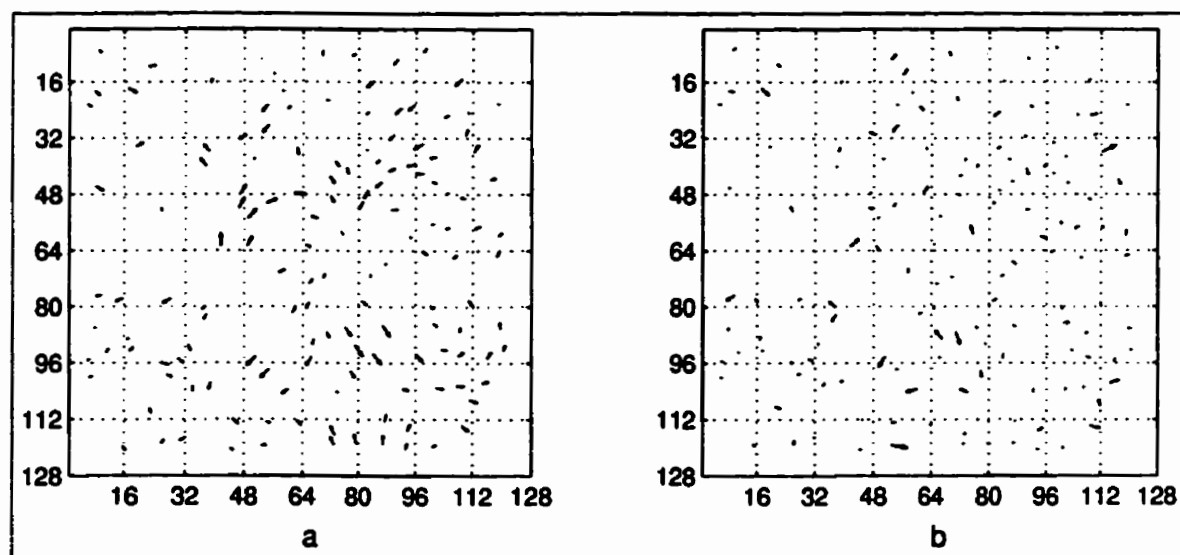


Figure 4.46: *a* and *b*: D and E for the original images (Figure 2.24a and its TPS-distorted version using the distortion depicted in Figure 2.23e).

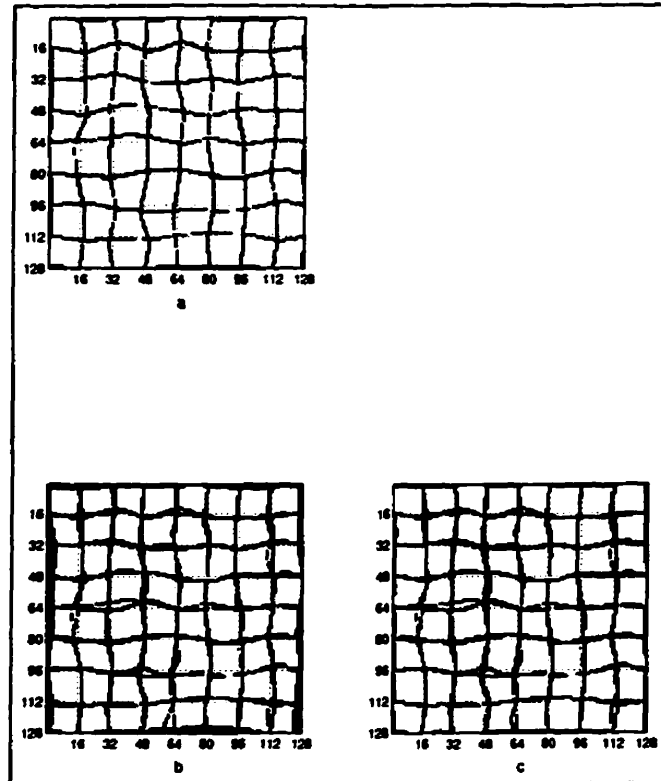


Figure 4.47: *a*: TPS distortion shown as a grid diagram. *b*: Registration achieved after using UICP. *c*: Registration achieved after using UICP with boundary condition.

The final registration achieved is shown in Figure 4.47 as a grid diagram. Figure 4.47a shows the actual distortion between the original images (TPS distortion), while Figure 4.47b shows the registration achieved after using the control points obtained using UICP, and TPS-unwarping. As can be seen, portions of the image, particularly the boundary are unregistered. As mentioned in the chapter “The starbyte transformation”, for the TPS distortion, the boundary of the reference image maps on to the boundary of the target image. This is not a restrictive assumption, because, in practice, the boundary of a given image has zero intensity and it is always possible to impose this boundary condition without loss of generality. If we apply this additional

boundary constraint along with the control points obtained through UICP, we obtain Figure 4.47c. A comparison of Figures 4.47b and c shows that the final registration in Figure 4.47c is much better than in Figure 4.47b. Hence, for the final registration (discussed in the chapter “Image registration”), we shall use the boundary constraint to obtain the final unwarped image.

4.9 Sinusoidal distortion

Here, the original images were Figure 2.2a and its sinusoidally-distorted counterpart using equation (2.2) (refer Figure 2.23f). The starbyte images obtained using protocol 1 are shown in the previous chapter (Figure 3.20a and b) and the cleaned-up density slices using an OC operation are also shown (Figure 3.21 and Figure 3.22). In this case, slices 3, 10 and 12 were altered on both sides, whereas slices 4 and 13 were altered only on the left-hand side using UICP. 97 control points were obtained. There were 4 slices with 10 or more identifiable regions (slices 3, 5, 10 and 12). These slices contributed 70 points, while the remaining contributed a total of 27 points. As before those slices with 10 or more regions have been shown. Figure 4.48, Figure 4.49, Figure 4.50 and Figure 4.51 show 20, 20, 18 and 12 regions on slices 3, 5, 10 and 12 respectively. Figure 4.52a and b show **D** and **E** respectively. Except for a couple of points, **E** is low in all the other cases. Looking at the **D** plot, it is clear that the distortion created by equation (2.2) does not follow any regular pattern (the arrows are in all directions), and also that the magnitudes of distortion are large. In spite of this, we were able to identify matching regions using UICP successfully.

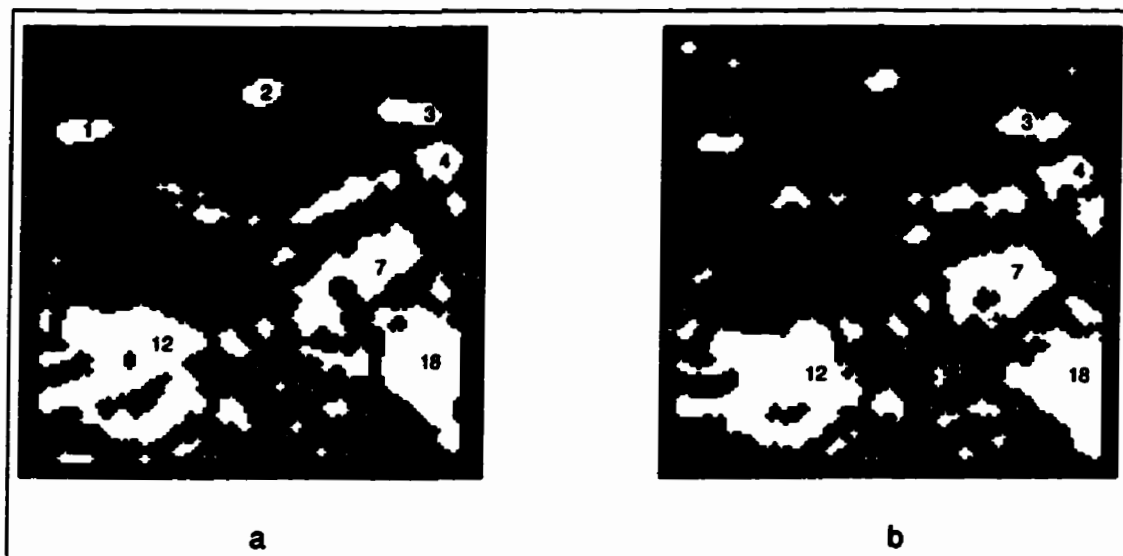


Figure 4.48: *a* and *b*: Figure 3.22d and Figure 3.21d with 20 regions marked after being modified by UICP.

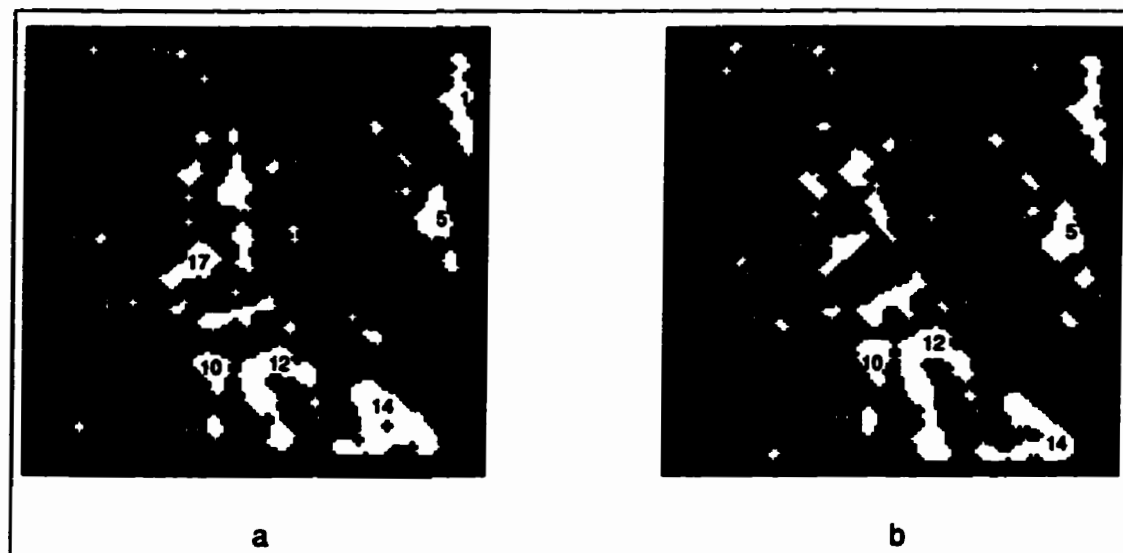


Figure 4.49: *a* and *b*: Figure 3.22f and Figure 3.21f with 20 regions marked.

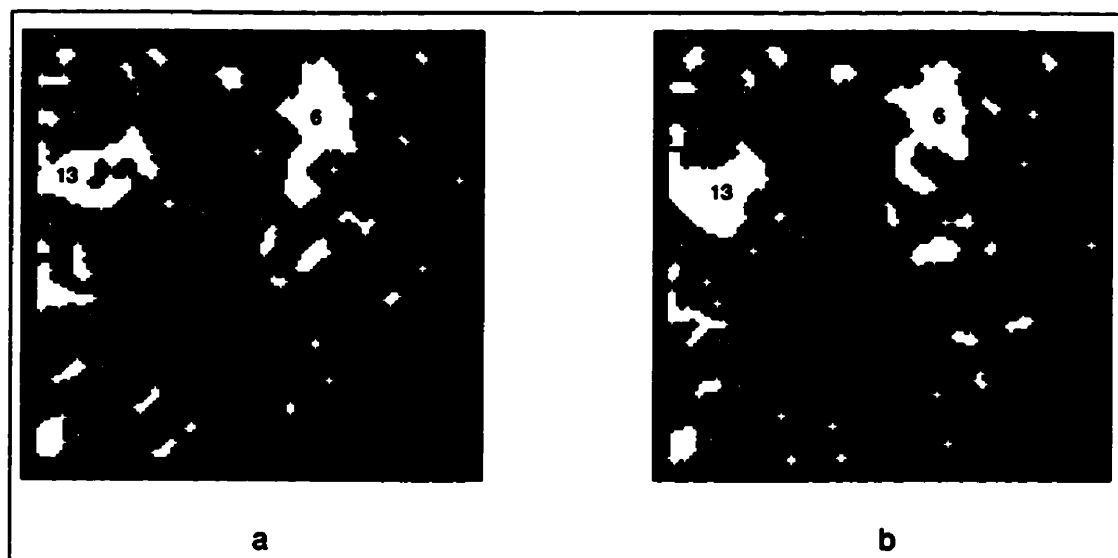


Figure 4.50: *a* and *b*: Figure 3.22k and Figure 3.21k with 18 regions marked after being modified by UICP.

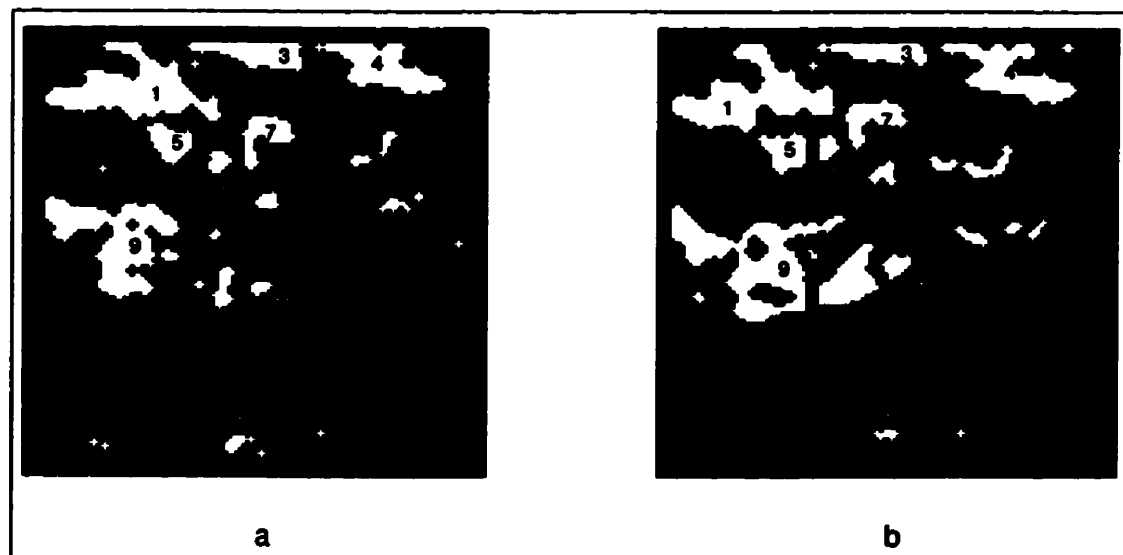


Figure 4.51: *a* and *b*: Figure 3.22m and Figure 3.21m with 12 regions marked after being modified by UICP.

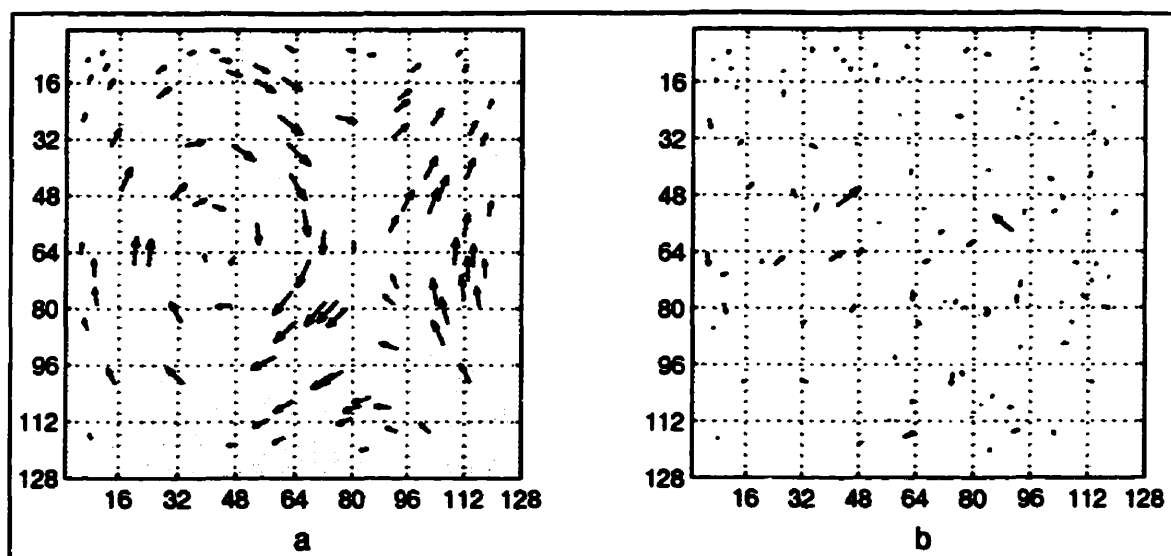


Figure 4.52: *a* and *b*: *D* and *E* for the original images (Figure 2.24a and its sinusoidally-distorted version using equation (2.2)).

Table (4.6) summarizes the results obtained using UICP for these 6 types of distortions. The number of control points obtained in each case, the largest error in pixels, and the percentage of control points over different *E* ranges are given.

4.10 Iterative UICP

Before we conclude this chapter, we would like to also discuss the possibility of iteratively using UICP. The need for this arises when there is a large amount of local distortion relating the original images. In this case, while one is able to get a reasonable number of control points using UICP, some of them may be inaccurate (e.g. the sinusoidal distortion with two large errors). In such a case, it would be desirable to create an intermediate image from the original image using the results of the first pass of UICP. UICP can once again be run on the cleaned-up STs of the intermediate and target images. The results obtained will be more reliable, because

Type of Distortion	N	Largest Error	Percentage of control points with				
			$E < 1$	$1 \leq E < 2$	$2 \leq E < 3$	$3 \leq E < 4$	$E \geq 4$
Bilinear	89	3.40	61.80	24.72	10.11	3.37	0.00
Translation	122	1.70	98.36	1.64	0.00	0.00	0.00
Rotation	60	4.15	56.67	33.33	8.33	0.00	1.67
Scaling	95	3.50	82.11	9.47	5.26	3.16	0.00
TPS	143	4.63	60.14	25.87	9.79	2.80	1.40
Sinusoid	97	8.14	54.64	29.90	9.28	3.09	3.09
*Sinusoid	94	5.46	72.34	17.02	6.38	3.19	1.06

* Results for the sinusoid distortion after the second pass of UICP.

Table 4.6: *Column 1:* Type of distortion. *Column 2:* N, the number of control points obtained using the user-interactive program. *Column 3:* Largest error in pixels. *Column 4:* Distribution of control points over different error (E) ranges measured in pixels.

performing the manual matching between the intermediate and target image would be simpler than the manual matching between the original and target image. We have demonstrated this using the sinusoidal distortion presented above. Using the control points given by the first run of UICP, we create an intermediate image from the original image using thin plate splines. A total of 94 points was obtained with the second pass of UICP.

Figures 4.53a to c shows the cleaned-up STs of the original, intermediate and target image. A close study of the three images shows that the intermediate image is closer to the target than the original is. However, it also shows that there are still portions of the intermediate image quite dissimilar to the target. Hence there is a need for another run of UICP in this case. The second run of UICP produced a total of 94 control points. While the individual slices have not been presented in this case, we have presented the D and E diagrams (refer Figure 4.54). The largest E value was about 5 pixels and less than in the previous case.

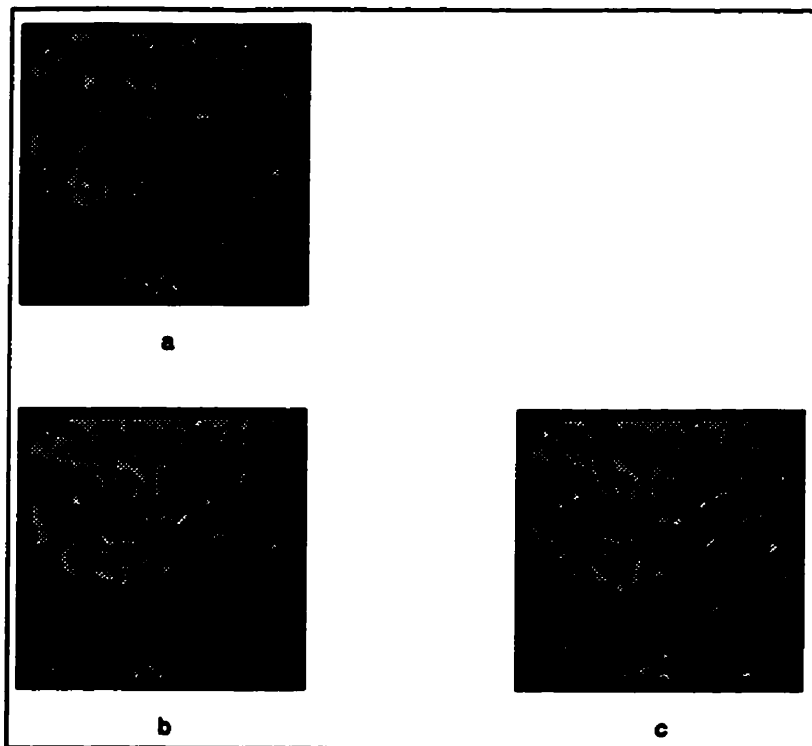


Figure 4.53: *a* to *c*: Cleaned-up STs of the original, intermediate and target images for the sinusoidal distortion.

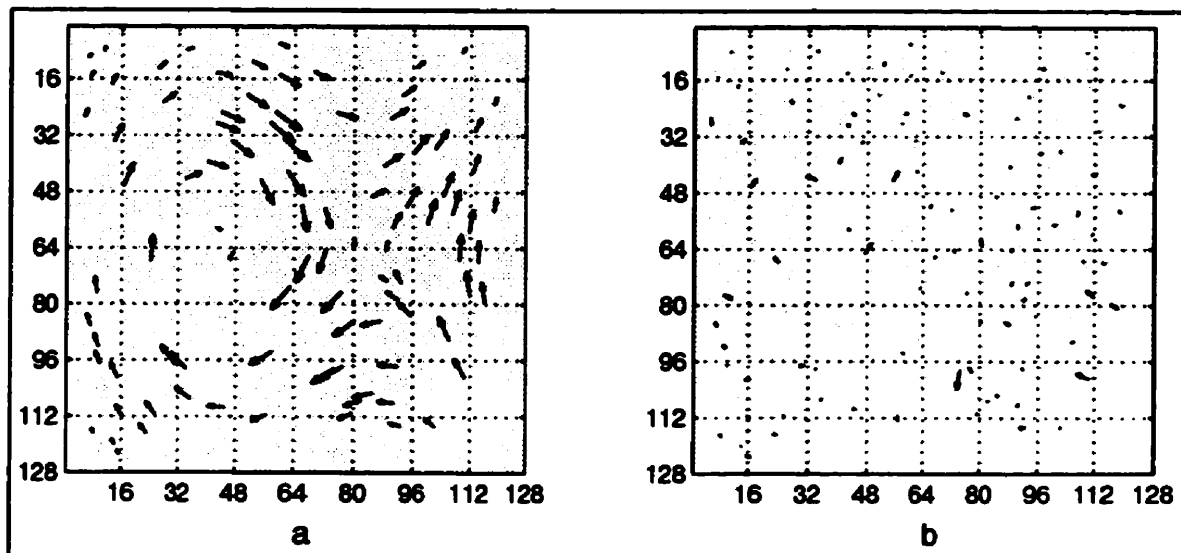


Figure 4.54: *a* and *b*: *D* and *E* for the intermediate and target images for the sinusoidal distortion.

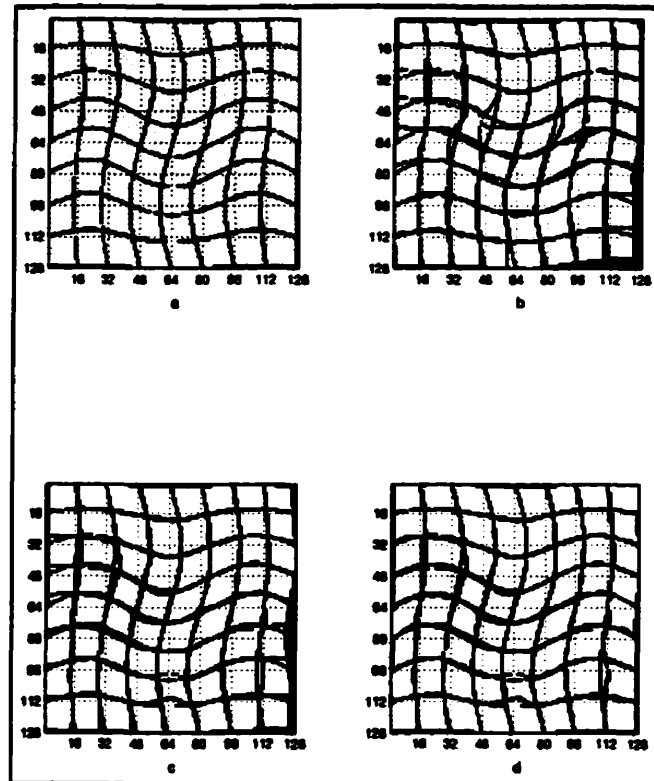


Figure 4.55: *a*: Sinusoidal distortion shown as a grid diagram (equation (2.2)). *b*: Registration achieved after the first pass of UICP. *c*: Registration achieved after the second pass of UICP. *d*: Registration achieved after the second pass of UICP with boundary condition.

Figure 4.55 shows the registration achieved as a grid diagram. Figure 4.55a shows the actual distortion between the given images, while Figure 4.55b shows the registration achieved after the first pass of UICP. As can be seen, while registration has been achieved in some parts of the image, there are portions of the image which are still misregistered. Figure 4.55c shows the registration achieved after the second pass of UICP. Most of the misregistration present after the first pass of UICP has been removed. But the boundary still remains unregistered. If we apply the boundary condition as was done for the TPS distortion, we obtain Figure 4.55d. The registration shown in Figure 4.55d is far better than that shown in Figure 4.55c. Hence, for the

final registration (discussed in Chapter 6), we shall use the boundary constraint to obtain the final unwarped image. The results obtained using the second pass of UICP are also presented in Table (4.6) (last column). By running UICP a second time, we have redistributed the control points across the E range. The number of points with E less than 1 pixel has increased significantly now.

4.11 Conclusions

UICP, a simple interactive program to extract control points from the starbyte-transformed images, was discussed in this chapter. Given two starbyte images, UICP is simple to use, and is based on the user's ability to identify matching regions in corresponding slices of the starbyte images. Since the starbyte transformation converts the original images to texture maps with clearly demarcable regions, identifying matching regions visually is simple. Also, since each slice is binary, region aggregation is also simple and independent of the initial seed point, so that the user can click anywhere on a region in order to specify a seed point. We showed for a range of distortions relating the original images (bilinear, translation, rotation, scaling, TPS and sinusoidal), that it is possible to extract a large number of accurate points using UICP. We also discussed using UICP iteratively. This can be done when the distortion relating the original images has a high degree of local variation. In this case, using UICP iteratively improves the accuracy of the results. These points can then be used to perform image registration. In the next chapter, we shall discuss an automatic procedure to extract control points from the starbyte images.

Chapter 5

AUTOMATIC DETECTION OF CONTROL POINTS

5.1 Introduction

UICP, an interactive program to extract control points, was discussed extensively in the previous chapter. We also presented results obtained using UICP for six different types of distortions applied on different original images. Using the same pairs of reference and target images i.e. the same set of distortions, in this chapter we shall discuss an automated method to extract control points. We shall first describe the algorithm, AUTOCP, used for automation. We shall then present the results obtained for the same set of examples as was used in the previous chapter.

5.2 Description of AUTOCP

As before, the individual slices of the STs of the original and target image are cleaned-up using an OC operation and reassembled to obtain the “cleaned-up” STs. After this, all the regions in each slice of both the STs are aggregated. AUTOCP uses 4 parameters for matching regions in a slice of the original image to regions in the corresponding slice in the target image. Thus each region is represented as a 4-element vector, comprising the X - and Y - locations of the centroids of the region, Hu’s first invariant [40], [57], [57] calculated over the region and the number of pixels in the region. In order to define Hu’s first invariant, we briefly review moment theory

below. This explanation has been taken from [40].

Given a two-dimensional continuous function $f(x, y)$ we define the moment of order $(p + q)$ by the relation

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \quad (5.1)$$

for $p, q = 0, 1, 2, \dots$. A uniqueness theorem ([88]) states that if $f(x, y)$ is piecewise continuous and has nonzero values only in a finite part of the $x - y$ plane, then moments of all orders exist and the moment sequence (m_{pq}) is uniquely determined by $f(x, y)$. Conversely (m_{pq}) uniquely determines $f(x, y)$. The *central moments* can be expressed as

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy \quad (5.2)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}$$

and

$$\bar{y} = \frac{m_{01}}{m_{00}}$$

For a digital image, the above equation becomes

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \quad (5.3)$$

The *normalized central moment* is given by:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad (5.4)$$

where

$$\gamma = \frac{p + q}{2} + 1$$

for $p + q = 2, 3, \dots$. From the second and third moments, a set of seven *invariant moments* has been derived by Hu [58]. These invariant moments are invariant to

translation, rotation and scaling. We have used Hu's first invariant as one of the four parameters in AUTOCP. Hu's first invariant ϕ_1 is given by:

$$\phi = \eta_{20} + \eta_{02} \quad (5.5)$$

Thus each slice in both STs was represented as a 4-column list containing these four parameters. Since these four quantities will have different ranges of values, normalization was done so that they all have the same ranges, so that they can be treated equally. This will be discussed in the next section.

Using the X - and Y -locations of the centroid of the region as parameters implies that the automated program cannot detect an indefinite amount of translation or any other distortion which pulls each pixel by an indefinite amount. Also using the number of pixels as a criterion for automation implies that AUTOCP cannot detect an indefinite amount of scaling or any other distortion which enlarges or shrinks each region by an indefinite amount. This does not pose a serious limitation to the use of AUTOCP because the types of transformations and the magnitude of distortions created by them has been chosen to be consistent with the amount of distortion observed empirically in breast imaging. For e.g., the translation observed not greater than 10 %, and rotation is not greater than 10 deg. Ideally, if the distortion were known to be only a rigid distortion, it would be enough to use Hu's invariants as criteria for matching regions. However, in a real breast imaging situation, local variations are also observed and hence it is more important to choose criteria which will be able to work reasonably well with all types of distortions. Using the other Hu's invariants did not give as good results as using the first invariant alone as a criterion, and hence it was decided that the above-mentioned four parameters are sufficient. It was also found that very small regions got mismatched and hence only regions equal to or larger than a size s were considered. s was fixed empirically at 10 pixels by trial

and error.

5.2.1 Acceptance/Rejection criteria

In order to avoid mismatches, we have to define rejection criteria within AUTOCP, which will ensure that while good matches are retained, bad matches are definitely rejected. Optimally, we would like all good matches to be retained and all bad matches to be rejected. If this were to happen, then AUTOCP would perform as well as UICP and we would get comparable results using both. However, in practice, since it is more important to reject bad matches than accept good ones, we have to sacrifice a number of good matches to be sure that all bad matches are eliminated. Hence the number of control points obtained using AUTOCP will be less than the number obtained using UICP. This is what we observed for the different distortions.

We have already explained that control points can be independently detected from each slice pair. AUTOCP loads the 4-column lists corresponding to the regions in the reference and target images. Regions in the target list are normalized so that values in every column are from 0 to 1. Then corresponding columns in the reference list are normalized using the *same constants used for normalization in the target list*. A region on the reference slice is compared with every region in the target slice. For every such comparison, an “error” is calculated between the 2 four-element vectors representing the regions. For e.g, let (r_1, r_2, r_3, r_4) be the four-element vector representing a region in the ST of the reference image while (t_1, t_2, t_3, t_4) represent the same for the target image. We calculated e as:

$$e = \sum_{i=1}^4 |r_i - t_i| \quad (5.6)$$

Thus e was the sum of the absolute difference between the two vectors, the difference

being calculated element-wise. For a region in the reference slice, a list of e values was obtained for every region the target slice. The smallest and second-smallest values were used for deciding whether a match was to be retained or eliminated. Let e_b represent the best match, while e_{sb} represent the second best match. Two heuristic criteria were used for acceptance/rejection. These were:

1. If e_b was greater than or equal to a constant T_1 , then the match was rejected.
2. If $\frac{e_{sb}}{e_b}$ was less than T_2 , then the match was rejected.

The first condition defines a bound on the error corresponding to the best match. The second condition defines an amount by which the best match is better than the second best match. By observing the accuracy of matches returned by the algorithm for various settings of T_1 and T_2 , empirical values for T_1 and T_2 were fixed at .07 and 3 respectively. It was observed that the same values could be used for all the transformations. The same values are also used in Chapter 9 to extract control points automatically from real breast MRI slice pairs.

Amongst matches passed by both these criteria, it was found that sometimes a region in one slice was matched to more than one region in the other. In such a situation, both matches were eliminated. For most of the distortions, portions of the target image were outside the border of the original images. Hence matches too close to the border of the image were found to be unreliable and these were deleted. We chose a border size of 16 pixels on all sides.

Using these conditions, as well as the two criteria discussed above we got accurate matches for the six distortions. However, because of the strong rejection criteria imposed, for the more complicated distortions, the number of control points yielded by AUTOCP was not sufficient for image registration. Hence, it was necessary to use AUTOCP iteratively. However, just a few runs of AUTOCP increased the number of

control points significantly.

We shall now briefly discuss using AUTOCP in the iterative mode. Results obtained using AUTOCP singly and iteratively will be presented for each distortion separately.

5.2.2 Iterative AUTOCP

Let R and T represent the reference and target STs respectively. The first run of AUTOCP produces a set of control points S_1 . We shall use this set S and R to generate an intermediate image I_1 . While I_1 should be close to T , if the distortion relating R and T is complicated and contains a large amount of local variation, this will not be the case. We will then use the STs of I_1 and T to obtain a new set of control points S_2 using AUTOCP. We found that S_2 is always much larger and better distributed than S_1 , because I_1 is closer to T than R is. We now use R and S_2 to produce a new intermediate image I_2 and continue the procedure. The procedure is stopped when the set of control points obtained at an iteration reaches a predefined size. This stopping criterion is heuristic and could have been defined in other different ways. For example, the procedure could be stopped if the percentage increase in control points is less than a threshold, or alternately, if subsequent iterations do not produce any significant increase in the number of control points. It is typically found that the number of control points increases at every iteration up to a stage and then either starts falling or going up and down with no particular pattern. This trend is shown for a few sample cases in chapter 7.

It can be noted that at every iteration, the current intermediate image is always created from the reference image and the current set of points, and not from the previous intermediate image. This is because the creation of every intermediate image involves interpolation. Using one intermediate image to create the next propagates

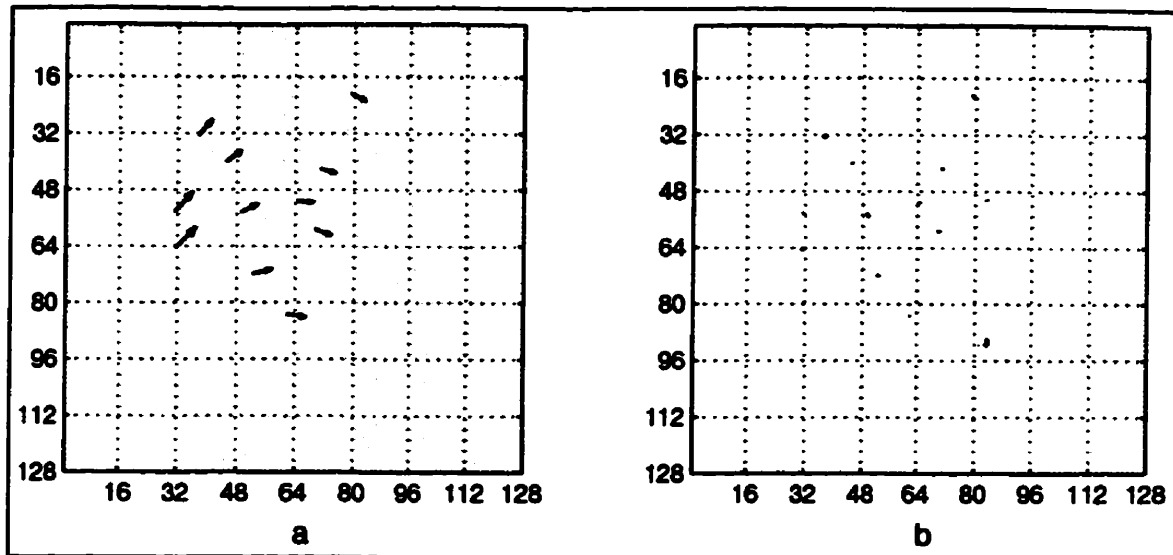


Figure 5.1: *a* and *b*: **D** and **E** for the reference and target images for the bilinear distortion.

the error due to interpolation. Hence it is much more accurate to use the reference image for creating each intermediate image. Also, since every intermediate image is made from the reference image, mapping back the obtained control points to the reference image is simple. Also, after mapping back the obtained control points to the reference image, the previous intermediate image can be discarded and the new one created out of the reference image alone.

We shall now present the results for each of the six distortions. The reference and target images for each case are the same as discussed in the previous chapter. Iterative AUTOCP was used for all the distortions except translation. For translation, only one run of AUTOCP was used.

5.3 Bilinear

Figure 3.4 and Figure 3.5 show the 16 slices for the STs of reference and target image. The first run of AUTOCP produced 11 points. The **D** and **E** diagrams are

shown in Figure 5.1. Even though all the 11 points are accurate, they are clearly not sufficient to obtain the unwarped image. Hence we need to run AUTOCP iteratively.

We produce an intermediate image using TPS and the reference image. The STs of the reference, intermediate and target image are shown in Figures 5.2a to c. It is clear that while the intermediate image is closer to the target image than the reference is, there still remain areas which are dissimilar. The second run of AUTOCP produced 53 control points.

The **D** and **E** diagrams are shown in Figure 5.3. It is clear that the distribution of points in the second set is much better than the first. Also the number of points obtained is much larger.

Figure 5.4a to c shows the registration achieved as a grid diagram. Figure 5.4a

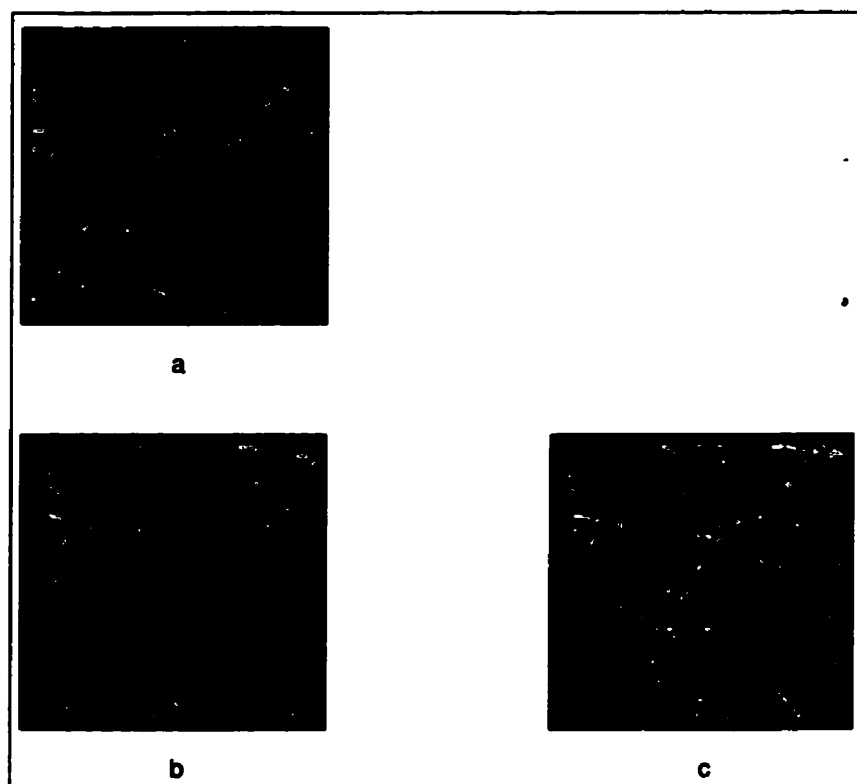


Figure 5.2: *a* to *c*: Cleaned-up STs of the original, intermediate and target images for the bilinear distortion.

shows the actual distortion between the given images, while Figure 5.4b shows the registration achieved after the first pass of AUTOCP. As can be seen, while registration has been achieved in some parts of the image, there are portions of the image which are still misregistered. Figure 5.4c shows the registration achieved after the second pass of AUTOCP. The registration is almost perfect.

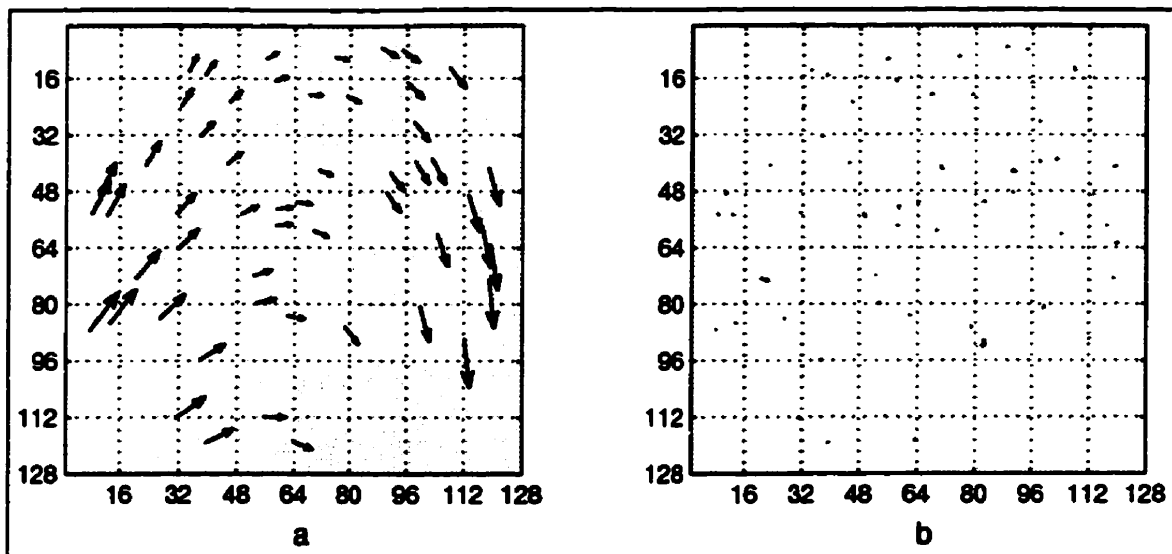


Figure 5.3: *a* and *b*: *D* and *E* for the intermediate and target images for the bilinear distortion.

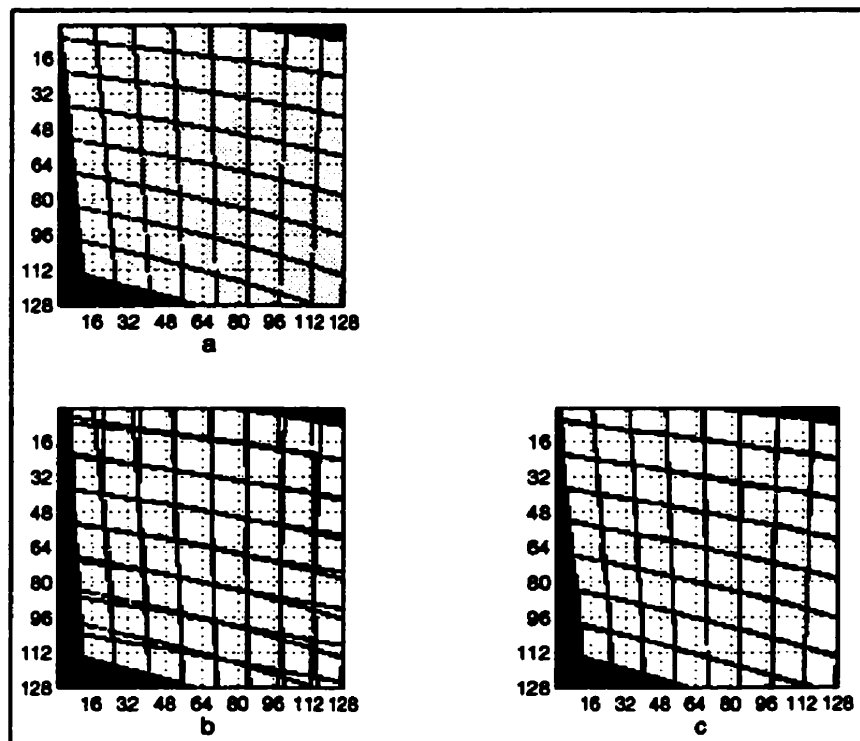


Figure 5.4: *a*: Bilinear distortion shown as a grid diagram (equation (2.1)). *b*: Registration achieved after the first pass of AUTOCF. *c*: Registration achieved after the second pass of AUTOCF.

5.4 Translation

Figure 3.4 and Figure 3.14 show the 16 slices for the STs of the reference and target image. In this case, a total of 17 points was obtained with the first run of AUTOCP. The **D** and **E** diagrams are shown in Figure 5.5. Figure 5.6a shows the actual distortion between the given images, while Figure 5.6b shows the registration achieved after the first pass of AUTOCP. Registration is perfect.

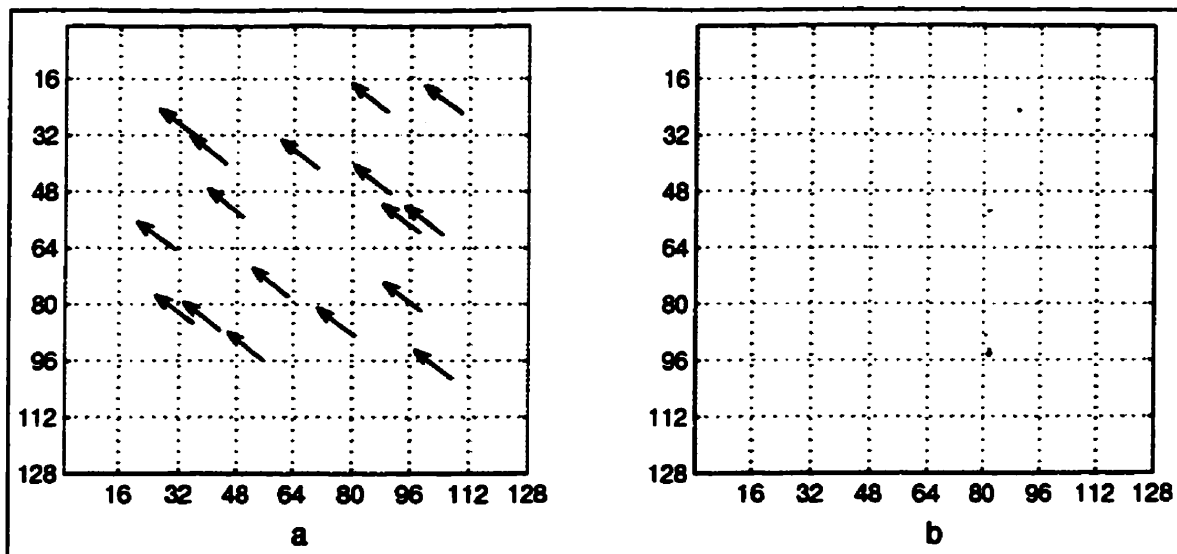


Figure 5.5: *a* and *b*: **D** and **E** for the reference and target images for translation.

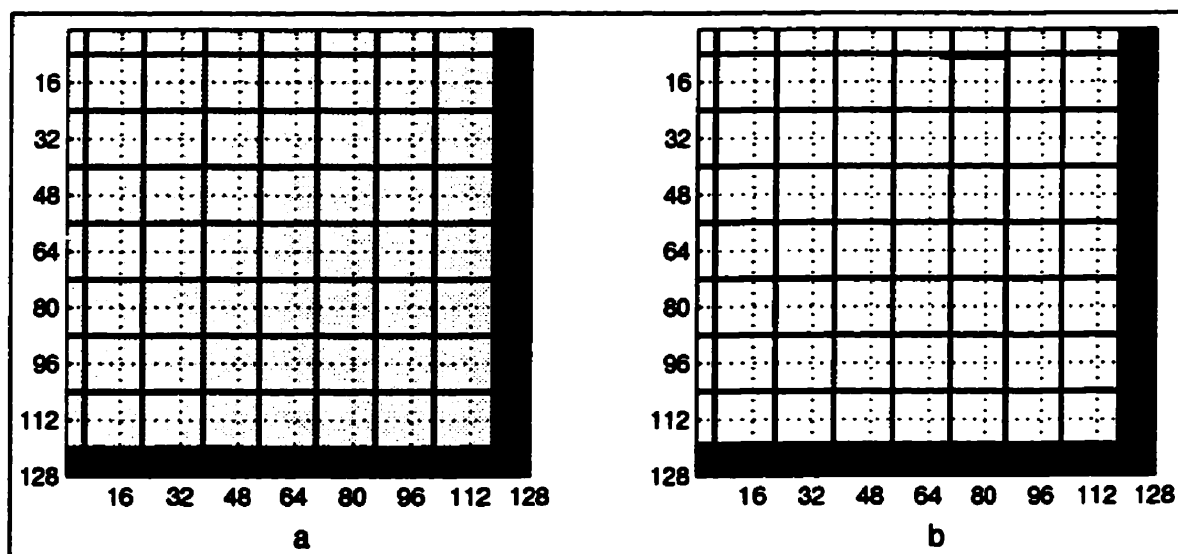


Figure 5.6: *a*: Translation shown as a grid diagram (equation (2.1)). *b*: Registration achieved after one run of AUTOCP.

5.5 Rotation

Figure 3.4 and Figure 3.15 show the 16 slices for the STs of the reference and target image. The first run of AUTOCP produced 9 points. The **D** and **E** diagrams are shown in Figure 5.7.

If a polynomial formula or an affine transformation were used for unwarping, these 9 points would be sufficient. However, these 9 control points are not sufficient to obtain the unwarped image using a general unwarping formula such as a TPS. Hence we need to run AUTOCP iteratively. We produce an intermediate image using TPS and the reference image. The STs of the reference, intermediate and target image are shown in Figures 5.8a to c.

It is clear that while the intermediate image is closer to the target image than the reference is, there still remain areas which are dissimilar. The second run of AUTOCP produced 45 control points. The **D** and **E** diagrams are shown in Figure 5.9. It is clear that the distribution of points in the second set is much better than the first. Also the number of points obtained is much larger.

Figure 5.10a to c shows the registration achieved as a grid diagram. Figure 5.10a shows the actual distortion between the given images, while Figure 5.10b shows the registration achieved after the first pass of AUTOCP. As can be seen, while registration has been achieved in some parts of the image, there are portions of the image which are still misregistered. Figure 5.10c shows the registration achieved after the second pass of AUTOCP. Except for the boundaries, where there is some misregistration, everywhere else, the registration is almost perfect.

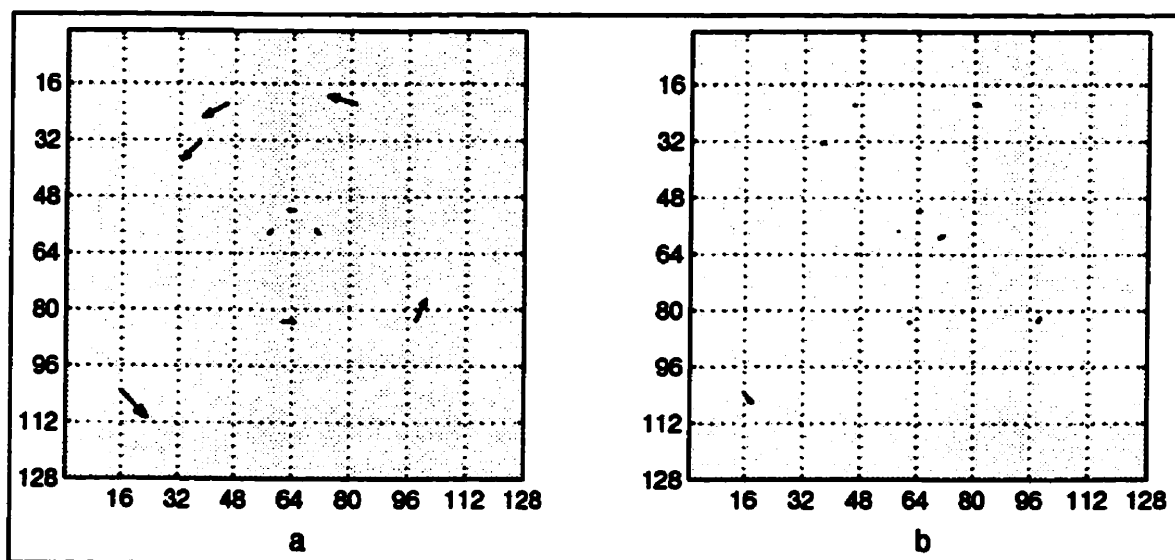


Figure 5.7: *a* and *b*: **D** and **E** for the reference and target images for 10 deg counter-clockwise rotation.

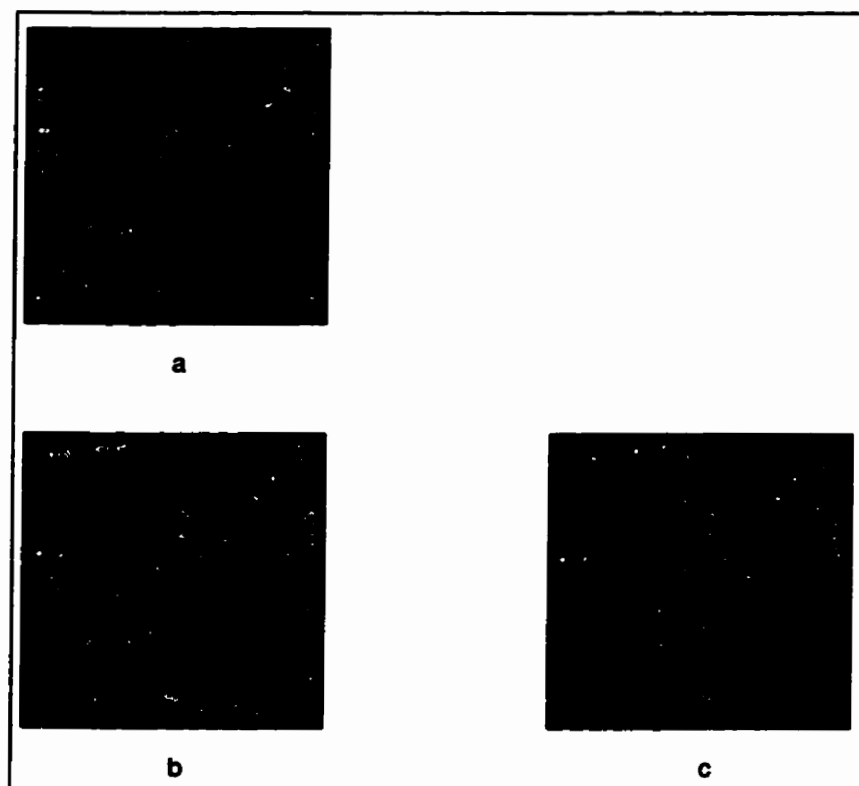


Figure 5.8: *a* to *c*: Cleaned-up STs of the original, intermediate and target images for rotation.

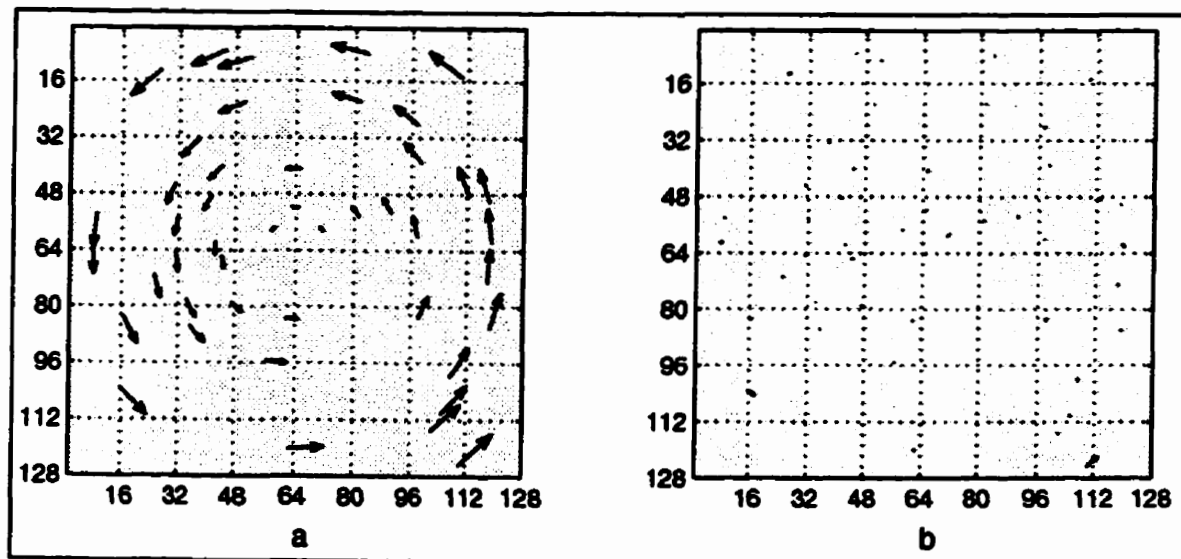


Figure 5.9: a and b : D and E for the intermediate and target images for rotation.

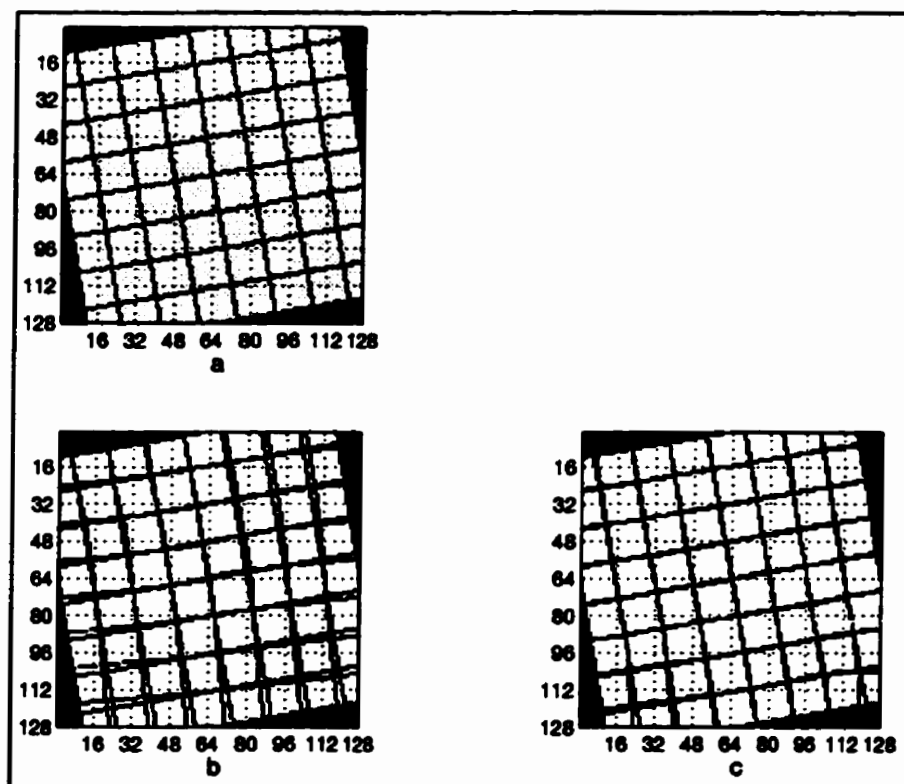


Figure 5.10: *a*: 10 deg counter-clockwise rotation shown as a grid diagram. *b*: Registration achieved after the first pass of AUTOCOP. *c*: Registration achieved after the second pass of AUTOCOP.

5.6 Scaling

Figure 3.4 and Figure 3.16 show the 16 slices for the STs of reference and target image. The first run of AUTOCP produced 24 points. The D and E diagrams are shown in Figure 5.11.

If a polynomial formula or an affine transformation were used for unwarping, these 24 points would have been more than sufficient. However, these 24 control points are not sufficient to obtain the unwarped image using a general unwarping formula such as a TPS. Hence we need to run AUTOCP iteratively. We produce an intermediate image using TPS and the reference image. The STs of the reference, intermediate and target image are shown in Figures 5.12a to c.

It is clear that while the intermediate image is closer to the target image than the reference is, there still remain areas which are dissimilar. The second run of AUTOCP produced 65 control points. The D and E diagrams are shown in Figure 5.13. It is clear that the distribution of points in the second set is much better than the first. Also the number of points obtained is much larger.

Figure 5.14a to c shows the registration achieved as a grid diagram. Figure 5.14a shows the actual distortion between the given images, while Figure 5.14b shows the registration achieved after the first pass of AUTOCP. As can be seen, while registration has been achieved in some parts of the image, there are portions of the image which are still misregistered. Figure 5.14c shows the registration achieved after the second pass of AUTOCP. The final registration is almost perfect.

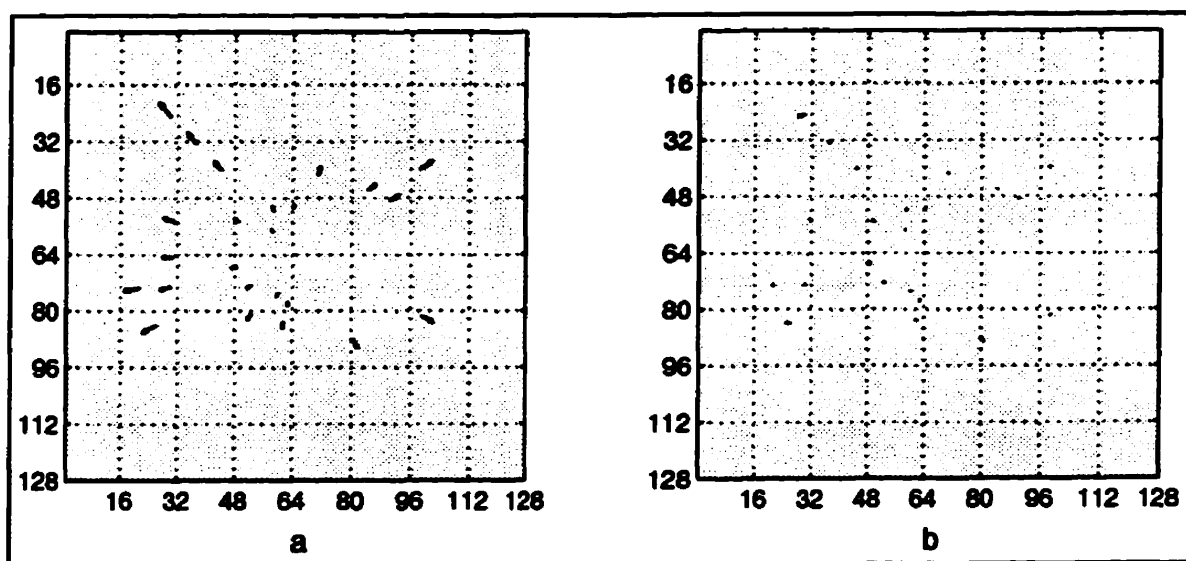


Figure 5.11: *a* and *b*: **D** and **E** for the reference and target images for 10 % enlargement.

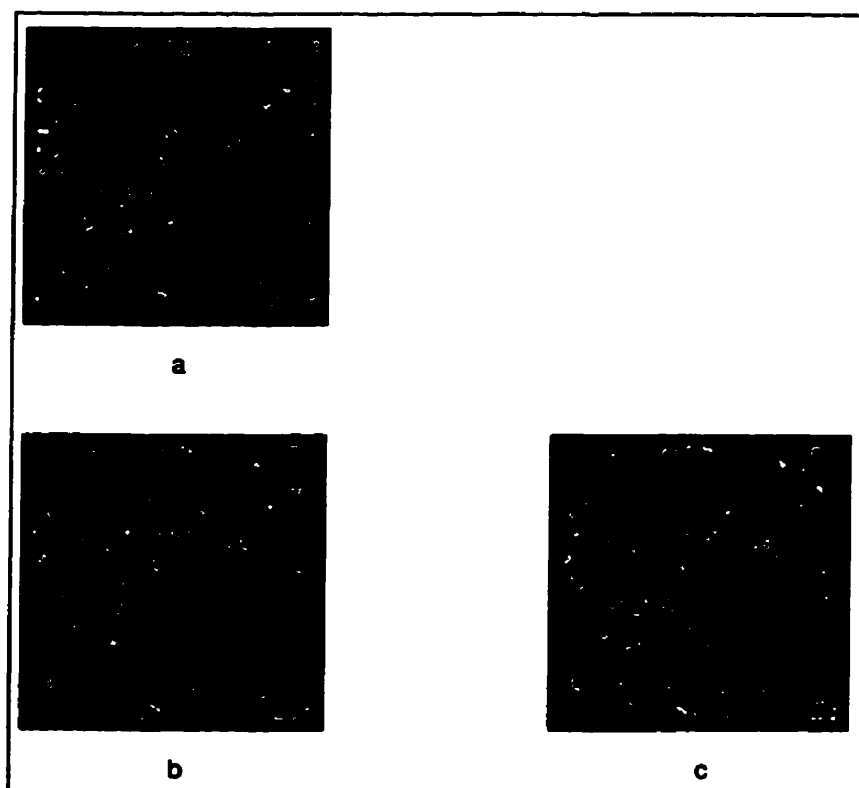


Figure 5.12: *a* to *c*: Cleaned-up STs of the original, intermediate and target images for scaling.

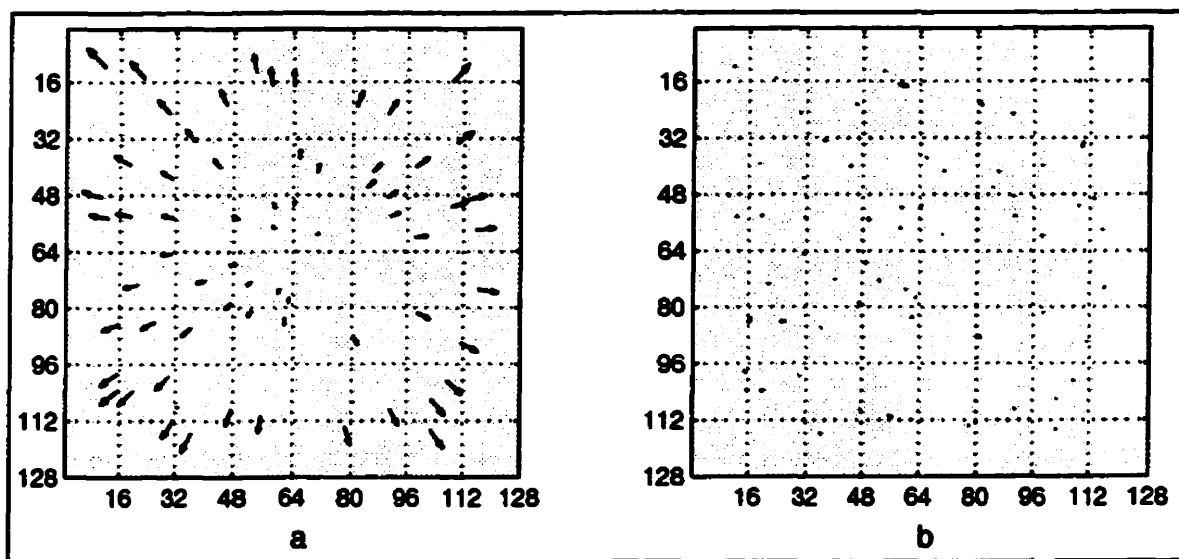


Figure 5.13: *a* and *b*: *D* and *E* for the intermediate and target images for scaling.

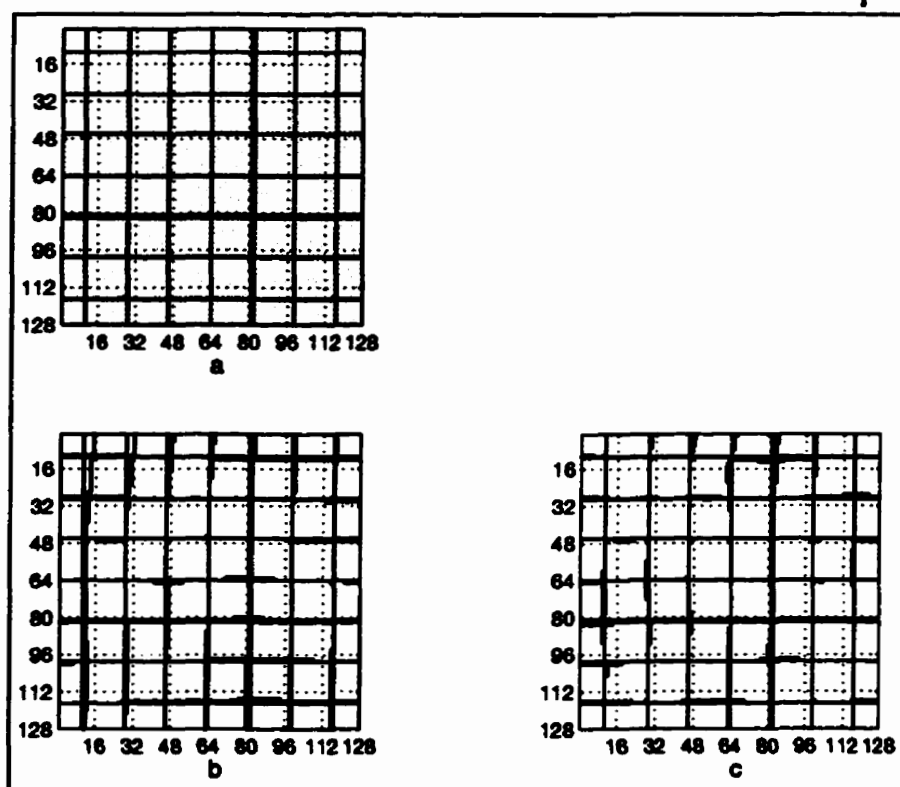


Figure 5.14: *a*: 10 % enlargement shown as a grid diagram. *b*: Registration achieved after the first pass of AUTOCP. *c*: Registration achieved after the second pass of AUTOCP.

5.7 TPS

Figure 3.18 and Figure 3.19 show the 16 slices for the STs of reference and target image. The first run of AUTOCP produced 10 points. The **D** and **E** diagrams are shown in Figure 5.16.

As in the bilinear case, even though all the 17 points are accurate, they are clearly not sufficient to obtain the unwarped image. Hence we need to run AUTOCP iteratively. We produce an intermediate image using TPS and the reference image. The STs of the reference, intermediate and target image are shown in Figures 5.15a to c.

It is clear that while the intermediate image is closer to the target image than the reference is, there still remain areas which are dissimilar. The second run of AUTOCP produced 39 control points. The **D** and **E** diagrams are shown in Figure 5.17. It is clear that the distribution of points in the second set is much better than the first.

Figure 5.18 shows the registration achieved as a grid diagram. Figure 5.18a shows the actual distortion between the given images, while Figure 5.18b shows the registration achieved after the first pass of AUTOCP. As can be seen, while registration has been achieved in some parts of the image, there are portions of the image which are still misregistered. Figure 5.18c shows the registration achieved after the second pass of AUTOCP. Even though the second run of AUTOCP increases the registration, there still remain small portions of the image which are misregistered. This is because the distortion linking the original pair of images is complicated, and also that the boundary is not registered. If we apply the boundary constraint as was done in the manual case with UICP, we obtain Figure 5.18d. We find that the registration shown in Figure 5.18d is better than that in Figure 5.18c. Hence, for the final registration (discussed in the Chapter 6), we shall use the boundary constraint to obtain

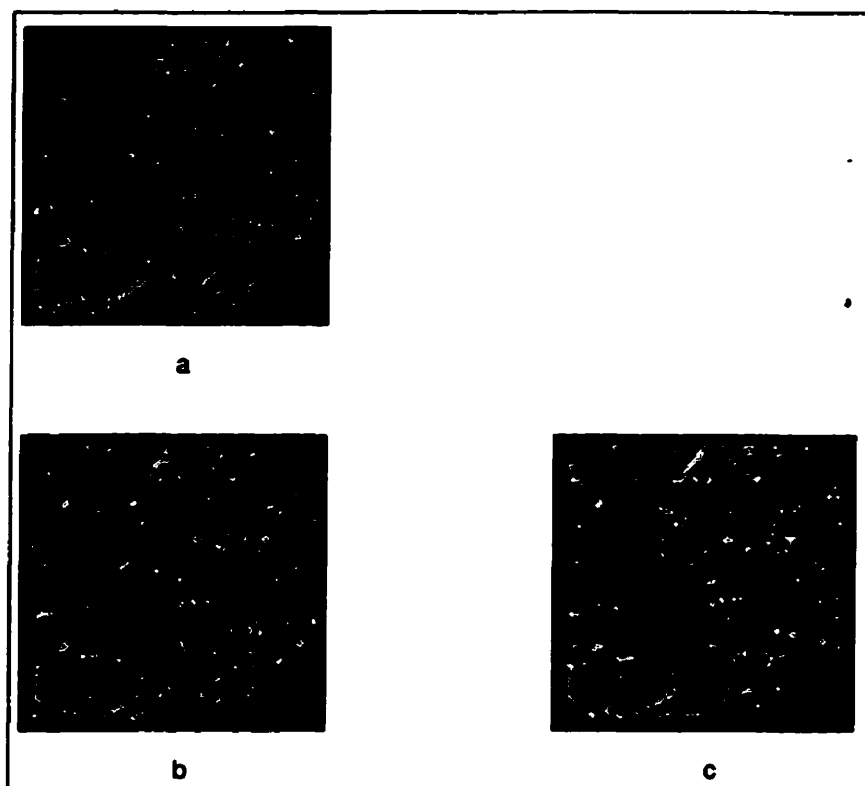


Figure 5.15: *a* to *c*: Cleaned-up STs of the original, intermediate and target images for the TPS distortion.

the final unwarped image.

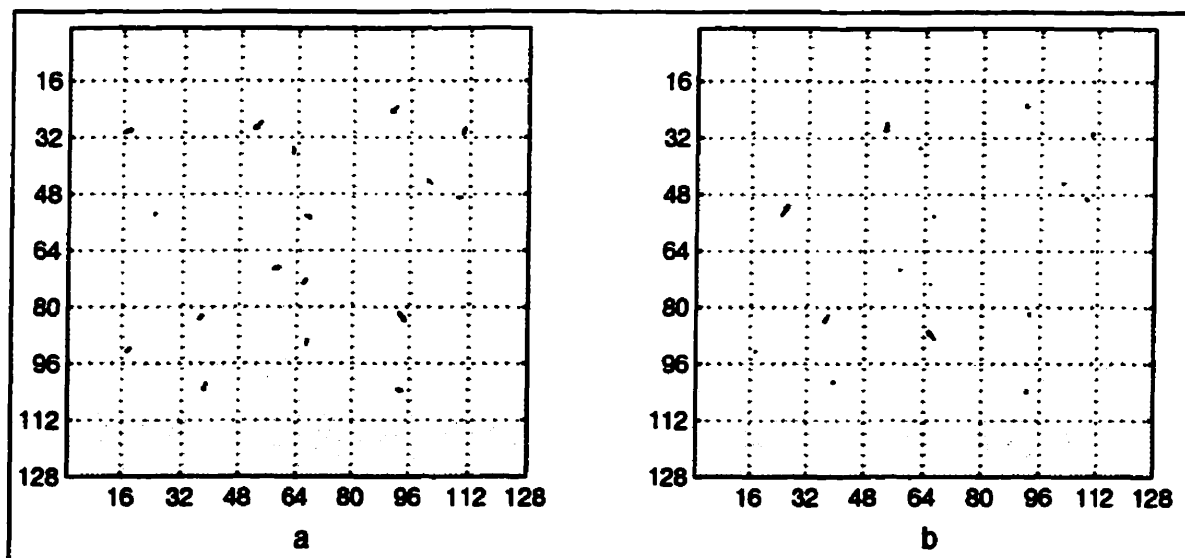


Figure 5.16: *a* and *b*: **D** and **E** for the reference and target images for the TPS distortion.

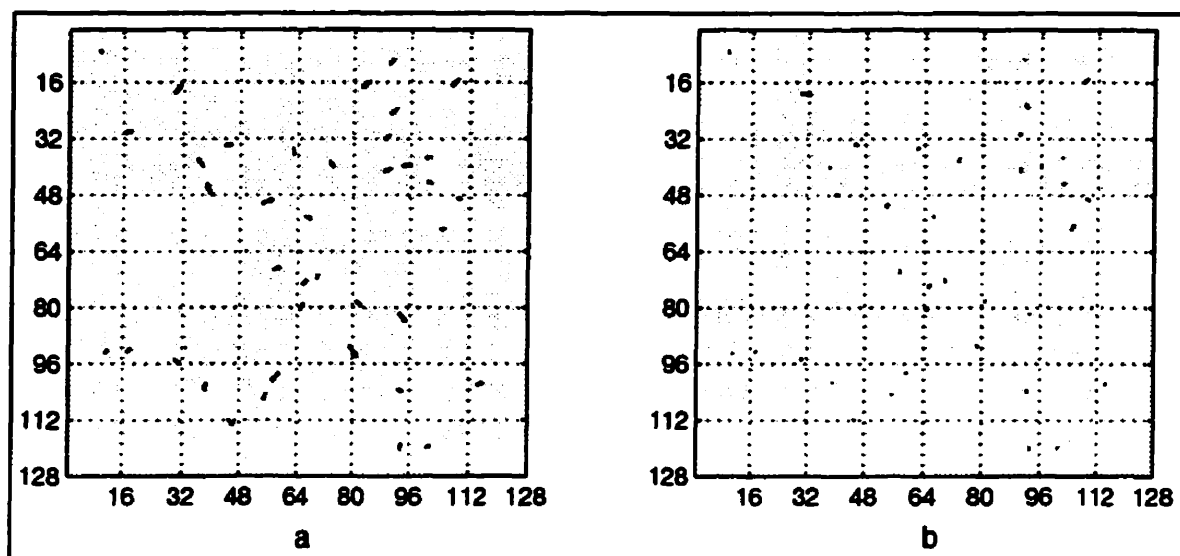


Figure 5.17: *a* and *b*: **D** and **E** for the intermediate and target images for the TPS distortion.

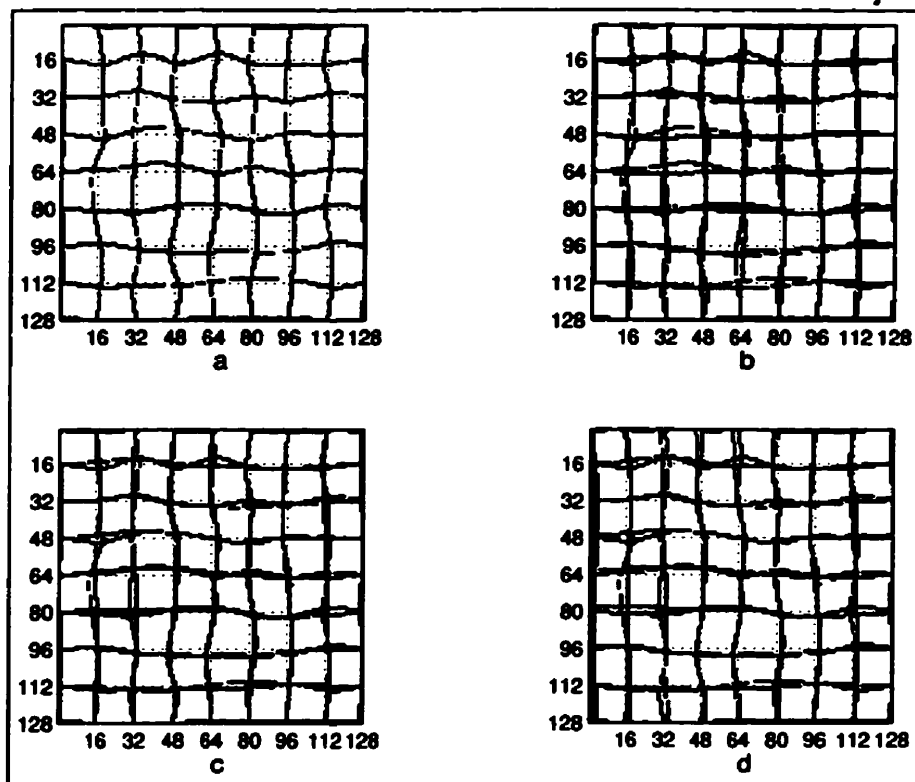


Figure 5.18: *a*: TPS distortion shown as a grid diagram. *b*: Registration achieved after the first pass of AUTOCP. *c*: Registration achieved after the second pass of AUTOCP. *d*: Registration achieved after the second pass of AUTOCP after imposing boundary condition.

5.8 Sinusoid

Figure 3.21 and Figure 3.22 show the 16 slices for the STs of reference and target image. The first run of AUTOCP produced 8 points. The D and E diagrams are shown in Figure 5.19.

As in the earlier cases, even though all the 8 points are accurate, they are clearly not sufficient to obtain the unwarped image. Hence we need to run AUTOCP iteratively. We found that 3 runs of AUTOCP were required in all. The STs of the reference, first intermediate, second intermediate and target image are shown in Figures 5.20a to d.

The second run of AUTOCP produced 29 control points, while the third produced 39 control points. The D and E diagrams are shown in Figure 5.21 for the second run, while Figure 5.22 shows the same for the third run of AUTOCP. From the second set, it is clear that there is one control point in error. In spite of this, other portions of the image are better registered because of the increased number of control points. We used this set of points (with the large error) and obtained the third set which is error-free.

Figure 5.23 shows the registration achieved as a grid diagram. Figure 5.23a shows the actual distortion between the given images, while Figure 5.23b shows the registration achieved after the first pass of AUTOCP. Registration is very poor. Figure 5.23c shows the registration achieved after the second pass of AUTOCP. Due to the one large error, the lower left part of the image is severely distorted. Figure 5.23d shows the registration after the third run of AUTOCP. While the registration is much better here, in the interior, the boundary is not well registered. As for the TPS distortion, we had imposed the condition that the boundary of the reference image maps on to the boundary of the target image. Hence if we now apply this additional constraint

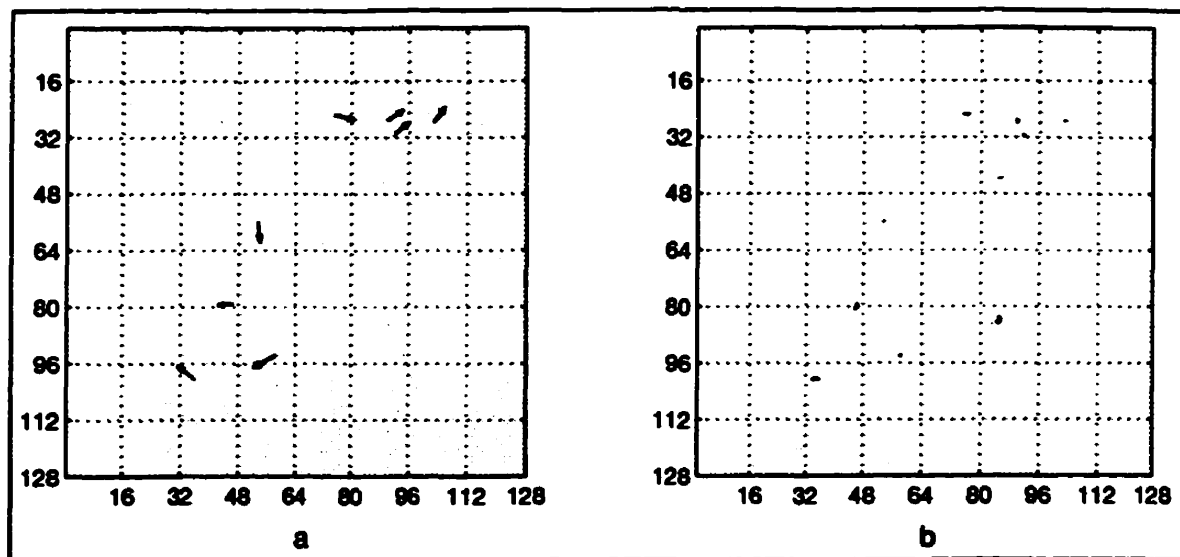


Figure 5.19: *a* and *b*: *D* and *E* for the reference and target images for the sinusoidal distortion.

on the results of the second intermediate image, we obtain Figure 5.23e. We note that there is a marked change, and that there is almost perfect registration, because of the imposition of the boundary condition. Hence, for the final registration (discussed in Chapter 6), we shall use the boundary constraint to obtain the final unwarped image.

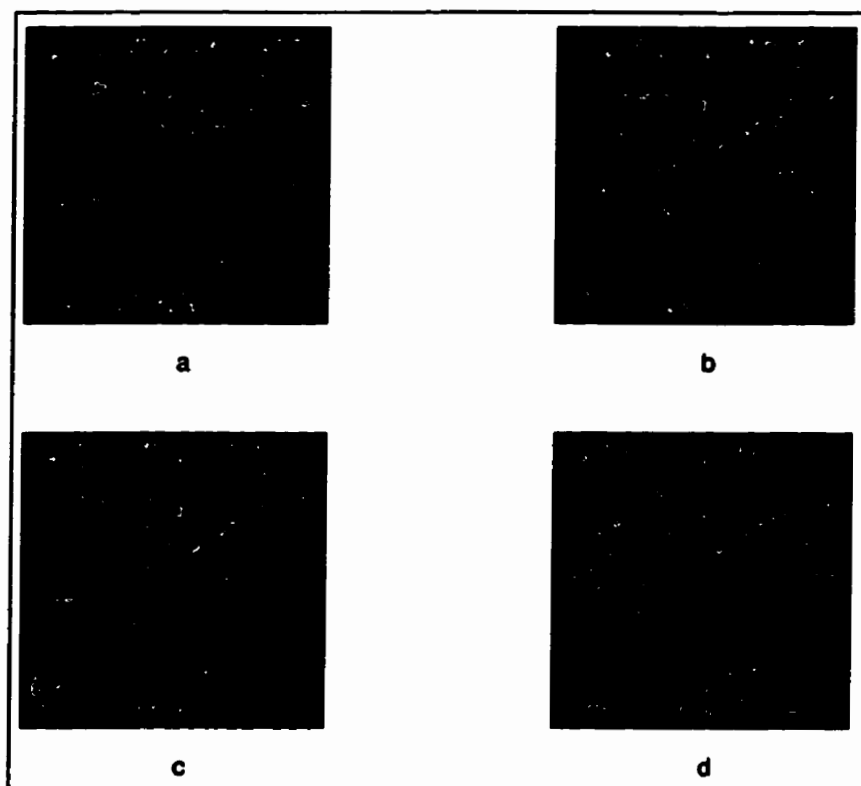


Figure 5.20: *a* to *d*: Cleaned-up STs of the original, first intermediate, second intermediate and target images for the sinusoidal distortion.

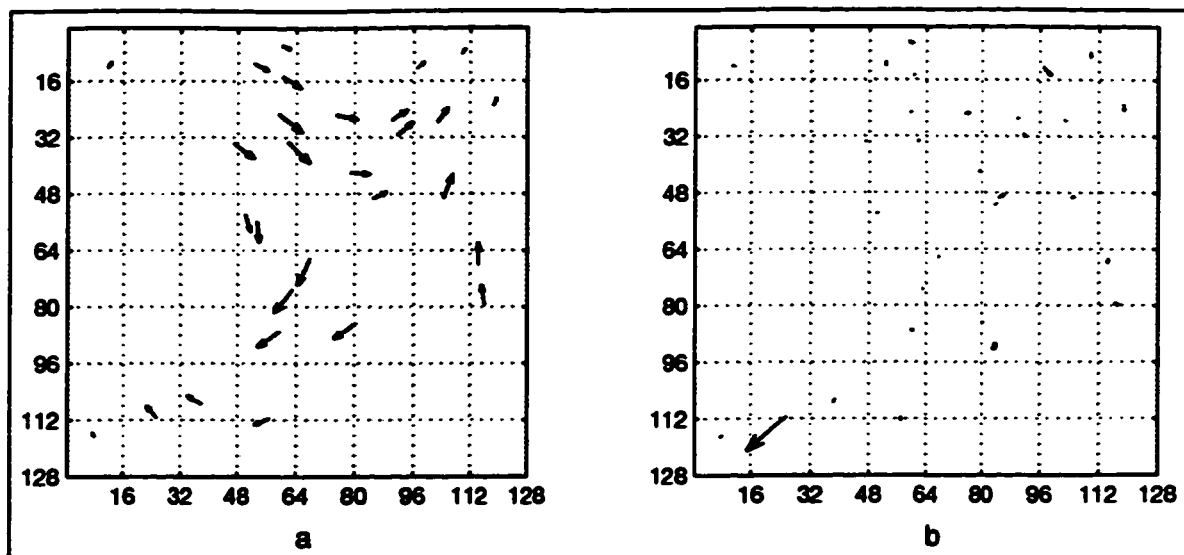


Figure 5.21: *a* and *b*: **D** and **E** for the first intermediate and target images for the sinusoidal distortion.

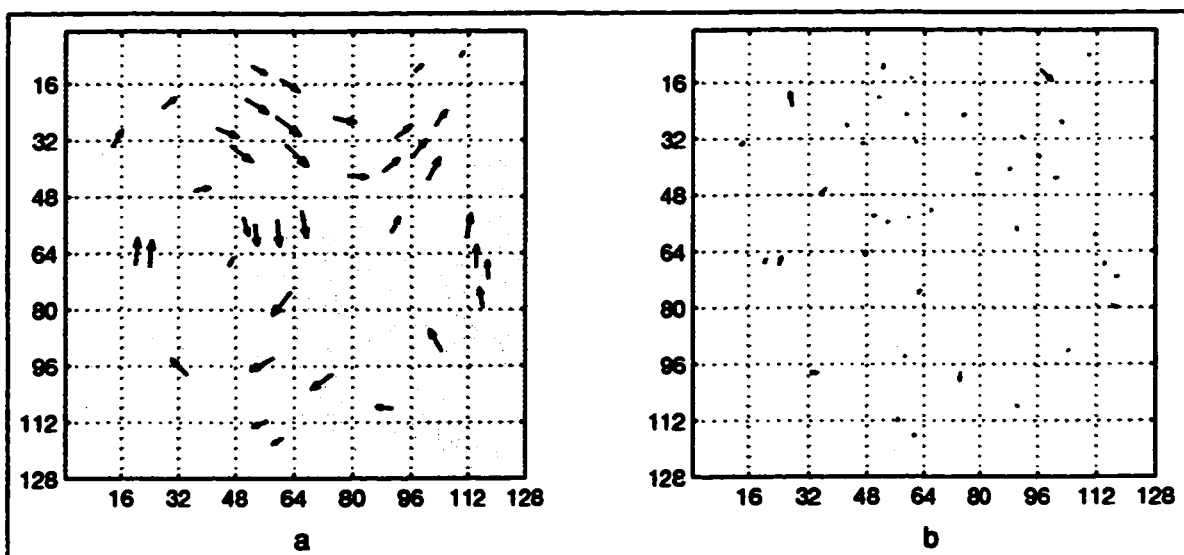


Figure 5.22: *a* and *b*: **D** and **E** for the second intermediate and target images for the sinusoidal distortion.

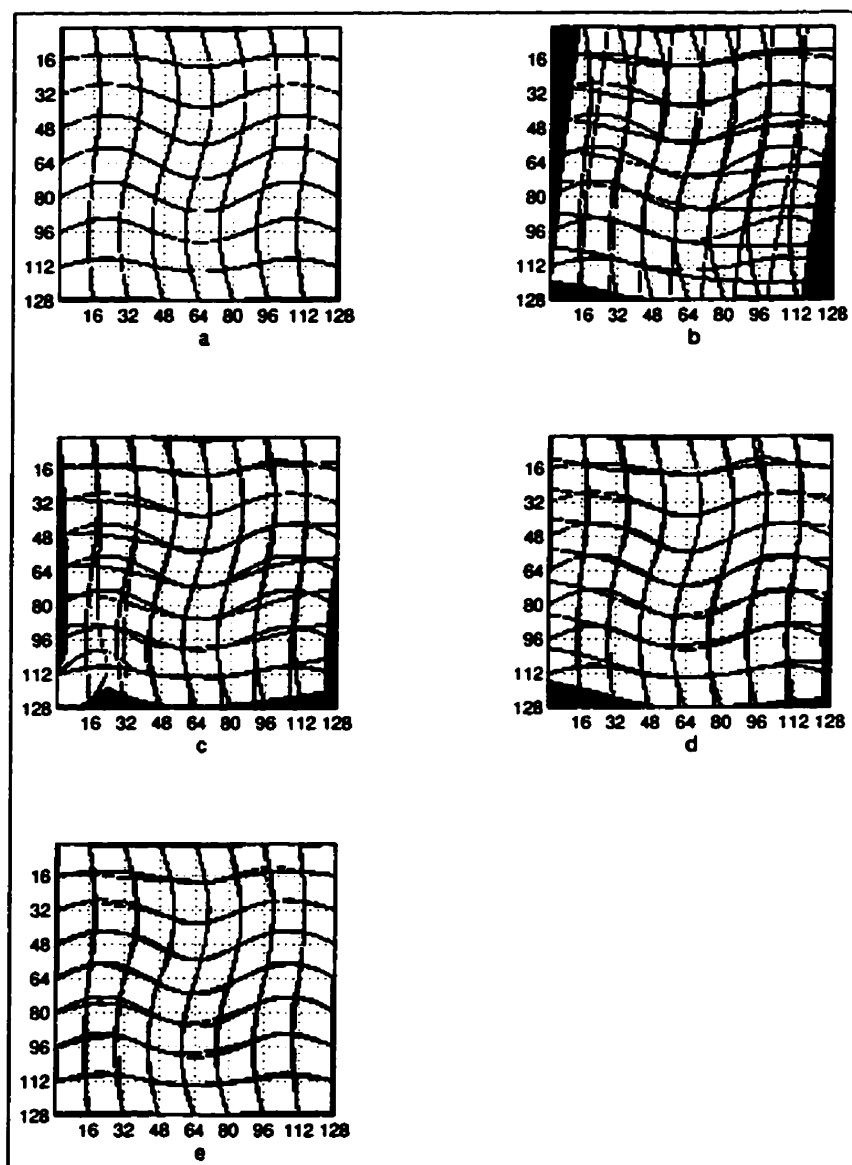


Figure 5.23: *a*: Sinusoidal distortion shown as a grid diagram. *b*: Registration achieved after the first pass of AUTOCP. *c*: Registration achieved after the second pass of AUTOCP. *d*: Registration achieved after the third pass of AUTOCP. *e*: Registration achieved after the third pass of AUTOCP after imposing boundary condition.

Type of Distortion	N	Largest Error	Percentage of control points with				
			$E < 1$	$1 \leq E < 2$	$2 \leq E < 3$	$3 \leq E < 4$	$E \geq 4$
Bilinear	53	2.34	92.45	5.66	1.89	0.00	0.00
Translation	17	0.60	100.00	0.00	0.00	0.00	0.00
Rotation	45	4.22	95.56	0.00	2.22	0.00	2.22
Scaling	65	2.32	89.23	9.23	1.54	0.00	0.00
TPS	39	2.87	84.62	12.82	2.56	0.00	0.00
Sinusoid	39	4.16	69.23	17.95	7.69	2.56	2.56

Table 5.1: *Column 1:* Type of distortion. *Column 2:* N , the number of control points obtained using AUTOCP. *Column 3:* Largest error in pixels. *Column 4:* Distribution of control points over different error (E) ranges measured in pixels.

Table (5.1) summarizes the results obtained using AUTOCP for the 6 types of distortions. The number of control points obtained in each case, as well as the largest error in pixels, and the percentage of control points over different E ranges is given. Results are given only for the final step of AUTOCP and not for the intermediate iterations. It is clear that while AUTOCP yields less number of control points than UICP, the accuracy of points is much better (compare Table (5.1) and Table (4.6)). This is because the rejection criteria in AUTOCP is more stringent.

5.9 Conclusions

AUTOCP, an automatic program used to extract control points from the startbyte-transformed images, was discussed in this chapter. Given the cleaned-up STs of the reference and target images, AUTOCP runs automatically. While it requires the specification of a few threshold values, these values are applicable for a wide range of distortions of different complexity. AUTOCP can be used singly as well as iteratively. More complex distortions require the iterative use of AUTOCP. Also, the iterative mode is indicated when one run of AUTOCP does not produce an adequate number

of control points. A comparison of the results obtained using AUTOCP and UICP shows that while AUTOCP yields less number of points as compared to UICP, these points are more accurate because of the stringent rejection criteria imposed within AUTOCP. The next chapter discusses image registration and compares results obtained using UICP and AUTOCP.

Chapter 6

IMAGE REGISTRATION

6.1 Introduction

The previous two chapters discussed how control points could be obtained manually using UICP and automatically using AUTOCP. We shall now compare the results obtained using UICP and AUTOCP.

At this stage, we would like to indicate how much time various stages of the manual and automatic registration procedures take. The times indicated below are CPU times for a Sparc 20 machine.

For both procedures, the starbyte transformation has to be calculated first, followed by the binary open and close operation to clean up individual slices. For a 128 x 128 image, we evaluated that the starbyte transformation with protocol 3, using a 9 x 9 neighborhood size takes about 8.5 seconds to calculate. Cleaning of individual slices using the binary morphological open and close operation (OC) and reassembling them takes 48.5 seconds. This is because the code [81] is not computationally fast and while there are a number of schemes for speeding the operators these have not been implemented [90]. Individual slices can be cleaned in parallel and independently of each other to speed up the process. While it is possible to speed up the process using grey scale OC operations [40], we found that there were a large number of isolated pixels remaining in the individual slices after the operation. Also, the regions boundaries were still rough.

After the OC operation, the manual and the automatic procedures are separate. For the manual procedure, selection of regions by a user takes approximately 15 to 20 minutes for a 4 bit protocol. Following this, the regions have to be grown on all slices, and their centroids calculated. This procedure takes a total time of about 1.4 seconds.

For the automatic procedure, region growing and preparation of the 4-element parameter list for the reference and target images takes about 2.2 seconds. The actual matching of the feature vectors takes about 0.0146 seconds.

Finally, the calculation of the unwarped image using TPS equations for the manual and automatic procedure take about 14.8112 seconds. This time can also be reduced by using fast evaluation algorithms for TPSs [91], [9].

For the automatic procedure, the total time from calculation of the starbyte transform to the unwarped image would be about 74.0 seconds, where the major portion of the time is used up for the OC operation. For the iterative mode of the automatic procedure, this would be the time for one iteration.

While code optimization was not the primary goal here, it may be possible to reduce all computation times by optimizing the code further.

It would also be appropriate at this point to present the operations required for image registration using UICP and AUTOCP in the form of flow charts for easy readability.

Figure 6.1 and Figure 6.2 show the various stages of the starbyte registration algorithm using UICP and AUTOCP respectively as flow charts.

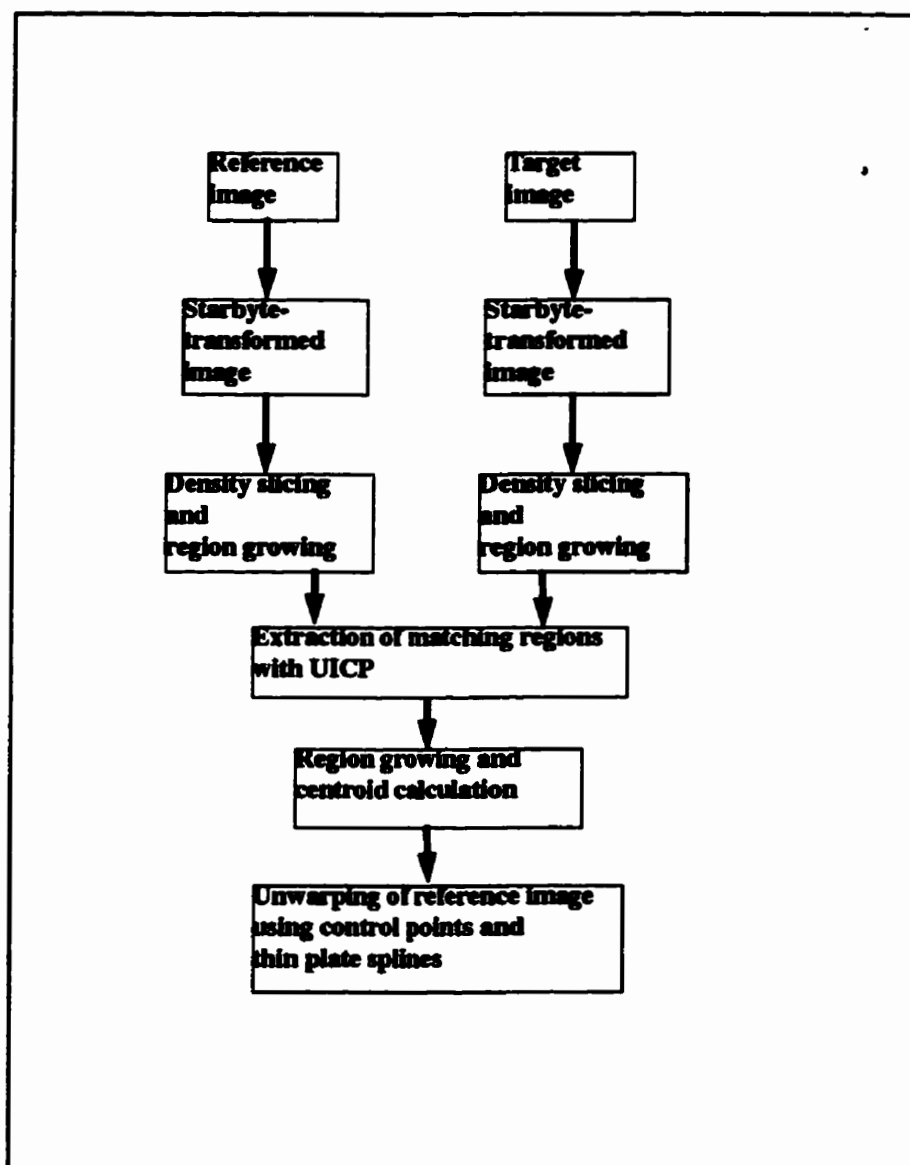


Figure 6.1: Flow chart showing various stages of the starbyte image registration algorithm using UICP.

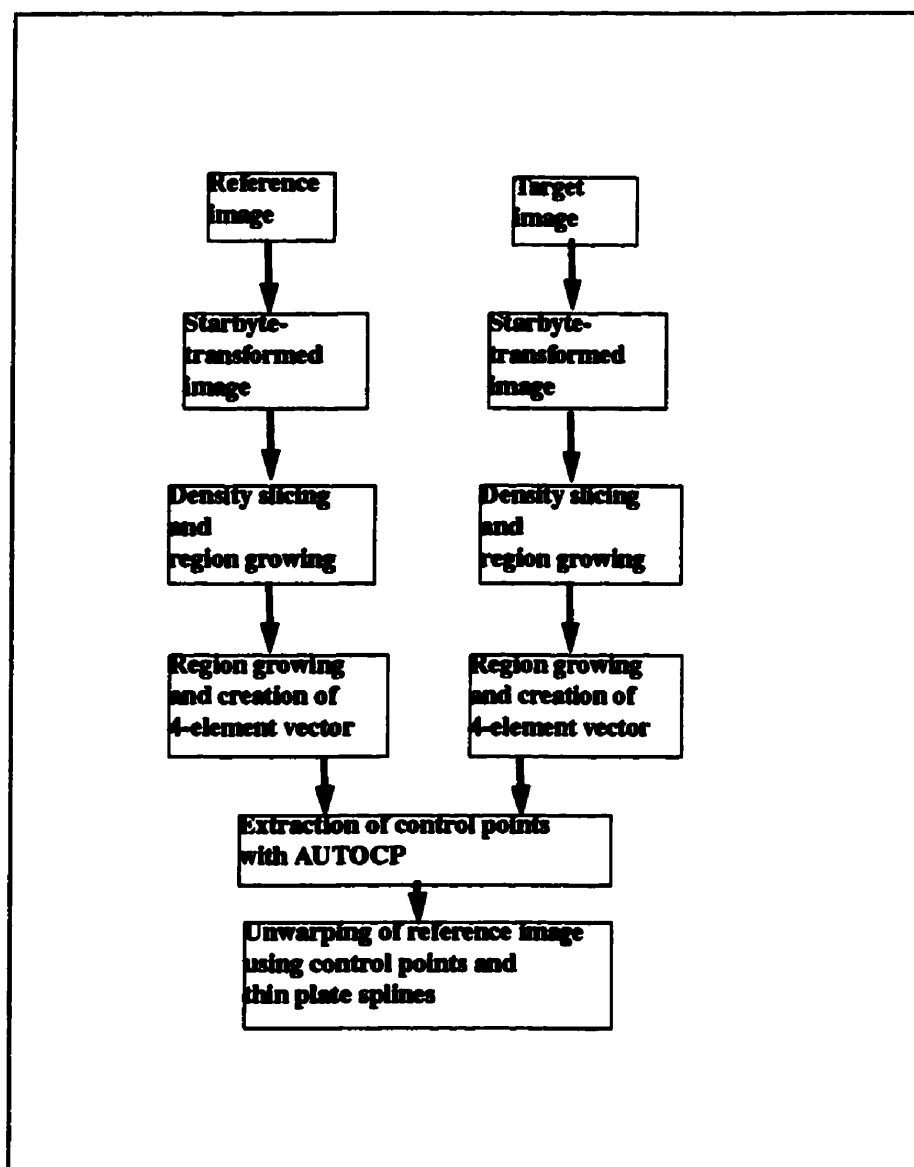


Figure 6.2: Flow chart showing various stages of the starbyte image registration algorithm using AUTOCAP.

For the 6 pairs of reference and target images corresponding to the six types of distortions considered, we shall now present the unwarped images obtained using UICP and AUTOCP. We shall also present the difference images between the target and reference, as well as between the target and the unwarped images. Since these difference images will be normalized to the same range for the sake of display, and since border effects will affect this normalization, we will present the difference images computed over a 100 x 100 matrix in the interior of the 128 x 128 images. Grid diagrams corresponding to the final registration for the both the manual and automatic case will be presented. Even though these grid diagrams were already presented in the previous chapters, we shall again present them to enable visual comparison of the results for the manual and automatic case. We shall present the results for each of the six transformations separately.

6.2 Bilinear distortion

Figures 6.3a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. These unwarped images and the ones shown for the other examples, were created by using the control points obtained through UICP and AUTOCP and TPS-unwarping. As can be seen, the unwarped images are close to the target image. Figure 6.4a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. It is clear that the misregistration error in Figure 6.4a, has been largely removed in Figure 6.4b and in Figure 6.4c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.5a to c shows the registration achieved as a grid diagram. Figure 6.5a shows the actual distortion between the given images.

Figure 6.5b shows the registration achieved with UICP, while Figure 6.5c shows the registration achieved with AUTOCP. Registration is almost perfect for both cases, though better for the automatic case.

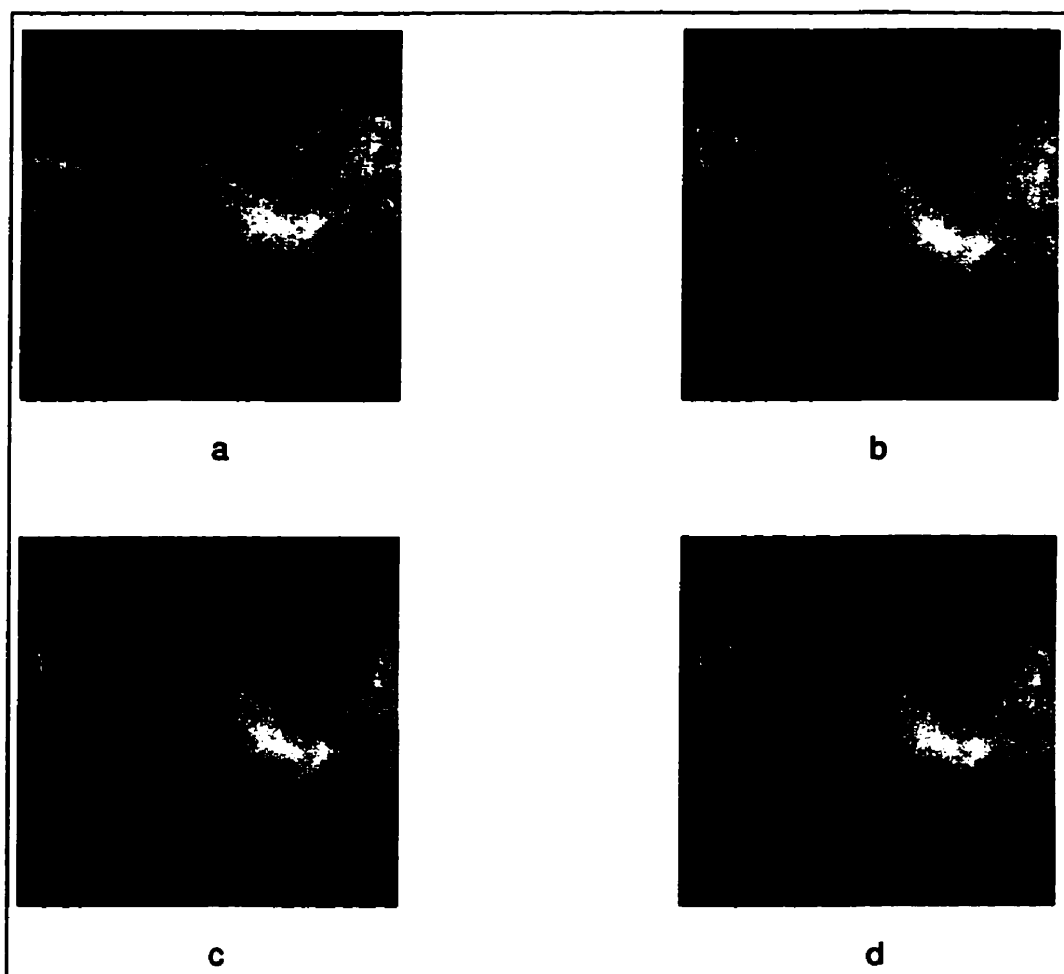


Figure 6.3: *a* to *d*: Reference, target and unwarped images obtained using UICP and AUTOCP respectively for the bilinear distortion.

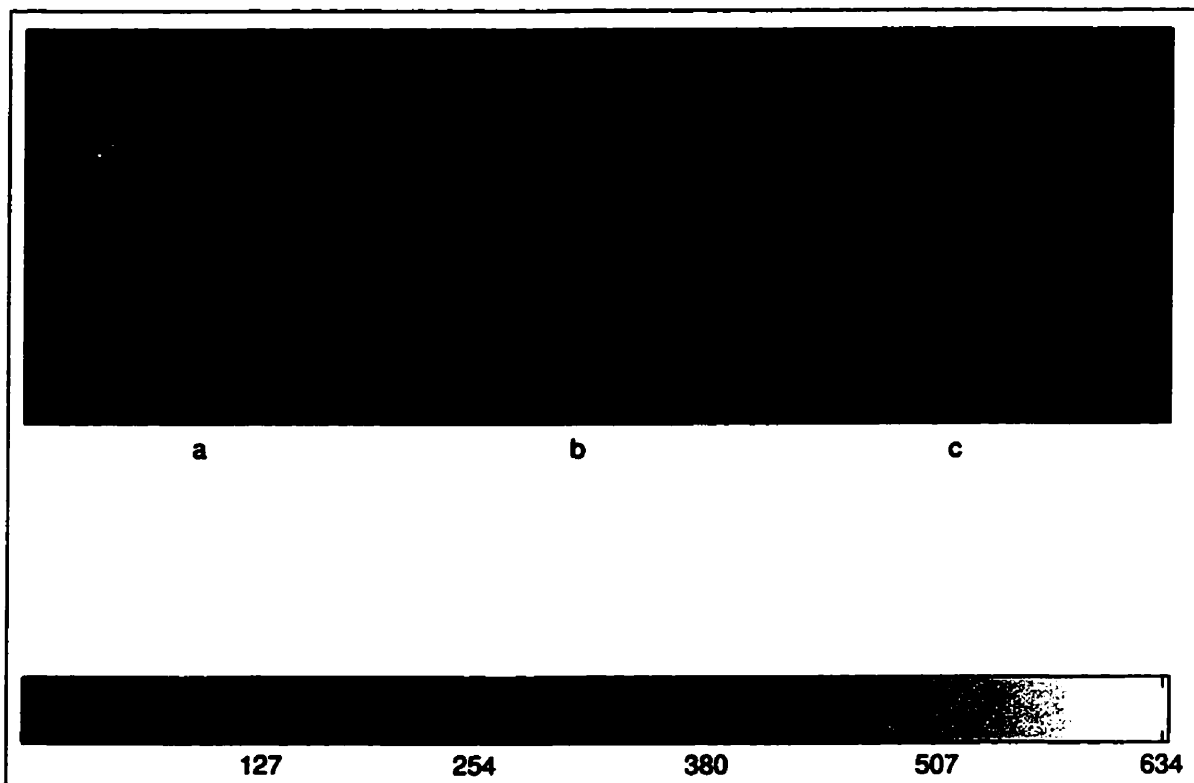


Figure 6.4: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the bilinear distortion. The grey scale bar below shows the range of values in the difference images.

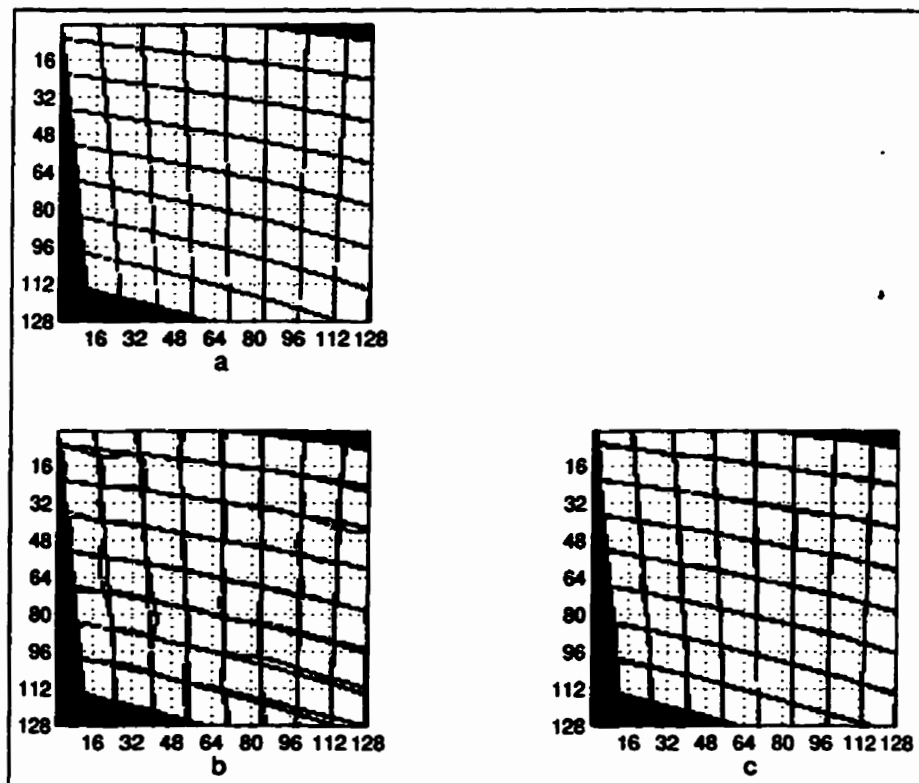


Figure 6.5: *a*: Bilinear distortion shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

6.3 Translation

Figures 6.6a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. The unwarped images are close to the target image. Figure 6.7a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. Again the misregistration error in Figure 6.7a, has been completely removed in Figure 6.7b and in Figure 6.7c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.8a shows the actual distortion between the given images. Figure 6.8b

shows the registration achieved with UICP, while Figure 6.8c shows the registration achieved with AUTOCP. Registration is perfect for both cases.

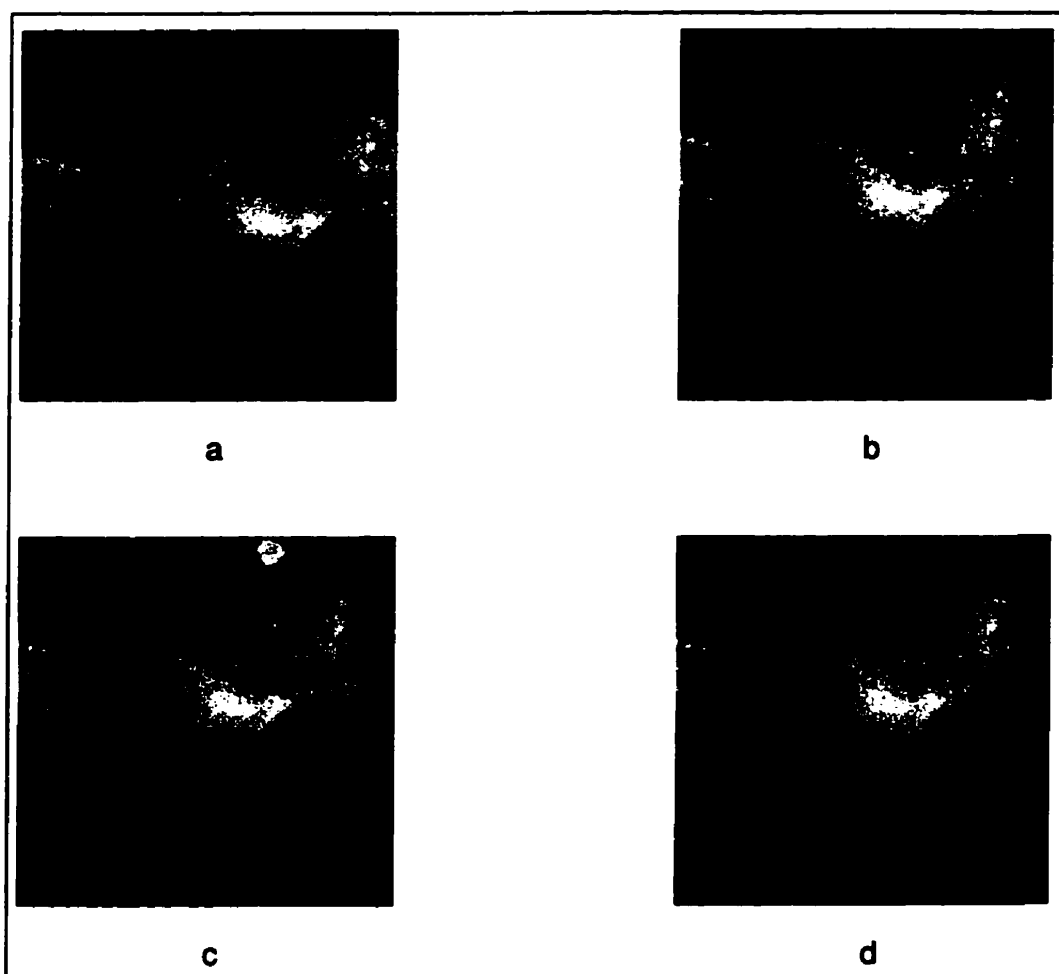


Figure 6.6: *a* to *d*: Reference, target and unwrapped images obtained using UICP and AUTOCP respectively for translation (8 pixels upwards and 10 pixels leftwards).

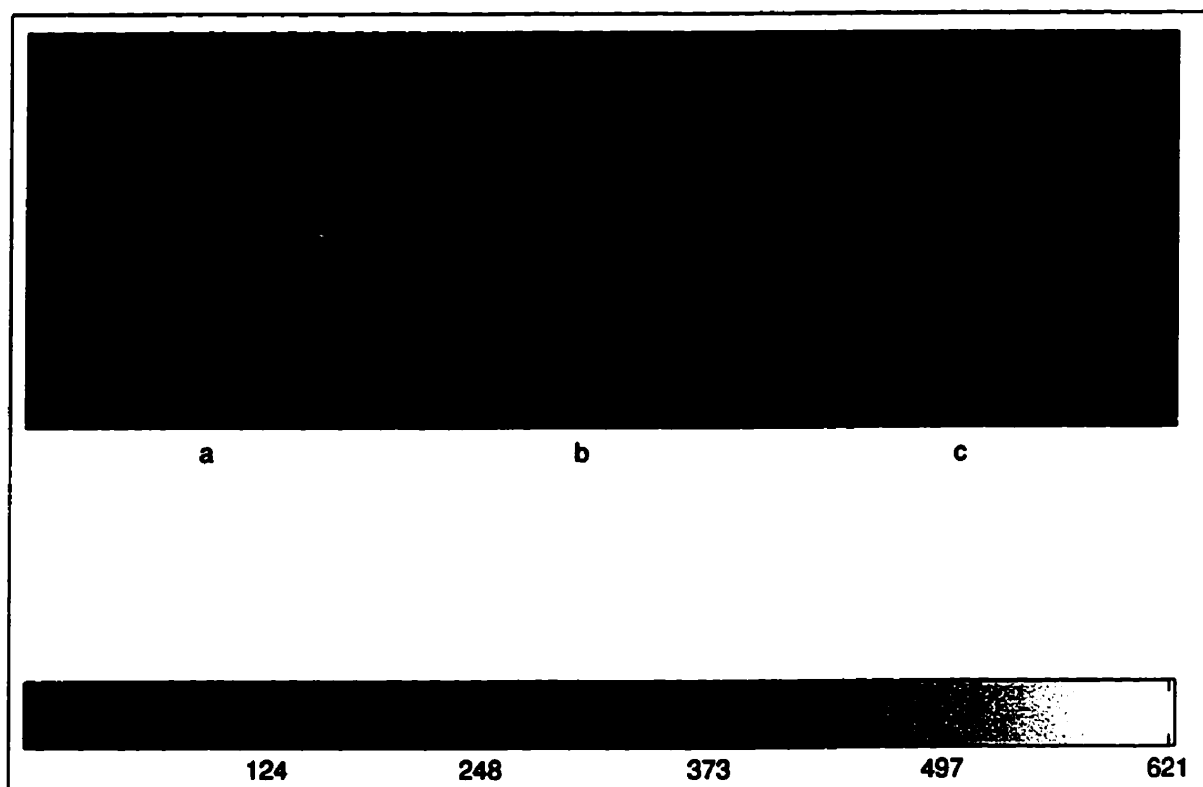


Figure 6.7: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for translation (8 pixels upwards and 10 pixels leftwards). The grey scale bar below shows the range of values in the difference images.

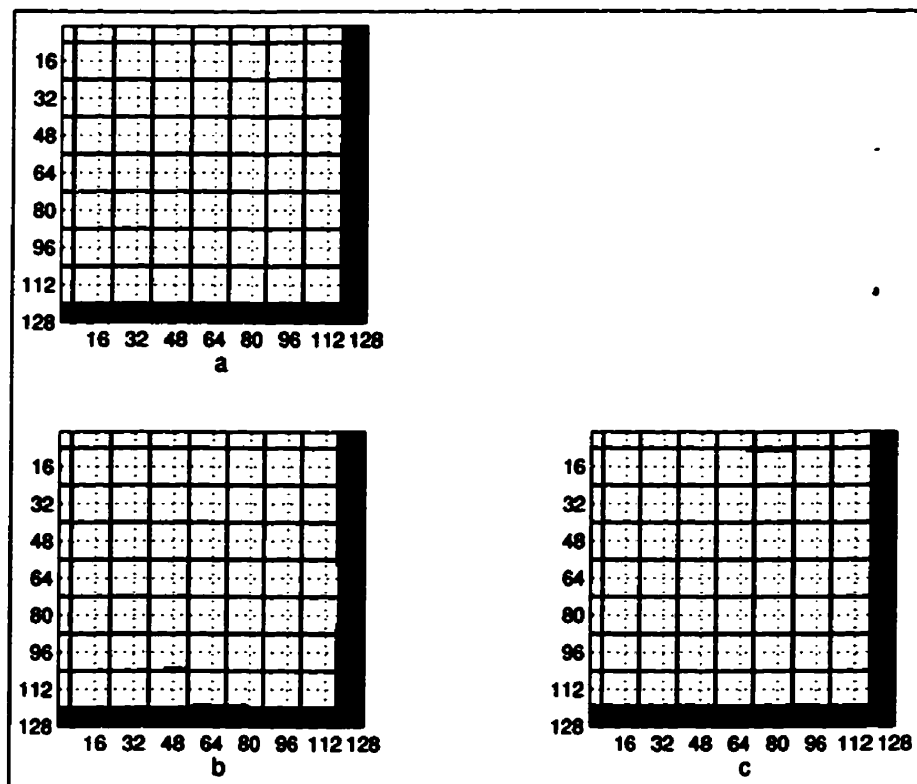


Figure 6.8: *a*: Translation (8 pixels upwards and 10 pixels leftwards) shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

6.4 Rotation

Figures 6.9a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. The unwarped images are close to the target image. Figure 6.10a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. Again the misregistration error in Figure 6.10a, has been largely removed in Figure 6.10b and in Figure 6.10c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.11a shows the actual distortion between the given images. Figure 6.11b

shows the registration achieved with UICP, while Figure 6.11c shows the registration achieved with AUTOCP. Registration is almost perfect for both cases and as in the bilinear case, better for the automatic case.

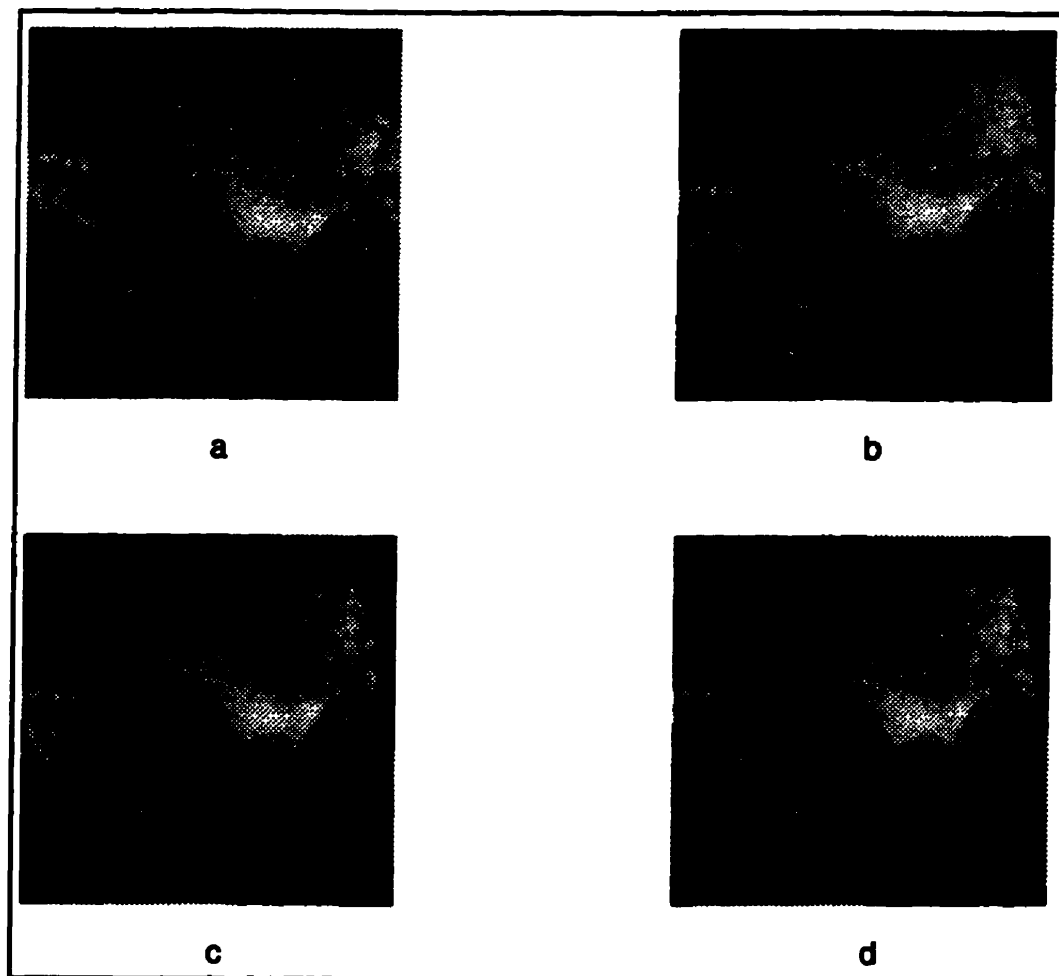


Figure 6.9: *a* to *d*: Reference, target and unwarped images obtained using UICP and AUTOCP respectively for rotation (10 deg counter-clockwise).

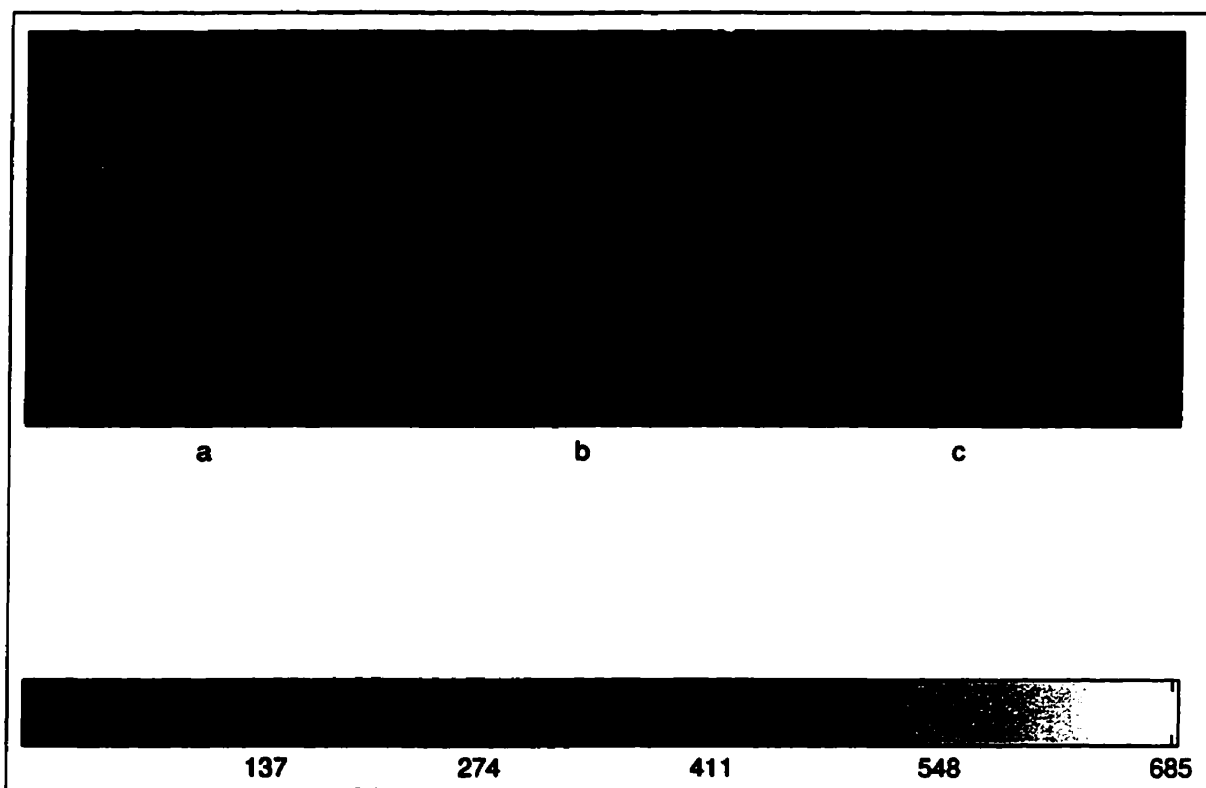


Figure 6.10: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for rotation (10 deg counter-clockwise). The grey scale bar below shows the range of values in the difference images.

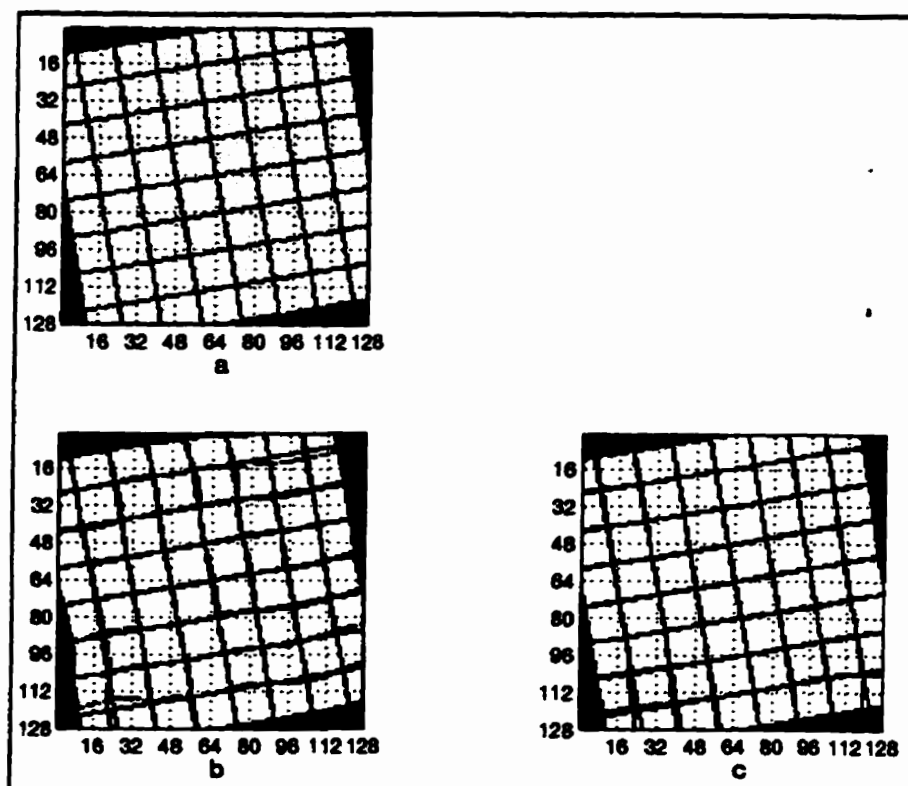


Figure 6.11: *a*: 10 deg counter-clockwise rotation shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

6.5 Scaling

Figures 6.12a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. The unwarped images are close to the target image. Figure 6.13a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. Again the most of the misregistration error in Figure 6.13a, is absent in Figure 6.13b and in Figure 6.13c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.14a shows the actual distortion between the given images. Figure 6.14b

shows the registration achieved with UICP, while Figure 6.14c shows the registration achieved with AUTOCP. Again, registration is good in both cases and better for the automatic case.

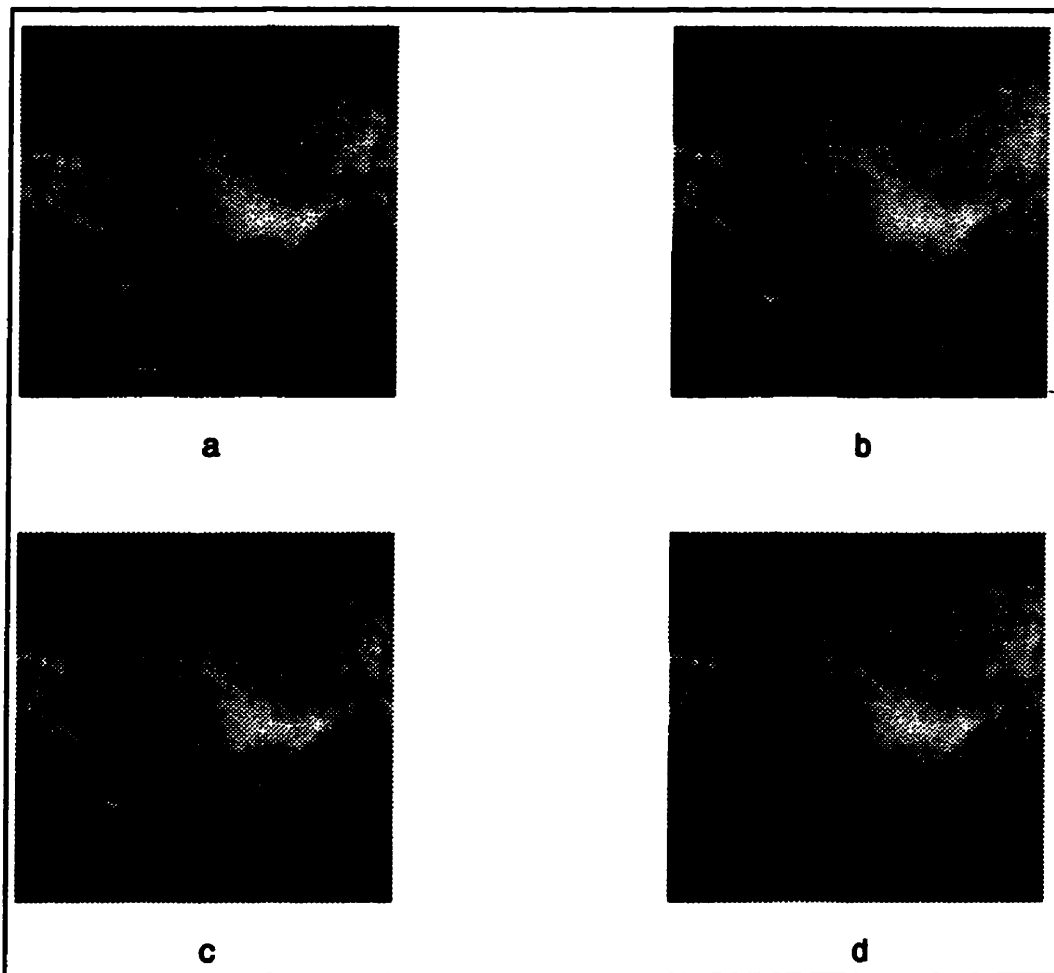


Figure 6.12: *a* to *d*: Reference, target and unwarped images obtained using UICP and AUTOCP respectively for scaling (10 % enlargement).

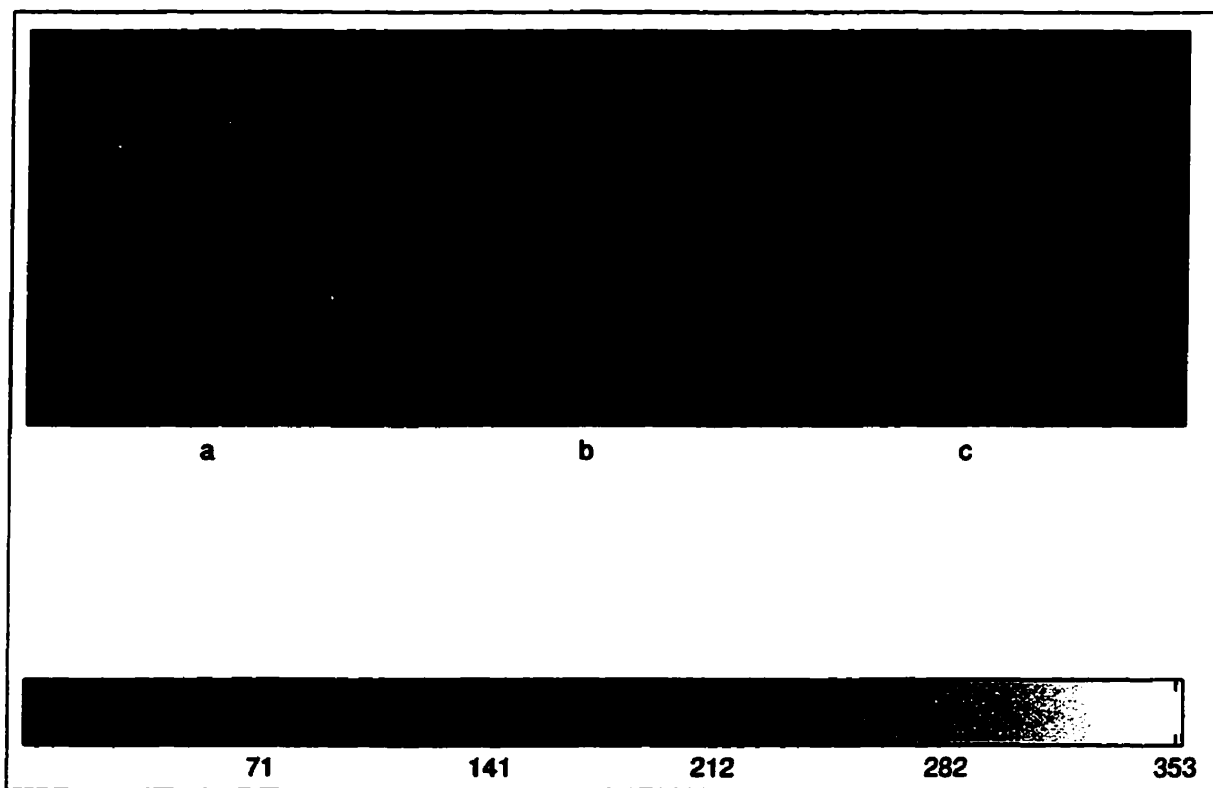


Figure 6.13: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for scaling (10 % enlargement). The grey scale bar below shows the range of values in the difference images.

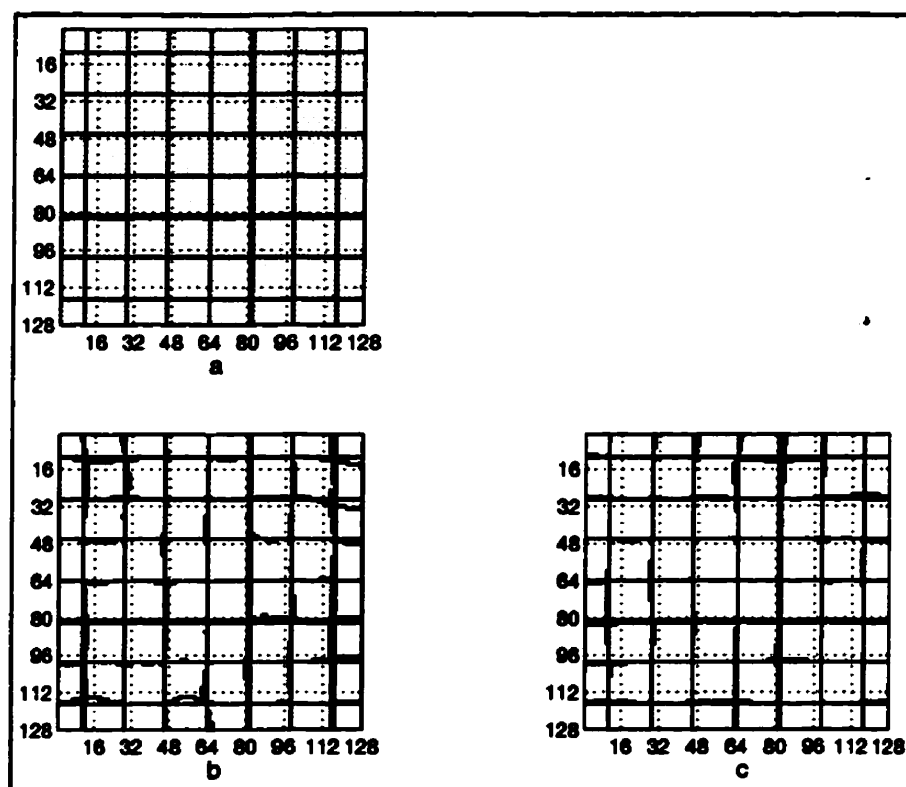


Figure 6.14: *a*: 10 % enlargement shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

6.6 TPS distortion

Figures 6.15a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. The unwarped images are close to the target image. Figure 6.16a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. Here, the misregistration error in Figure 6.16a is definitely less than that in Figure 6.16b and in Figure 6.16c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.17a shows the actual distortion between the given images. Figure 6.17b

shows the registration achieved with UICP, while Figure 6.17c shows the registration achieved with AUTOCP. While a large amount of registration has taken place, there is still some misalignment. This is because the original distortion is complicated. Registration is almost equal for the two cases though slightly better for the manual.

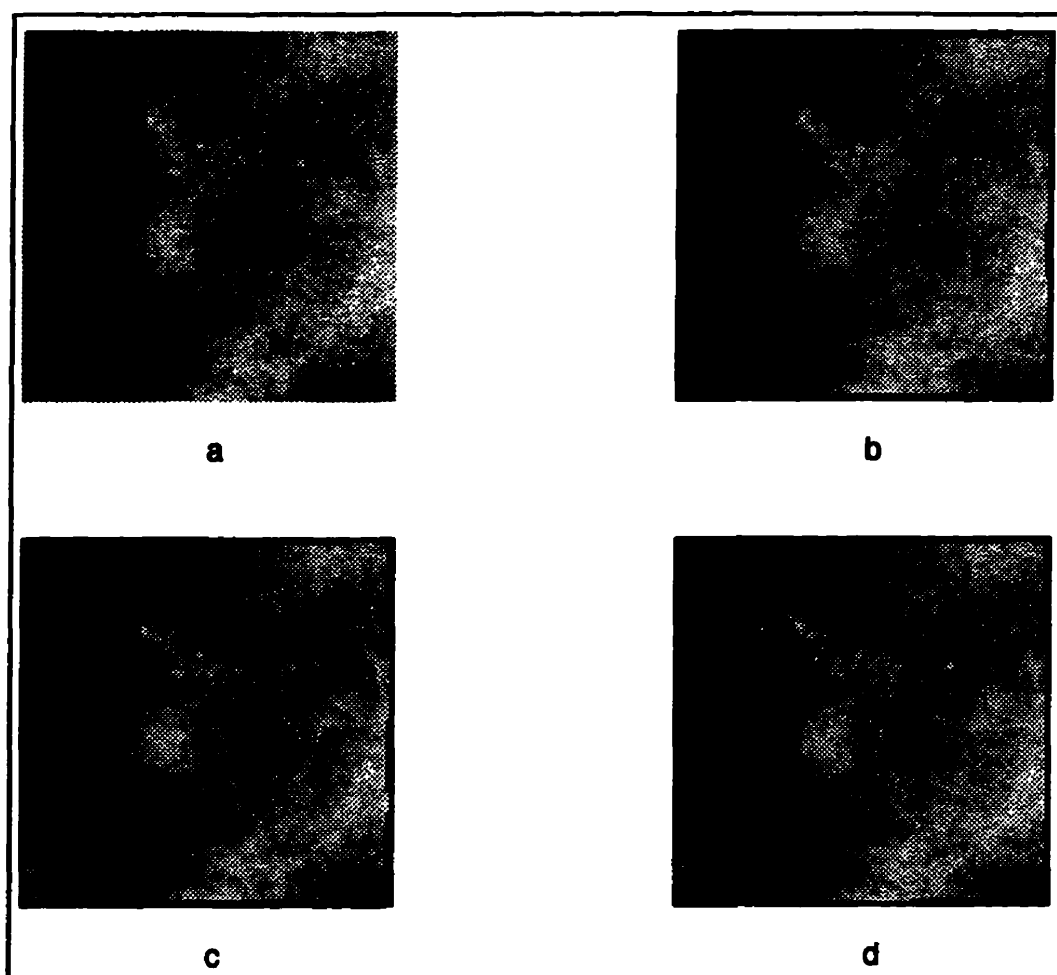


Figure 6.15: *a* to *d*: Reference, target and unwarped images obtained using UICP and AUTOCPP respectively for the TPS-distortion.

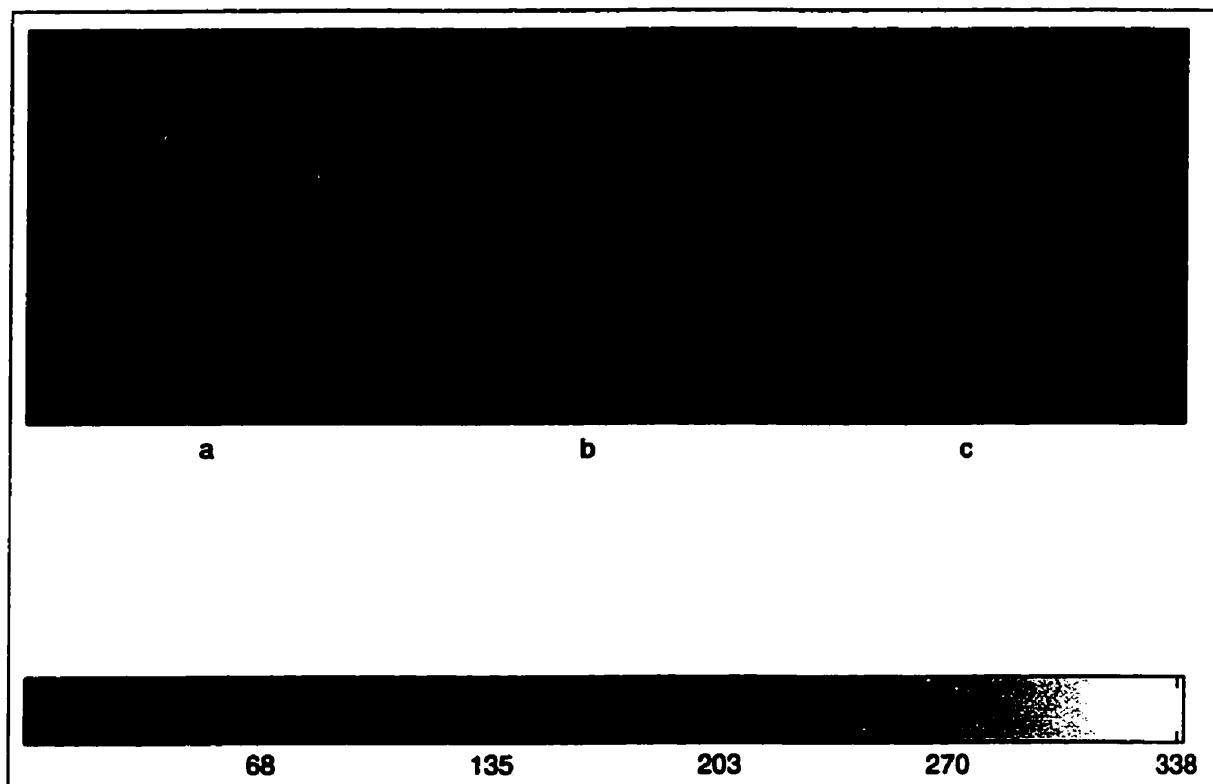


Figure 6.16: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the TPS-distortion. The grey scale bar below shows the range of values in the difference images.

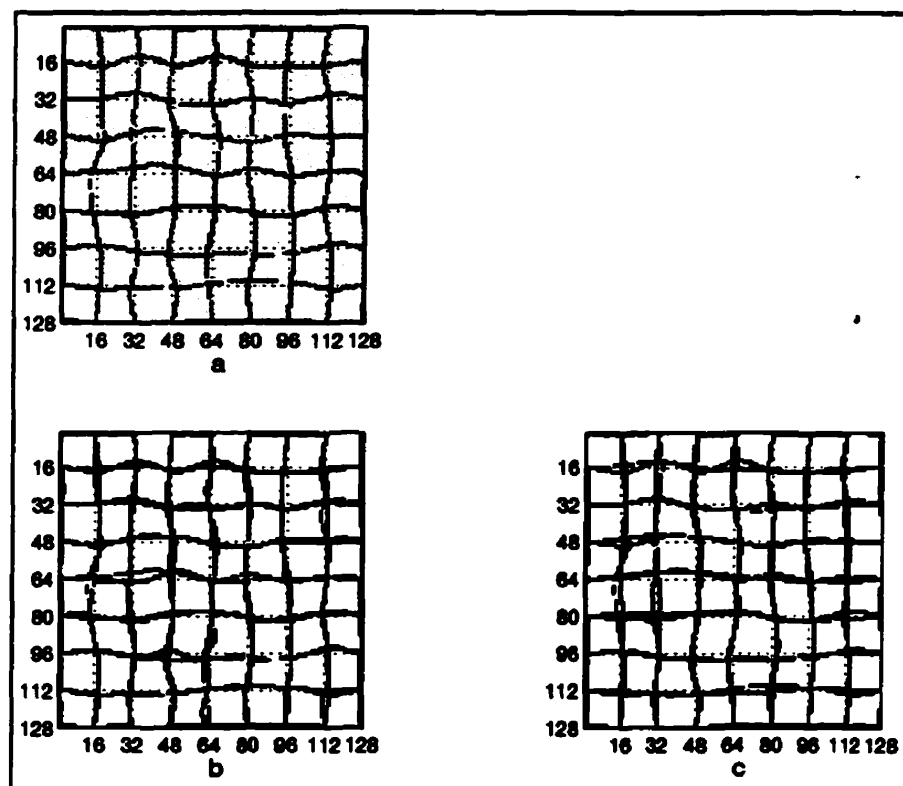


Figure 6.17: *a*: TPS-distortion shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

6.7 Sinusoidal distortion

Figures 6.18a to d show the reference, target, and the unwarped images obtained using UICP and AUTOCP respectively. The unwarped images are close to the target image. Figure 6.19a to c shows the difference images between the target and the reference, target and the unwarped image for the manual case, and the target and the unwarped image for the automatic case respectively. Here too, the misregistration error in Figure 6.19a has been largely removed in Figure 6.19b and in Figure 6.19c. The range of values in the difference images is shown below the images, as a grey scale bar. Figure 6.20a shows the actual distortion between the given images. Figure 6.20b

shows the registration achieved with UICP, while Figure 6.20c shows the registration achieved with AUTOCP. Registration is almost equal for both cases, though better for the manual.

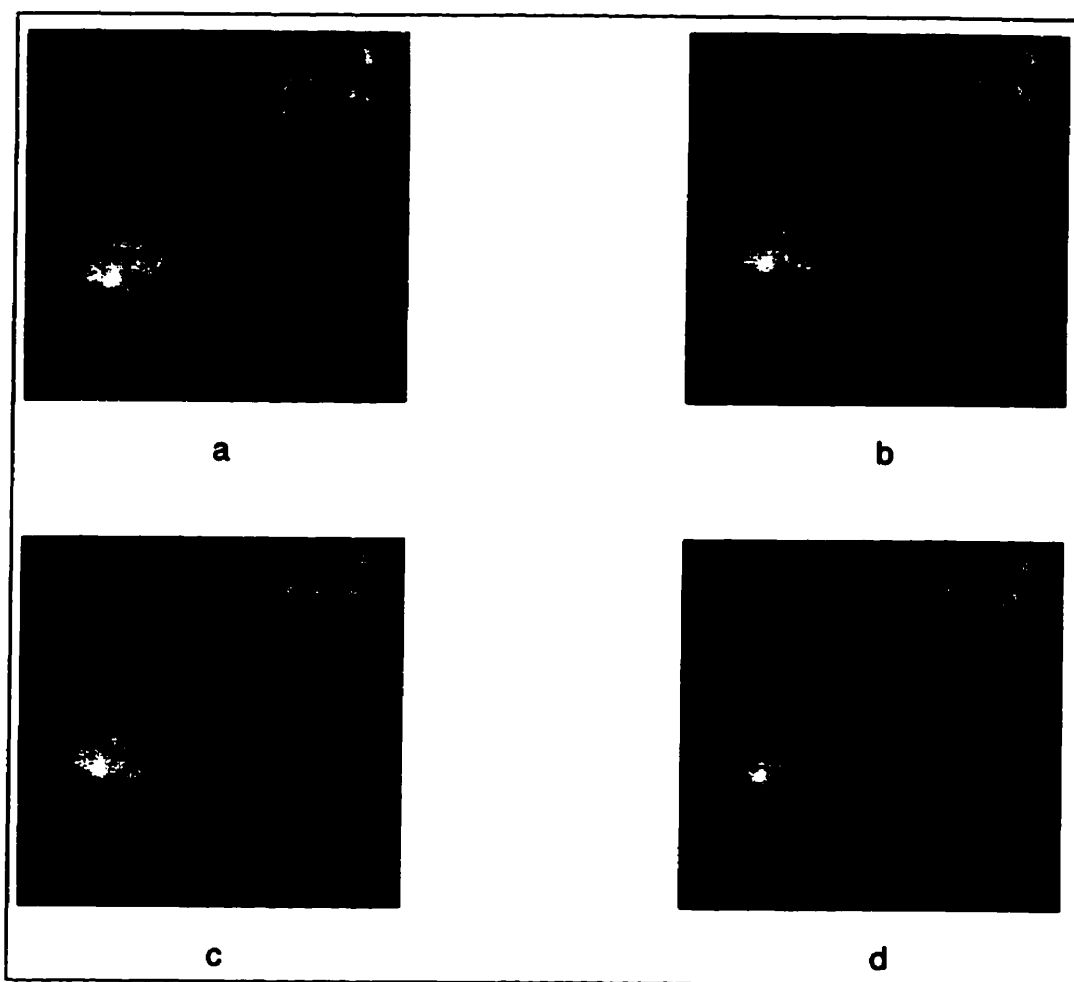


Figure 6.18: *a* to *d*: Reference, target and unwarped images obtained using UICP and AUTOCP respectively for the sinusoidal distortion.

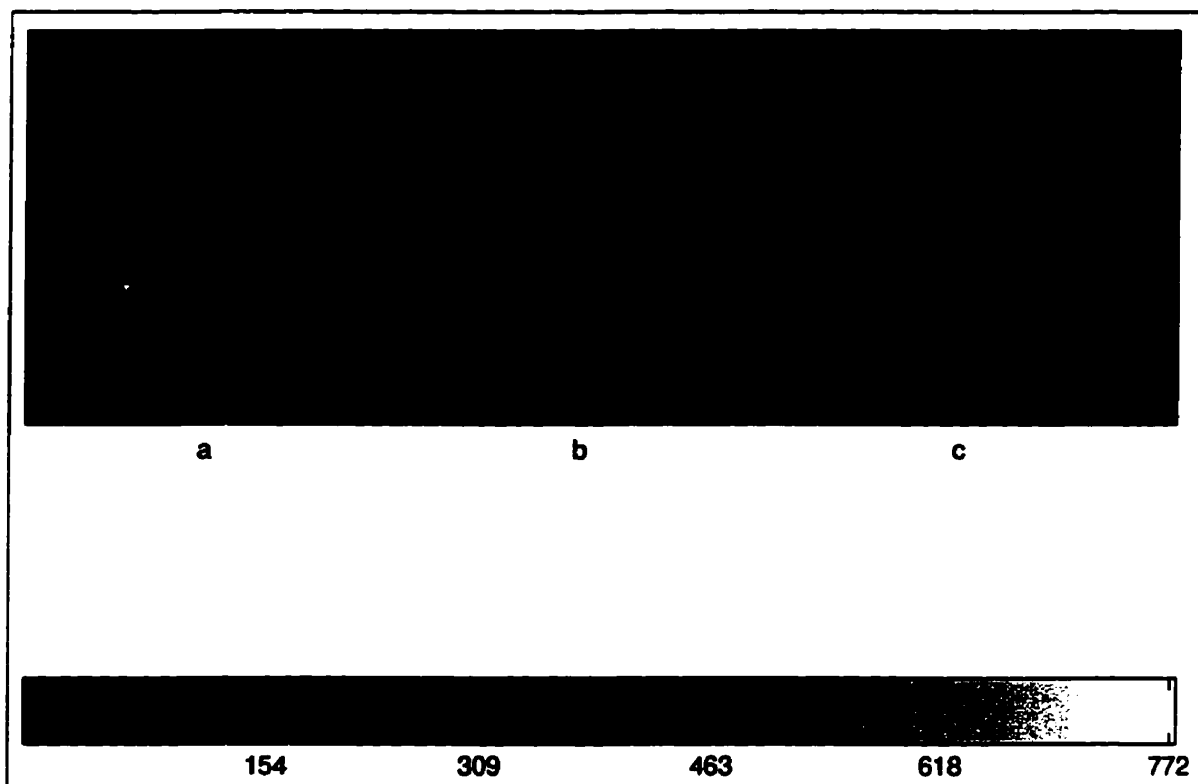


Figure 6.19: *a* to *c*: Difference images between the target and the reference, and the target and the unwarped images for the manual and automatic case for the sinusoidal distortion. The grey scale bar below shows the range of values in the difference images.

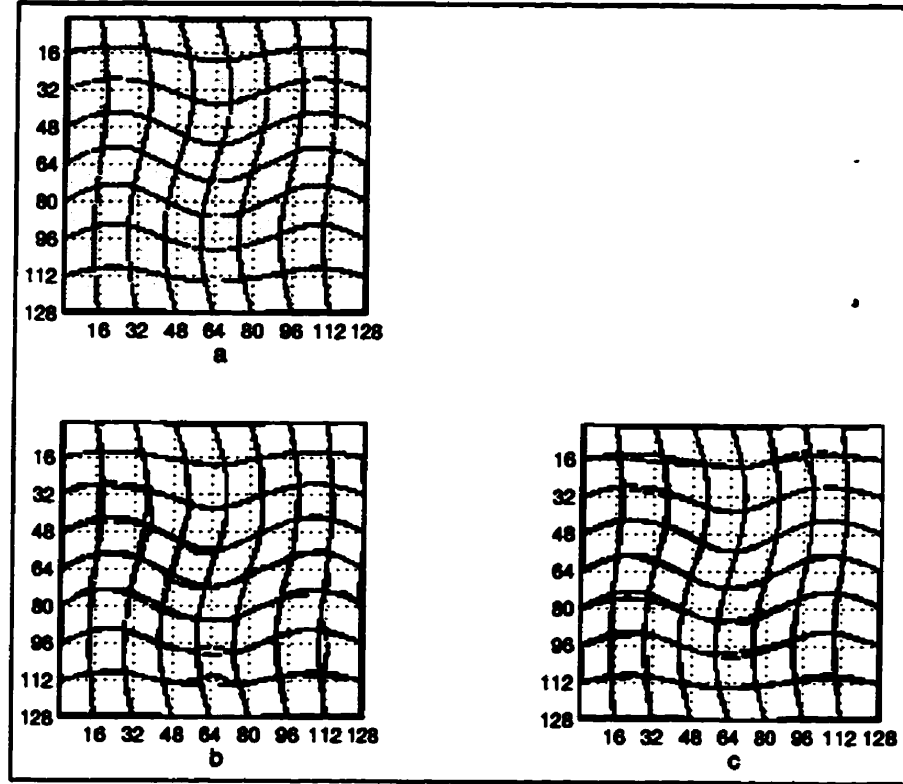


Figure 6.20: *a*: Sinusoidal distortion shown as a grid diagram. *b*: Registration achieved with UICP. *c*: Registration achieved with AUTOCP.

Registration was also evaluated numerically by calculating an error measure G , equal to the root mean square error per pixel between the target and the reference, as well as the target and the unwarped images for both cases. G between two images I_1 and I_2 is given by:

$$G = \frac{1}{n^2} \sqrt{\sum_{i=0}^n \sum_{j=0}^n \left(\frac{I_1(i,j)}{\bar{I}_1} - \frac{I_2(i,j)}{\bar{I}_2} \right)^2} \quad (6.1)$$

where n is the size of the images. Since the images are square, the number of rows and columns in the images are equal. The symbol G has been used because this error indicates the error in grey level between the images. \bar{I}_1 is the mean value of image I_1 while \bar{I}_2 is the mean value of image I_2 . Column 2 of Table (6.1) shows N , the number

Type of Distortion	N		G		
	M	A	U	M	A
Bilinear	89	53	0.0024	0.0010	0.0009
Translation	130	17	0.0035	0.0008	0.0007
Rotation	60	45	0.0024	0.0013	0.0008
Scaling	95	65	0.0006	0.0002	0.0002
TPS	143	39	0.0017	0.0008	0.0009
Sinusoid	94	39	0.0016	0.0006	0.0006

Table 6.1: *Column 1:* Type of distortion. *Column 2:* N , the number of control points obtained using UICP (M) and AUTOCP (A). *Column 3:* G , the root mean square error per pixel between the target and the reference (U), target and the final unwarped images for the manual (M) and automatic (A) case.

of control points obtained using UICP and AUTOCP respectively for the six types of distortions. The values shown here are those obtained at the final iteration, wherever the iterative mode was used. Values for G between the target and the reference, as well as the target and the unwarped images for both cases are presented in column 3.

It can be seen from the values in column 3 of Table (6.1) that in all cases, the error was much higher for the unregistered case than for the other two, showing that registration has taken place. For the simpler distortions, it is seen that AUTOCP has performed better than UICP. This also agrees with visual observation of the difference images. For the TPS and sinusoidal distortion, UICP has done better than AUTOCP. This is probably because more control points are required to describe these complicated distortions and since UICP has more control points than AUTOCP, for these distortions, G is less for UICP than AUTOCP.

We also evaluated the performance of AUTOCP and UICP using E , the registration error magnitude introduced in the earlier chapters. E_T , the average geometrical error for all pixels, is the average of E values considered over all pixels in the image.

Type of Distortion	N	Manual		N	Automatic	
		Average E_{CP}	E_T		Average E_{CP}	E_T
Bilinear	89	1.0063	1.2924	53	0.4540	0.5898
Translation	130	0.0148	0.0707	17	0.0427	0.0907
Rotation	60	0.9889	1.1980	45	0.4903	0.6590
Scaling	95	0.7078	0.8442	65	0.4477	0.5280
TPS	143	0.8625	1.0096	39	0.3731	1.1195
Sinusoid	94	0.6760	0.9305	39	0.4996	0.9439

Table 6.2: *Column 1:* Type of distortion. *Column 2 (Manual): Subcolumn 1:* N , the number of control points obtained using UICP. *Subcolumn 2:* E_{CP} , average geometrical error calculated control points. *Subcolumn 3:* E_T , average geometrical error calculated over all pixels. *Column 3 (Automatic): Subcolumn 1:* N , the number of control points obtained using AUTOCP. *Subcolumn 2:* E_{CP} , average geometrical error calculated over control points. *Subcolumn 3:* E_T , average geometrical error calculated over all pixels.

Referring to points in the target image by (x, y) , their *calculated* points in the reference image by (x^*, y^*) , and the *actual* points in the reference image by (\hat{x}, \hat{y}) , E_T is given by:

$$E_T = \frac{1}{n^2} \sum_{i=0}^{n^2} \sqrt{(x_i^* - \hat{x}_i)^2 + (y_i^* - \hat{y}_i)^2} \quad (6.2)$$

The summation in the above equation is over all pixels in the image. We also evaluated E_{CP} , the average geometrical error for the control points. E_{CP} is the average of E values considering only the control points. Hence the equation for E_{CP} is similar to that for E_T except that the summation is only over the control points. Table (6.2) gives values for E_{CP} and E_T for UICP and AUTOCP. E_T in all cases is seen to be of the order of a pixel. This shows that accurate registration has taken place. It is observed that in all cases, E_T is higher than E_{CP} . This is probably because the calculation for E_T also considers the boundary where some amount of

misregistration is still present. E_{CP} and E_T values are well correlated for the simpler distortions. However, it is observed that for the TPS and sinusoidal distortion, there is a wide difference between the two values, especially in the automatic case. This may be because these are more complicated distortions, and even though the control points extracted are accurate in the automatic case, they are not sufficient to describe the complete transformation. Hence regions which lack control points are registered poorly causing an increase in E_T .

An important difference between AUTOCP and UICP is that while control points for much more severe transformations can be extracted with UICP, the same cannot be said of AUTOCP. The use of the 4 parameters within AUTOCP (discussed in the previous chapter) limits the range of distortions which can be successfully used with AUTOCP. For example, a very large transformation would not work with AUTOCP because of the use of the X - and Y - location of the centroids in the parameter list. Similarly a very large scaling would not work because of the use of the number of pixels in a region in the parameter list. However these would work with UICP, because selection of matching regions is done by eye.

6.8 Conclusions

In this chapter, we have presented and compared the results obtained using UICP and AUTOCP. For the six types of distortions studied, both UICP and AUTOCP gave accurate control points. These control points were used with a TPS-unwarping procedure to obtain unwarped images. The unwarping process was demonstrated using grid diagrams. Registration achieved in all cases was good as shown through the difference images and the error values. For the more complicated distortions, for e.g. the TPS-distortion, the difference images showed some amount of misalignment,

but this was far less than the misalignment between the reference and target images.

In the next two chapters, the performance of the starbyte transformation is evaluated for additive noise and the presence of simulated tumors in the target image.

Chapter 7

CONTROL POINT ACCURACY WITH ADDED NOISE

7.1 Introduction

In the previous chapters, image registration with the starbyte transformation was discussed in detail, using simulated distortions. We showed that using the starbyte transformation, control points can be extracted accurately. Since accurate matching depends on the extraction of accurate control points, accurate matching is achievable with the starbyte transformation. While control points extracted using the starbyte transformation are accurate, we would like to study, in a controlled way, how this accuracy is affected when noise is added to the images. This is the focus of this chapter.

As in the previous chapters, we shall perform the experiments on portions of mammograms. Hence we shall select a noise model most appropriate for mammograms. However for noise studies with breast images from other modalities, noise models appropriate to the particular modality will have to be used.

The noise in X-ray images comes from three different sources: quantum, film and screen [14], [7]. While the film noise has a Gaussian distribution, the quantum and screen noise have Poisson distributions. However, because of the Central Limit Theorem, assuming that the total noise in an X-ray image is Gaussian is not a bad approximation. Hence, we have assumed here that the noise in a breast image has

a Gaussian distribution. In order to study the effect of noise on the accuracy of control points, we added noise with a Gaussian distribution to the reference and target image pairs for two of the simulated distortions studied previously, namely the bilinear and sinusoidal distortions. These two distortions are selected because they represent distortions of large magnitude and are also reasonably complicated.

We have estimated approximately that the standard deviation of noise in a mammogram is approximately about 4% of the mean density value of the mammogram. This was estimated empirically in the following way using reasonably constant portions of different mammograms from the Nijmegen database using an empirical method suggested to us by Dr Nico karssemeijer [64].

For each constant portion selected, a horizontal difference matrix and a vertical difference matrix were created by tabulating differences between a pixel and its horizontal right neighbor and vertical bottom neighbor in the original portion respectively. These difference matrices gave an indication of the typical statistical fluctuation present in mammograms over constant portions. Since these portions were reasonably constant, we assumed that the anatomical fluctuation due to tissue inhomogeneity was small compared to the radiographic noise.

The standard deviation of these difference matrices is directly related to the standard deviation of the noise in these constant portions by a factor of 0.72 approximately. This factor was arrived at empirically in the following way:

We created a random noise matrix having Gaussian distribution with zero mean using a standard mathematical package like Matlab [76]. The standard deviation of this matrix was varied. For each value of the standard deviation, a horizontal and vertical difference matrix were created as described above. It was found that the standard deviation of the difference matrices was always approximately 0.72 times the standard deviation of the original random matrix. Hence, the standard deviation

of the actual noise in these constant portions would be the standard deviation of these difference matrices *divided by* 0.72. The average standard deviation of the difference matrices was about 3%. Thus the standard deviation of the noise in these constant portions was about 4%.

The number 0.72 had no correlation to the size of the random matrix. However it was found that this factor increased slightly for very small matrix sizes. As can be seen, if this number increases, the estimated standard deviation *decreases*. Hence we assumed the worst case value of 0.72, in order to arrive at our estimate.

This is not the ideal way of adding noise to the images because the noise added in this way is spatially uncorrelated to the image. However, the screen-film system responds non-linearly to x-ray exposure, and hence by adding noise in this way, we would overestimate the noise in low and high optical density regions [87], [14].

The experiment should be performed by adding noise to originally noise-free reference and target images. But, since in these cases, the target was created from the reference by an artificial distortion, noise (which is already present in the reference), is passed on to the target. But since a study of the accuracy of control points with addition of noise can be best done using simulated distortions (where we know the transformation relating the given pair of images), this noise study was performed by adding artificial noise to both the reference and target images. The noise was assumed to have a zero mean, while the standard deviation (SD) of the noise was varied as a percentage of the average of the mean density value of the reference and target images. Results are presented for 3 different noise levels. These three cases are where the SD of the noise is 1%, 5% and 10% of the average of the mean density value of the reference and target images.

This chapter should be viewed as a preliminary study which observes the effect of perturbations to the grey level (in the form of additive noise) on the performance of the

starbyte algorithm. Hence since noise is being used merely to create a perturbation to the grey level of the images, the actual noise model used is not really critical. Besides, the noise model will vary with the imaging modality.

We shall refer to the three noise levels as the 1%, the 5% and the 10% noise level. Empirically speaking, the 1% noise level represents a small perturbation to the grey level, the 5% and the 10% represent a medium and large perturbation respectively. Since the noise is additive, and since the SD selected is based on the average value, pixels of low grey value will have high noise. Also adding noise in this way, implies that several pixels in the image will have noise much higher than the SD level chosen.

It would seem intuitive that filtration has to be used at some stage to smooth the noise. As has been demonstrated earlier, if we use a larger neighborhood size for the ST, this implies calculating the starbyte values at each pixel by averaging over a larger number of pixels. Hence using a larger neighborhood size ST increases the amount of smoothing. Thus, we compare the performance at each noise level for the starbyte transformation using two different neighborhood sizes. We also compare these results to those obtained using direct filtration of the original images.

At each of the noise levels, the performance of the starbyte algorithm was evaluated for a 9×9 and 15×15 neighborhood size as well as for a 9×9 neighborhood size after the original images were filtered by a 5×5 moving average filter. Results are presented separately for the two distortions. AUTOCP was run iteratively on each case. UICP was run only once (i.e. not iteratively) on the cases with the highest noise level, i.e. 10%. For AUTOCP, we present the D and E vector plots only for the iteration giving the largest number of control points. The reference and target images for the bilinear distortion are Figure 2.2b and Figure 2.3b while the reference and target images for the sinusoidal distortion are Figure 2.2a and its sinusoidally-distorted version Figure 2.19c. Protocol 3 was used in both the cases.

7.2 Bilinear distortion

Figures 7.1a and b show the original reference and target images. Figures 7.1c and d, e and f, and g and h, show the reference-target pair with 1%, 5% and 10% noise added. Figures 7.2a and b show the STs of the reference-target pair using a 9 x 9 neighborhood size, while c and d show 15 x 15 STs of the reference and target. e and f show 9 x 9 STs of filtered reference and target images. Figures 7.2g to l, Figures 7.2m to r, and Figures 7.2s to x show the same for the reference-target pair with 1%, 5% and 10% noise added.

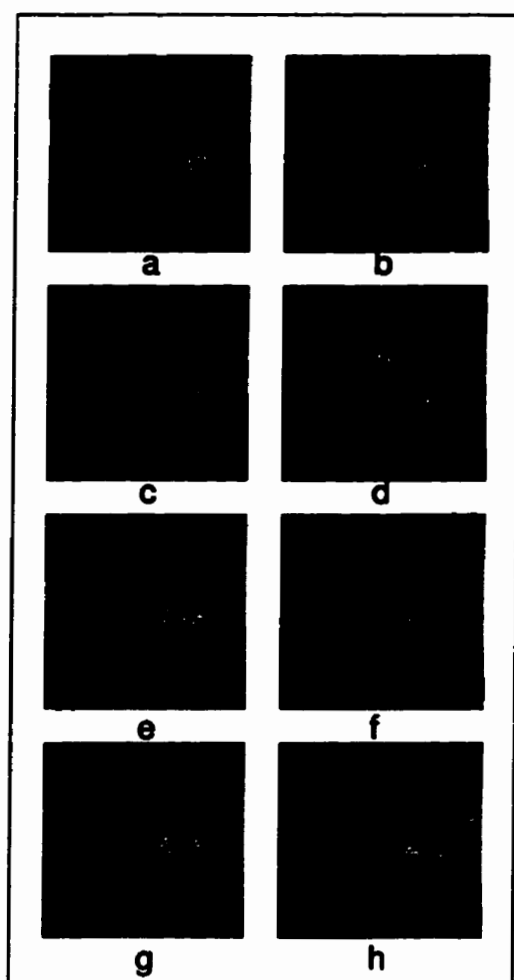


Figure 7.1: *a* and *b*: Reference and target images for the bilinear distortion. *c* and *d*: *a* and *b* with 1% noise. *e* and *f*: *a* and *b* with 5 % noise. *g* and *h*: *a* and *b* with 10% noise.

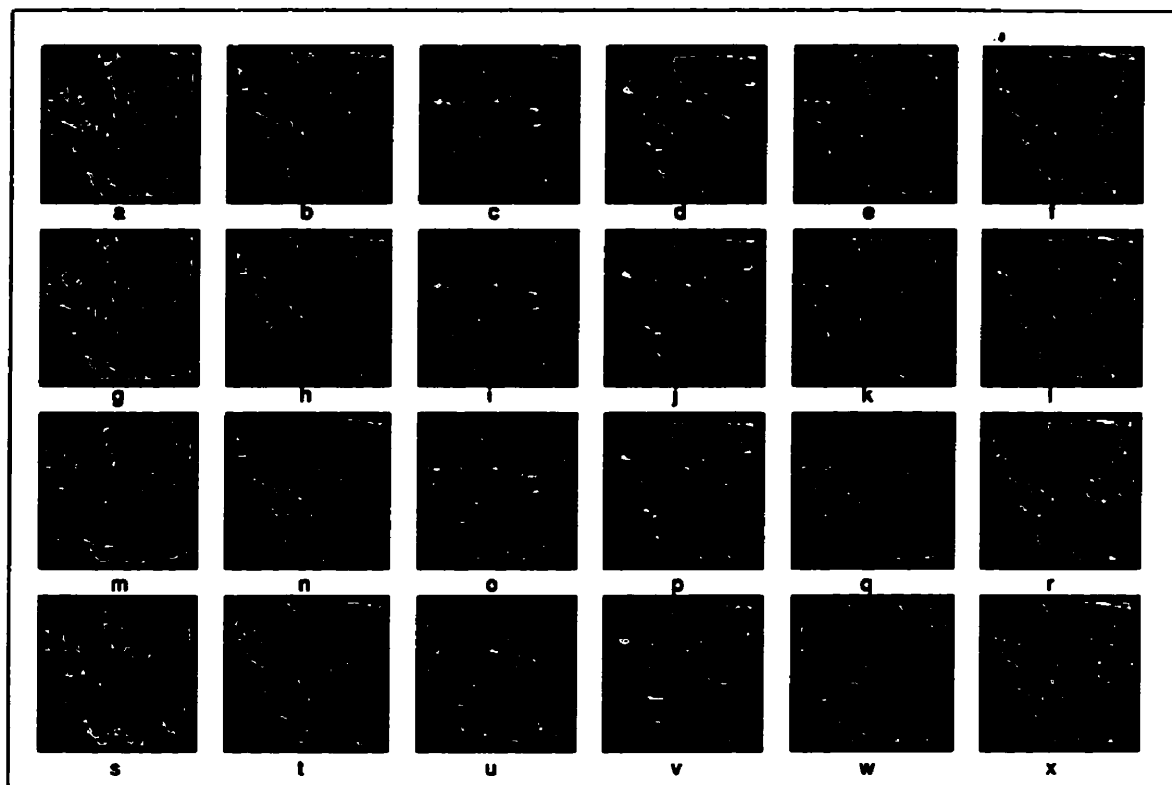


Figure 7.2: *a* to *d*: 9×9 and 15×15 STs of the reference-target pair for the bilinear distortion. *e* and *f*: 9×9 STs of filtered reference and target images. *g* to *l*: Corresponding ST images for the reference-target pair with 1% noise. *m* to *r*: Corresponding ST images for the reference-target pair with 5% noise. *s* to *x*: Corresponding ST images for the reference-target pair with 10% noise.

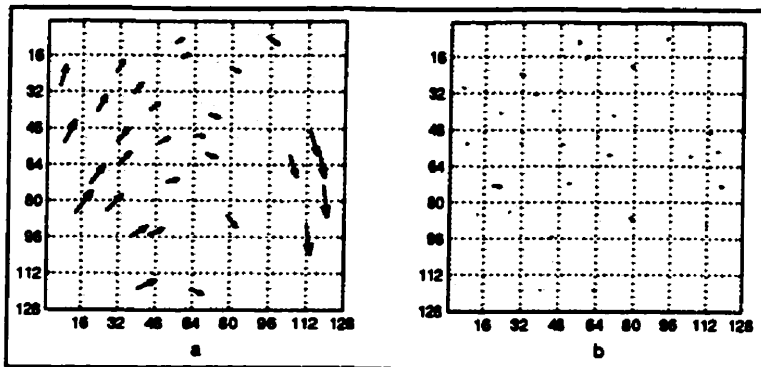


Figure 7.3: *a* and *b*: *D* and *E* for 1 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.

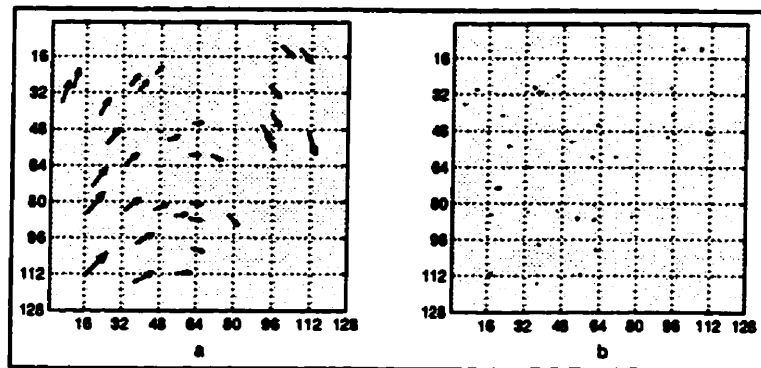


Figure 7.4: *a* and *b*: *D* and *E* for 1 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.

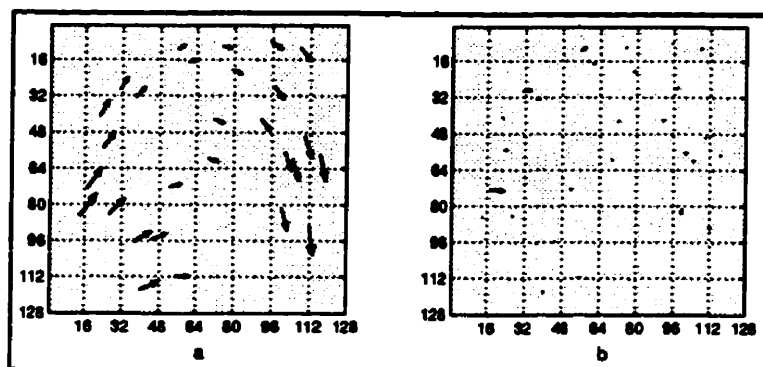


Figure 7.5: *a* and *b*: *D* and *E* for 1 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.

Noise Level	Method	I	Largest Error	N	Percentage of control points with				
					$E < 1$	$1 \leq E < 2$	$2 \leq E < 3$	$3 \leq E < 4$	$E \geq 4$
1 %	9 x 9	2	3.567	30	73.33	20.00	3.333	3.333	0.000
	15 x 15	2	2.166	32	78.12	15.62	6.250	0.000	0.000
	F 9 x 9	2	6.002	28	75.00	14.29	3.571	3.571	3.571
5 %	9 x 9	5	5.331	28	25.00	53.57	10.710	7.143	3.571
	15 x 15	4	3.690	22	36.36	31.82	18.180	13.640	0.000
	F 9 x 9	5	4.142	21	61.90	19.05	14.290	0.000	4.762
10 % (A)	9 x 9	3	15.890	12	25.00	25.00	25.000	0.000	25.000
	15 x 15	5	2.715	14	35.71	35.71	28.570	0.000	0.000
	F 9 x 9	5	8.571	15	26.67	40.00	0	13.330	20.000
10 % (M)	9 x 9	-	5.646	31	12.90	45.16	22.58	9.677	9.677
	15 x 15	-	7.626	40	27.50	27.50	15.000	12.500	17.500
	F 9 x 9	-	7.636	32	21.88	34.38	18.750	12.500	12.500

Table 7.1: *Column 1:* Noise Level. A and M stand for “Automatic” and “Manual”. *Column 2:* ST method used. F stands for “Filtered”. *Column 3:* I , number of iterations. *Column 4:* Largest error in pixels. *Column 5:* N , the number of control points obtained using the user-interactive program. *Column 6:* Break up of control points over different error (E) ranges measured in pixels.

Figure 7.3 and Figure 7.4 show the **D** and **E** diagrams after 2 iterations, for 1 % noise level using a 9 x 9 and a 15 x 15 neighborhood size ST, while Figure 7.5 shows the results of using a 9 x 9 neighborhood ST on filtered original images. As mentioned earlier the filtration used was a 5 x 5 moving average filter. For simplicity of notation, we shall refer to the case of applying a 9 x 9 neighborhood size ST on filtered original images as the “filtered 9 x 9” ST. The number of control points obtained in each case, the largest error and the distribution of control points over different E ranges is given in Table (7.1). Table (7.1) also gives the number of iterations needed to obtain the results presented in the **D** and **E** diagrams. We have presented the results for that iteration where the highest number of control points was obtained. The algorithm was allowed to run for a fixed number of iterations, and the results

corresponding to that iteration which gave the maximum number of control points are presented. It was typically observed that the number of control points increases at every iteration up to a stage and then either starts falling or going up and down with no particular pattern. Some sample graphs showing the variation of the number of control points with iteration number are provided in the next section.

We could have also selected the results for the iteration before the iteration with the highest control points. The control points at this iteration, while less in number, may be more accurate. The rationale for this is as follows: Let I represent the iteration giving the maximum number of control points, and hence let $I - 1$ represent the previous iteration. Since the results from $I - 1$ were used to generate the intermediate image for I and since I resulted in an increase in control points, this could have happened because the control points at $I - 1$ were accurate, whereas when going from I to $I + 1$ (the next iteration), the number of points falls, and this is also probably because I contains some inaccurate points.

As seen from Table (7.1) as well as the **D** and **E** diagrams, using ST in the three different ways described above does not make a significant difference in this case, since the noise level is low. However, comparing the three figures, it is seen that Figure 7.4 does not give any control points in the lower right region of the image area while the other two methods give a few points in this area. In the same way, while some regions in the image area are better represented in Figure 7.4 (in terms of the number of control points), they are not as well represented in the other two. Figure 7.6, Figure 7.7 and Figure 7.8 show the **D** and **E** diagrams for the 5% noise case using the 9×9 , 15×15 , and filtered 9×9 ST. Figure 7.9, Figure 7.10 and Figure 7.11 show the same for the 10% noise level. For the 5% noise level, the three different methods give comparable results,

For the 10% noise level, the 9×9 neighborhood size ST gives poor results because

of a few large errors. While the control points are better distributed with the filtered 9 x 9 ST, the errors are also larger here as compared to the 15 x 15 ST. It must be noted that the amount of noise added here is excessive (because pixels on an average have a change of about 10% of their value and several pixels have a much larger change) and hence some large errors can be expected to occur. It can be noted, observing the diagrams for all three noise levels, that the number of control points obtained through the method decreases with increasing noise level. However the decrease is much sharper, going from the 5% noise level to the 10% noise level, as compared to going from the 1% noise level to the 5% noise level.

UICP was run on the 10% noise level and the results for these are presented in Figure 7.12, Figure 7.13 and Figure 7.14 as well as summarized in Table (7.1). With UICP a larger number of points have been obtained as compared to AUTOCP. This is because of the stringent rejection criteria with AUTOCP.

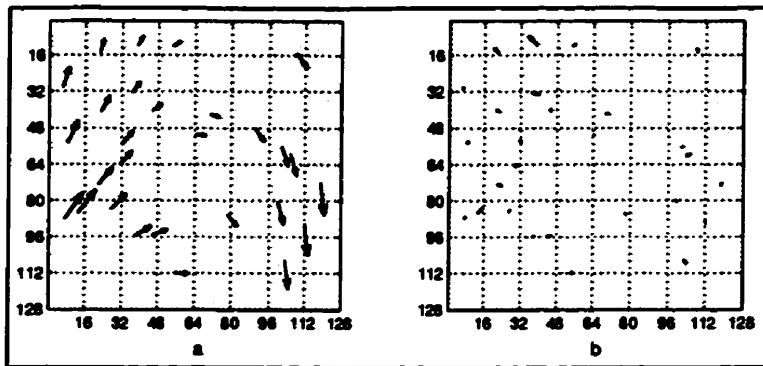


Figure 7.6: *a* and *b*: *D* and *E* for 5 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.

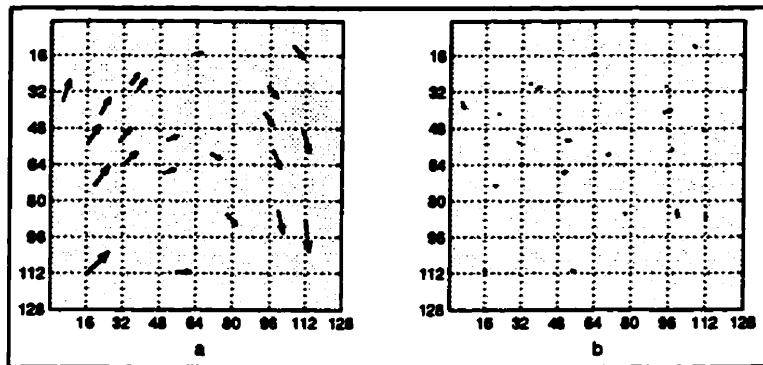


Figure 7.7: *a* and *b*: *D* and *E* for 5 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.

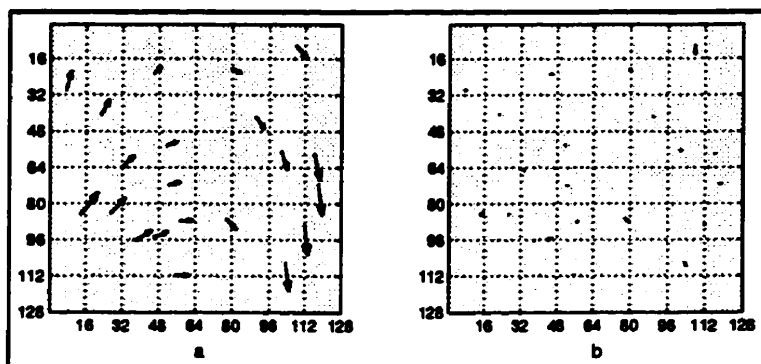


Figure 7.8: *a* and *b*: *D* and *E* for 5 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.

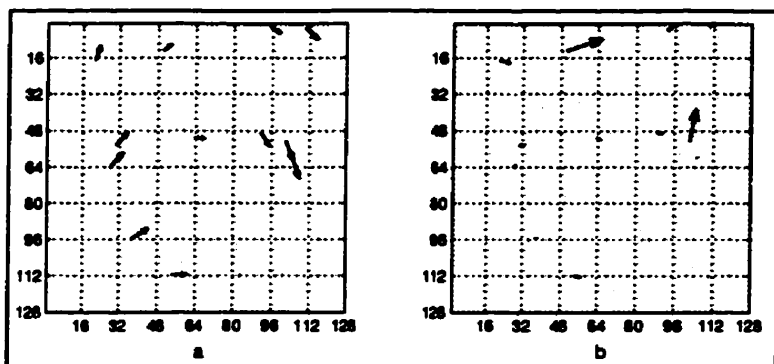


Figure 7.9: *a* and *b*: **D** and **E** for 10 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.

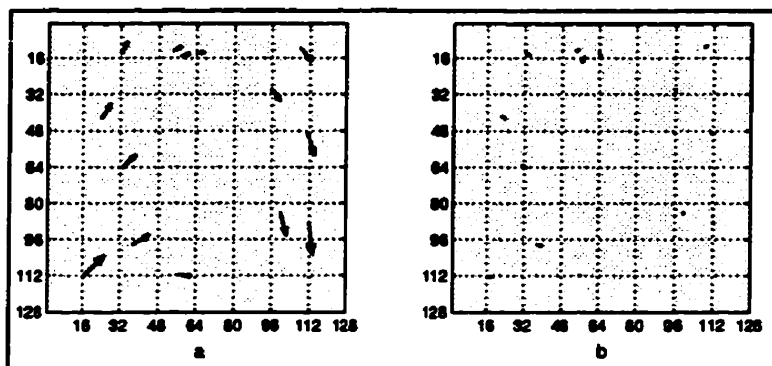


Figure 7.10: *a* and *b*: **D** and **E** for 10 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.

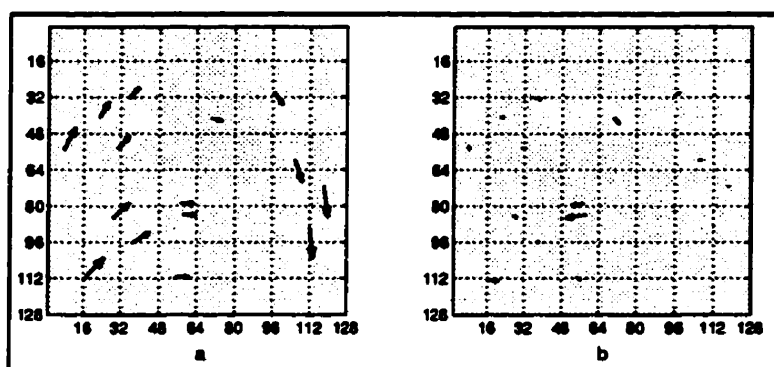


Figure 7.11: *a* and *b*: **D** and **E** for 10 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.

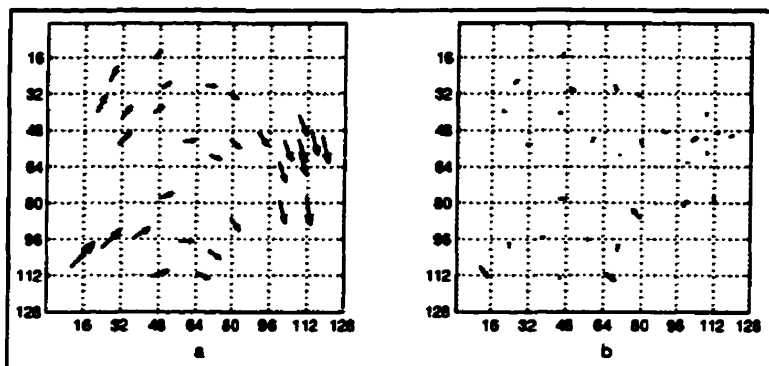


Figure 7.12: *a* and *b*: *D* and *E* obtained using UICP for 10 % noise using a 9 x 9 neighborhood size ST for the bilinear distortion.

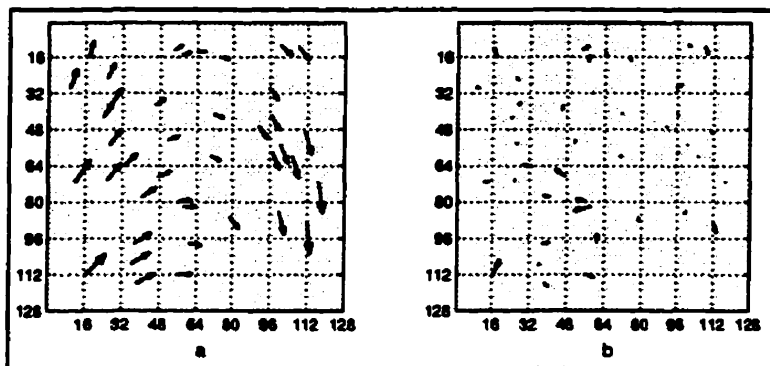


Figure 7.13: *a* and *b*: *D* and *E* obtained using UICP for 10 % noise using a 15 x 15 neighborhood size ST for the bilinear distortion.

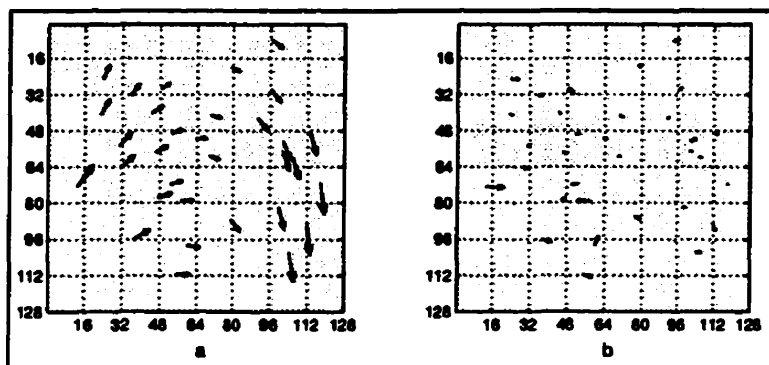


Figure 7.14: *a* and *b*: *D* and *E* obtained using UICP for 10 % noise using a filtered 9 x 9 neighborhood size ST for the bilinear distortion.

7.3 Sinusoidal distortion

Figures 7.15a and b show the original reference and target images. Figures 7.15c and d, e and f, and g and h, show the reference-target pair with 1%, 5% and 10% noise added.

Figures 7.16a and b show the STs of the reference-target pair using a 9 x 9 neighborhood size, while c and d show 15 x 15 STs of the reference and target. e and f show 9 x 9 STs of filtered reference and target images. Figures 7.16g to l, Figures 7.16m to r, and Figures 7.16s to x show the same for the reference-target pair with 1%, 5% and 10% noise added.

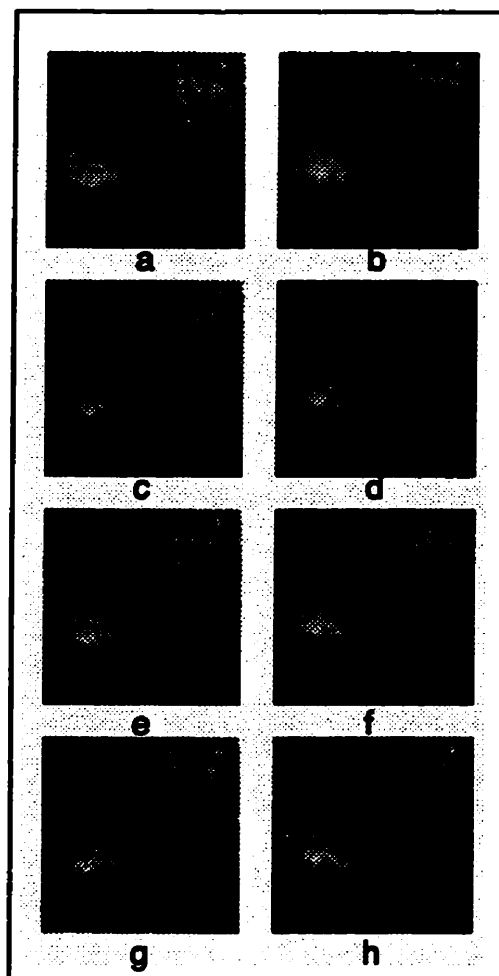


Figure 7.15: *a* and *b*: Reference and target images for the sinusoidal distortion. *c* and *d*: *a* and *b* with 1 % noise. *e* and *f*: *a* and *b* with 5 % noise. *g* and *h*: *a* and *b* with 10% noise.

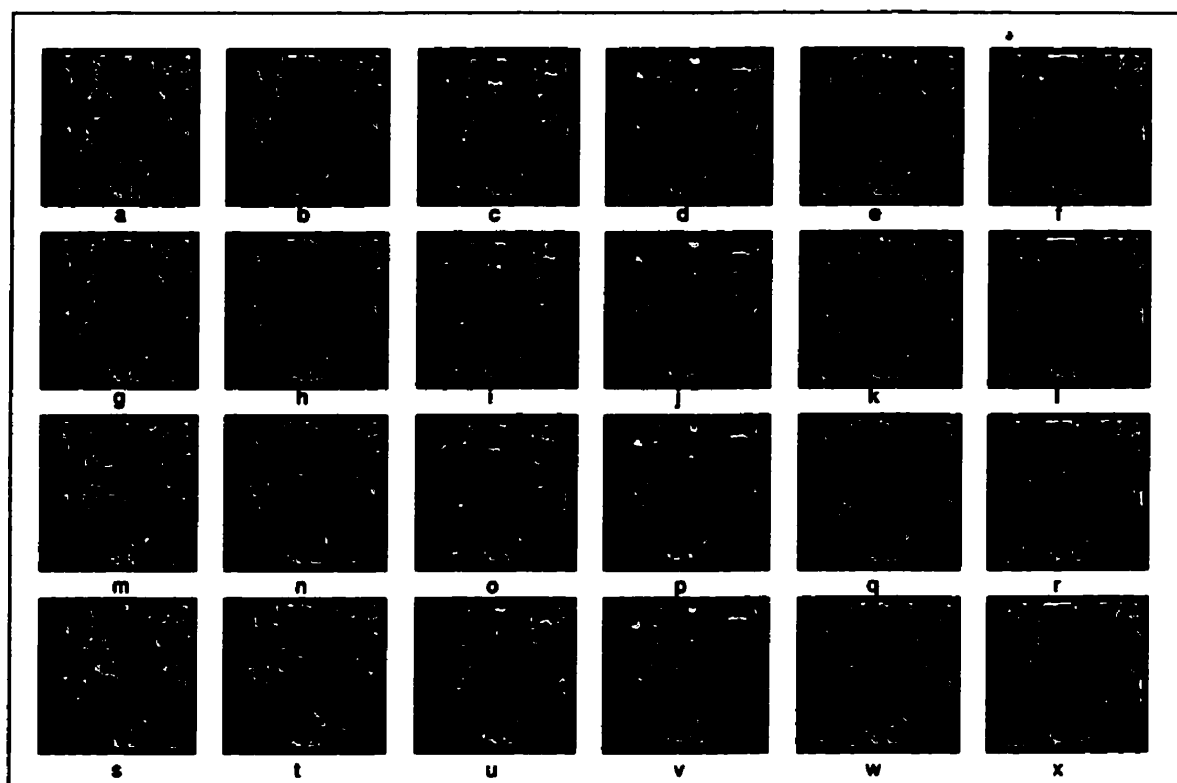


Figure 7.16: *a* to *d*: 9 x 9 and 15 x 15 STs of the reference-target pair for the sinusoidal distortion. *e* and *f*: 9 x 9 STs of filtered reference and target images. *g* to *l*: Corresponding ST images for the reference-target pair with 1 % noise. *m* to *r*: Corresponding ST images for the reference-target pair with 5 % noise. *s* to *x*: Corresponding ST images for the reference-target pair with 10 % noise.

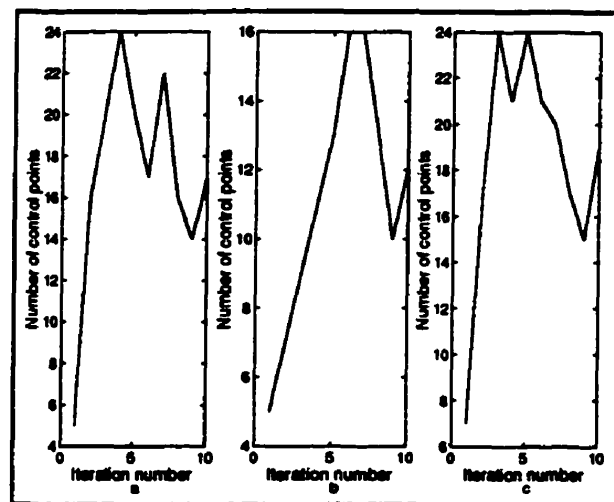


Figure 7.17: *a to c*: Variation of the number of control points with iteration number for the sinusoidal distortion with 5 % noise using the 9 x 9, 15 x 15, and filtered 9 x 9 ST.

Figure 7.18, Figure 7.19 and Figure 7.20 show the D and E diagrams for the 1% noise case using the 9 x 9, 15 x 15, and filtered 9 x 9 ST. In this case the filtered 9 x 9 ST gave the best results. Not only were the number of control points greater and their placement more accurate using the filtered 9 x 9 ST, but the spatial distribution of control points was also better.

Figure 7.21, Figure 7.22 and Figure 7.23 show the D and E diagrams for the 5% noise case using the 9 x 9, 15 x 15, and filtered 9 x 9 ST. In this case too the filtered 9 x 9 ST gave the best results. The graphs in Figure 7.17a to c show the variation of the number of control points with iteration number for the 5 % noise case using the 9 x 9, 15 x 15, and filtered 9 x 9 ST.

Figure 7.24, Figure 7.25 and Figure 7.26 show the **D** and **E** diagrams for the 10% noise case using the 9 x 9, 15 x 15, and filtered 9 x 9 ST. In this case, all three gave comparable results. Figure 7.27, Figure 7.28 and Figure 7.29 show the **D** and **E** diagrams for the 10% noise case using the 9 x 9, 15 x 15, and filtered 9 x 9 ST obtained using UICP. In this case too, the results are comparable, though the 15 x 15 ST has more errors than the other two.

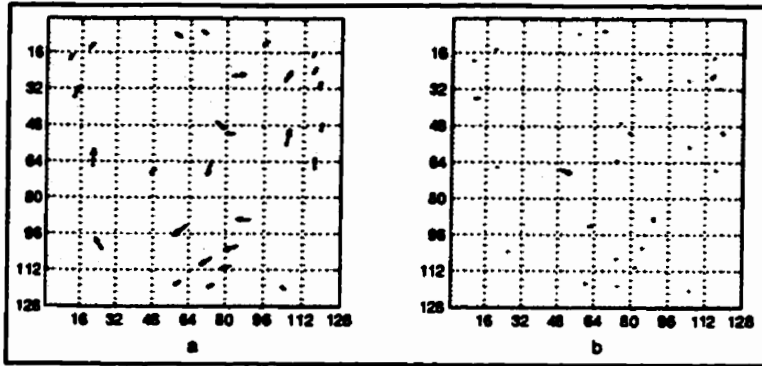


Figure 7.18: *a* and *b*: **D** and **E** for 1 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.

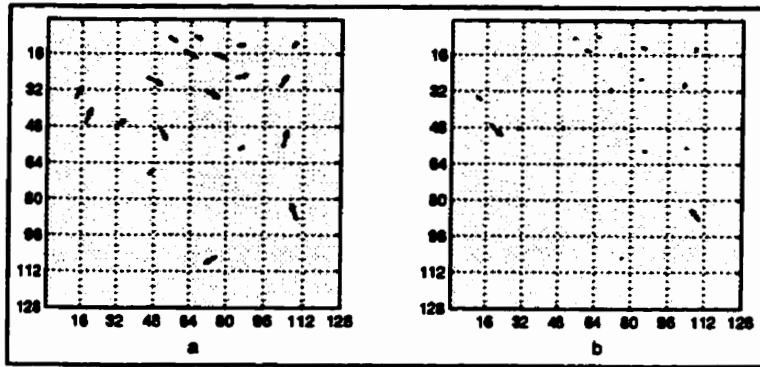


Figure 7.19: *a* and *b*: **D** and **E** for 1 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.

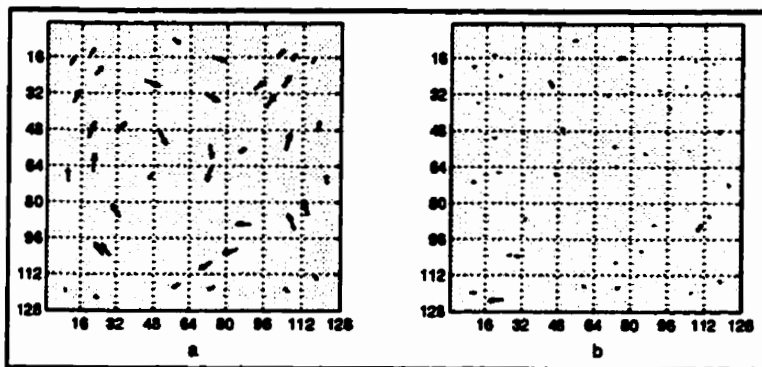


Figure 7.20: *a* and *b*: **D** and **E** for 1 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.

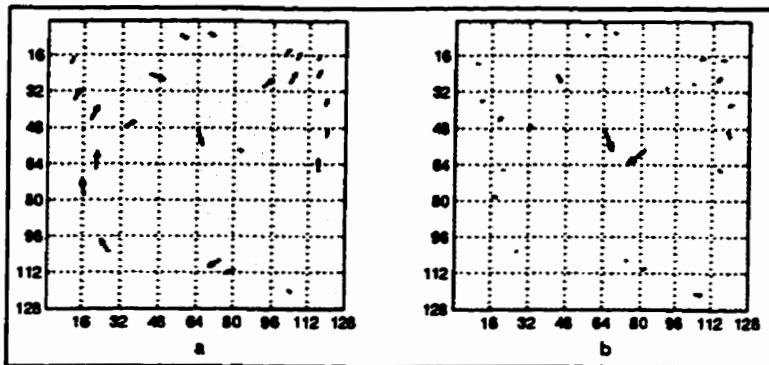


Figure 7.21: *a* and *b*: **D** and **E** for 5 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.

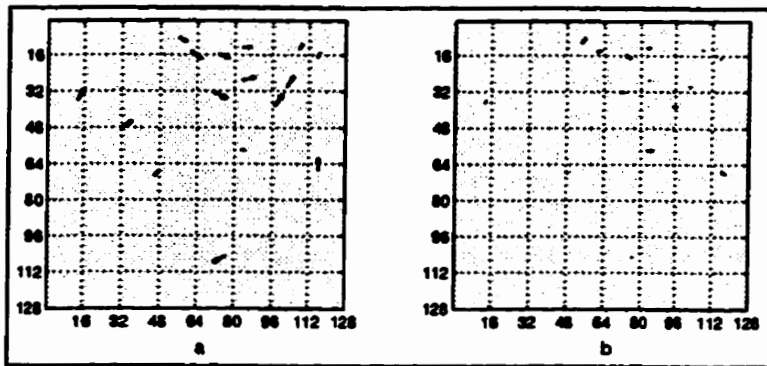


Figure 7.22: *a* and *b*: **D** and **E** for 5 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.

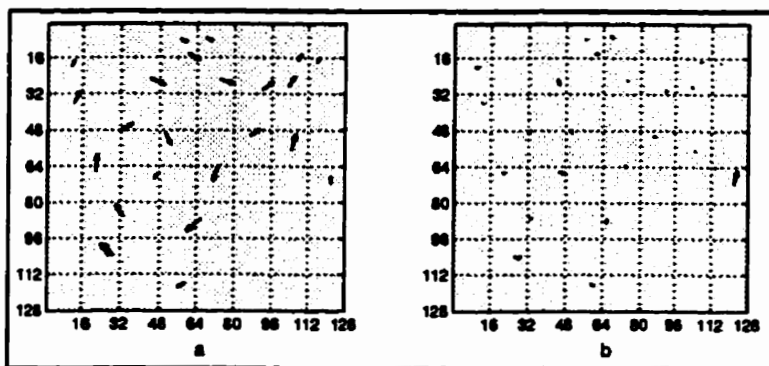


Figure 7.23: *a* and *b*: **D** and **E** for 5 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.

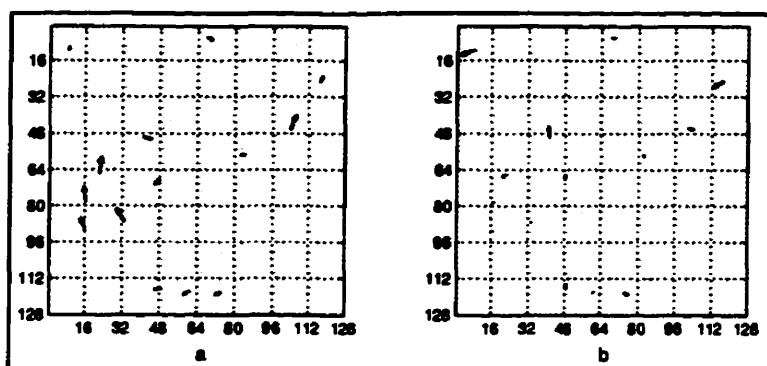


Figure 7.24: *a* and *b*: D and E for 10 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.

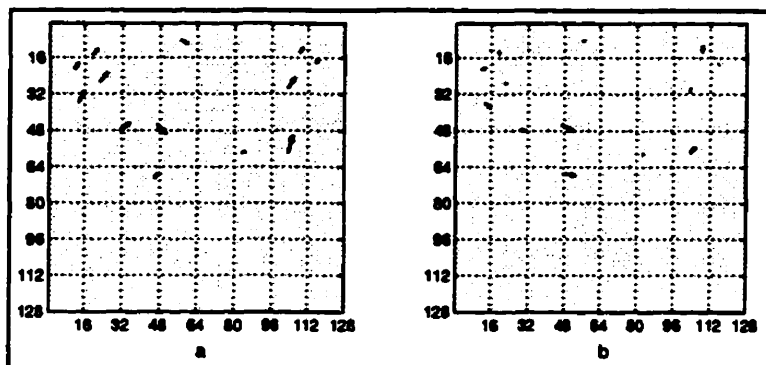


Figure 7.25: *a* and *b*: D and E for 10 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.

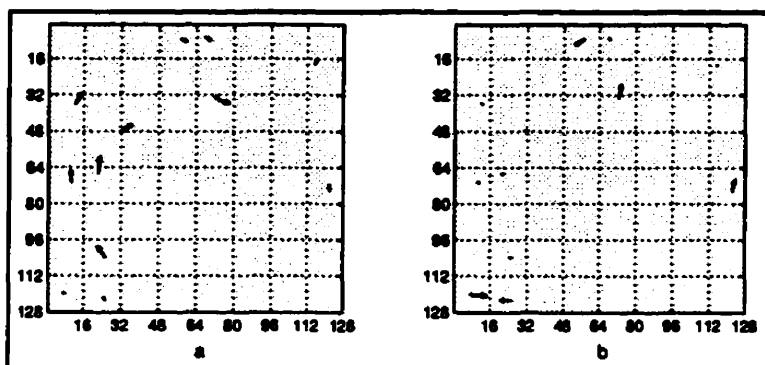


Figure 7.26: *a* and *b*: D and E for 10 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.

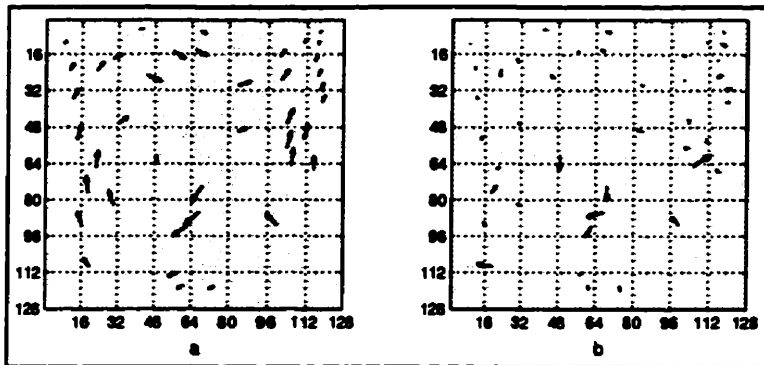


Figure 7.27: *a* and *b*: D and E obtained using UICP for 10 % noise using a 9 x 9 neighborhood size ST for the sinusoidal distortion.

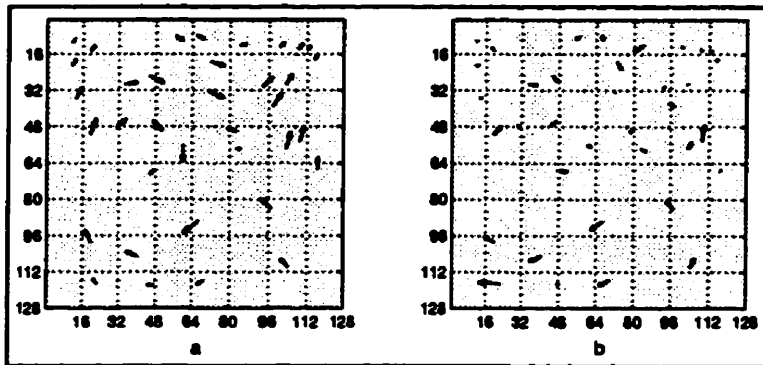


Figure 7.28: *a* and *b*: D and E obtained using UICP for 10 % noise using a 15 x 15 neighborhood size ST for the sinusoidal distortion.

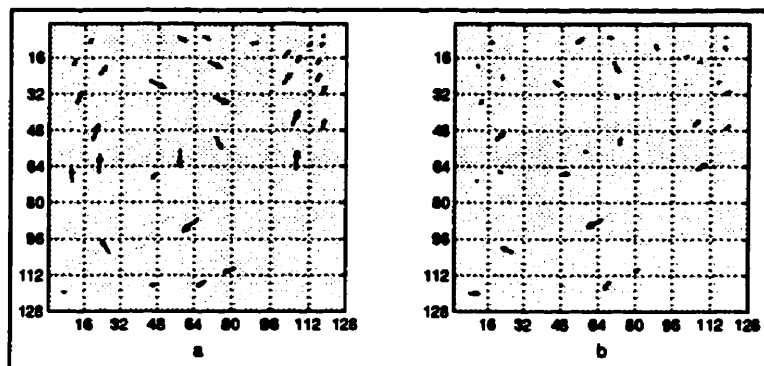


Figure 7.29: *a* and *b*: D and E obtained using UICP for 10 % noise using a filtered 9 x 9 neighborhood size ST for the sinusoidal distortion.

Noise Level	Method	I	Largest Error	N	Percentage of control points with				
					$E < 1$	$1 \leq E < 2$	$2 \leq E < 3$	$3 \leq E < 4$	$E \geq 4$
1 %	9 x 9	5	5.278	28	75	17.860	3.571	0.000	3.571
	15 x 15	4	6.679	19	47.37	42.110	0.000	0.000	10.530
	F 9 x 9	5	5.662	40	60.00	27.500	7.500	2.500	2.500
5 %	9 x 9	4	9.853	24	37.50	33.330	12.500	8.333	8.333
	15 x 15	6	2.736	16	37.50	43.750	18.750	0.000	0.000
	F 9 x 9	5	6.203	24	50.00	37.500	8.333	0.000	4.167
10 % (A)	9 x 9	13	6.530	14	28.57	28.570	21.430	0.000	21.430
	15 x 15	13	5.804	13	30.77	23.080	15.380	15.38	15.380
	F 9 x 9	4	6.936	12	41.67	8.333	8.333	0.000	41.670
10 % (M)	9 x 9	-	8.524	41	26.83	41.460	12.200	2.439	17.070
	15 x 15	-	9.676	36	19.44	25.000	16.670	8.333	30.560
	F 9 x 9	-	7.208	34	23.53	32.350	14.710	11.760	17.650

Table 7.2: **Sinusoidal Distortion:** *Column 1:* Noise Level. A and M stand for "Automatic" and "Manual". *Column 2:* ST method used. F stands for "Filtered". *Column 3:* I , number of iterations. *Column 4:* Largest error in pixels. *Column 5:* N , the number of control points obtained using the user-interactive program. *Column 6:* Break up of control points over different error (E) ranges measured in pixels.

The number of iterations, the number of control points obtained in each case, the largest error and the distribution of control points over different E ranges is given in Table (7.2).

It can be seen from the tables that as the noise level increases, the number of control points obtained using AUTOCF falls. Also, the number of iterations needed at each stage is seen to increase with noise level.

7.4 Conclusions

The performance of the starbyte transformation was studied when noise was added to the original images. The additive noise was assumed to have a Gaussian distribution. The performance of the starbyte algorithm at different noise levels was studied.

The effect of change in the neighborhood size as well as pre-filtering of the original images, on the accuracy of control point positions was also studied. While the performance of the algorithm definitely degraded with increase in the noise level, even at high noise levels, while the number of control points with large error increased, accurate control points were also extractable with the algorithm. The total number of points extractable with AUTOCIP was found to decrease with increase in noise level. With UICP, at the same noise level, a much larger number of points could be extracted. However, the percentage of control points over different E ranges was approximately the same for both AUTOCIP and UICP. In the next chapter, we shall study the effect of adding simulated tumors to the performance of the starbyte algorithm.

Chapter 8

STARBYTE PERFORMANCE WITH SIMULATED TUMORS

8.1 Introduction

In the previous chapter, we studied the effect of additive noise on the performance of the starbyte algorithm. It was found that while an increase in added noise resulted in an increase in the number of inaccurate points, it was also seen that accurate points were still extractable with the algorithm even at high noise levels. It was also noted that for large noise levels, while AUTOCP extracted fewer points, it was still possible to extract a large number of points using UICP. However, the percentage of control points across different E ranges was approximately the same for both AUTOCP and UICP.

In this chapter, the performance of the starbyte algorithm is evaluated when simulated tumors are added to the target image. In order to do a reasonable test of the algorithm's performance with added tumors, the tumors were added in such a way that they are not clearly discernible in the target image. Also, as will be seen, a plain subtraction between the target and reference does not clearly reveal the tumors. Hence, this is a true test of the algorithm's capacity to register the target and reference, so that the tumors become visible when the target and reference are matched up.

The experiments were again performed for the same two distortions as the earlier

chapter, namely the bilinear and sinusoidal distortion. As before, these two distortions were selected because they represent distortions of large magnitude and are also reasonably complicated.

A tumor T , modeled as a circular disk with a specific density level and radius, can be represented as a function $T(d, r)$. A "tumor" image was first created which contained four tumors (two added to the image and two subtracted from the tumor image): $T(d_1, r_1)$, $T(d_1, r_2)$, $T(d_2, r_1)$, $T(d_2, r_2)$. These were placed at different locations in the tumor image, which was zero everywhere else. The tumor image was then filtered with a 5 x 5 moving average filter, so that the tumors were blurred. This new blurred tumor image was added to the target image. The numerical values chosen for d_1 and d_2 were approximately 10 % and 20 % of the maximum density value of the target images, while the values for r_1 and r_2 were chosen to be approximately 1.5% and 3% of the total size of the images (or 2 and 4 pixels, respectively). The locations of the tumors for the two distortions were different and hence the tumor image used for the two cases was different. Again the results for each distortion are presented separately. Protocol 3 was used in both cases.

8.2 Bilinear distortion

Figures 8.1a and b show the reference and target images for the bilinear distortion. Figure 8.1c shows the target image with the tumors added. Figure 8.2a and b show the STs of the reference and target images using a 9 x 9 neighborhood size. When comparing the STs, it can be seen that corresponding regions in the two images can be identified, in spite of the presence of the four additional tumors in the target image.

Figure 8.3a and b, and Figure 8.4a and b, show the D and E diagrams for the manual and automatic case. For the automatic case, three iterations were required

to get this result.

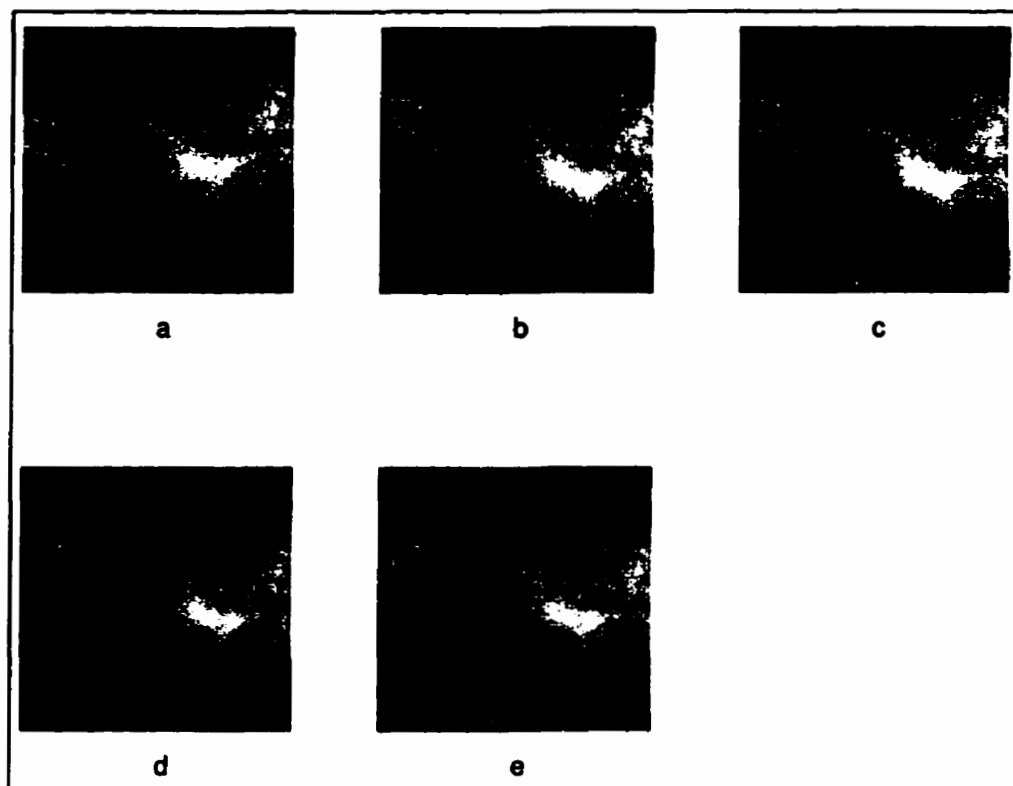


Figure 8.1: *a* and *b*: Reference and target images for the bilinear distortion. *c*: *b* with tumor image added. *d* and *e*: Unwarped images obtained using UICP and AUTOCIP.

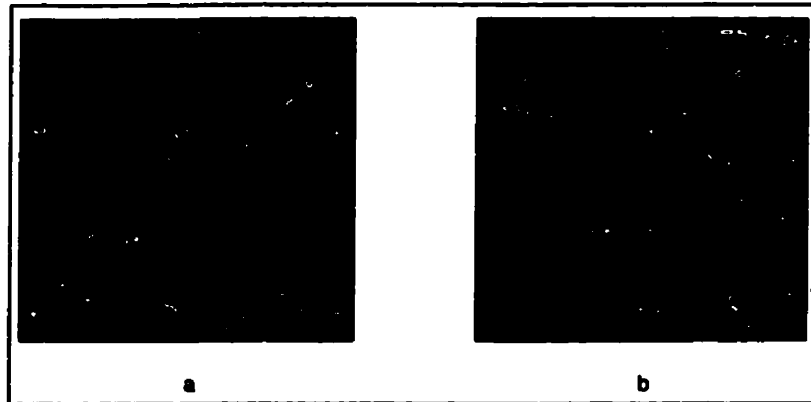


Figure 8.2: *a* and *b*: STs of the reference and target images for the bilinear distortion using a 9×9 neighborhood size.

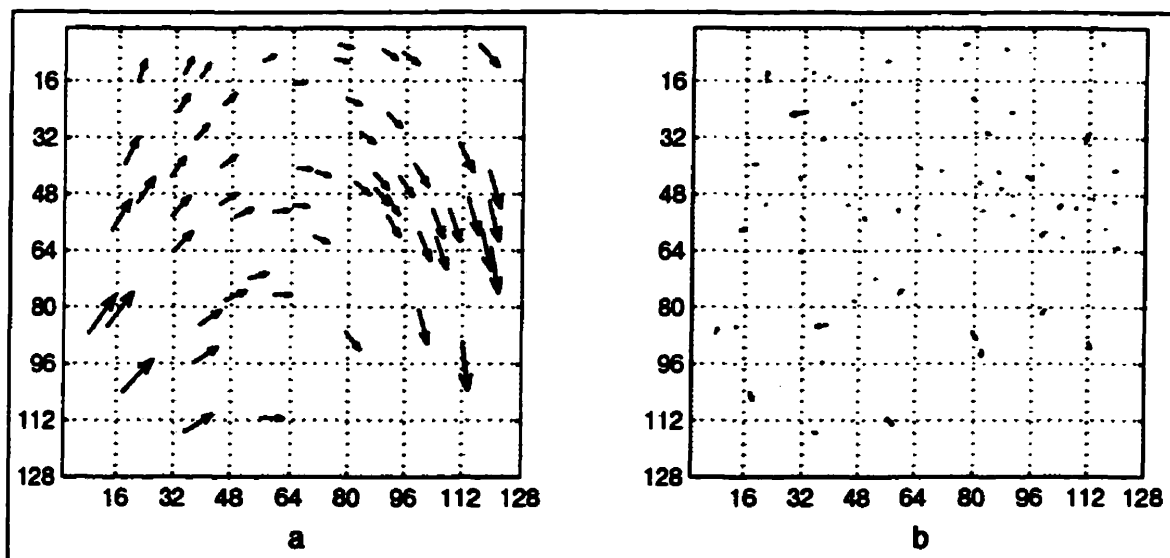


Figure 8.3: *a* and *b*: D and E vector diagrams for the bilinear distortion obtained using UICP, with 4 simulated tumors present in the target image.

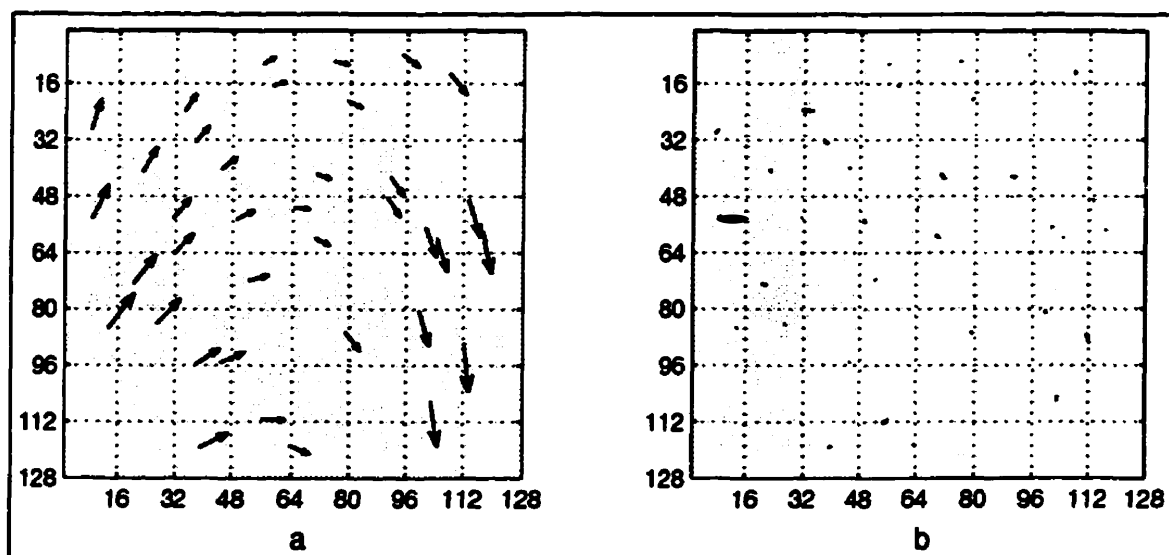


Figure 8.4: *a* and *b*: D and E vector diagrams for the bilinear distortion obtained using AUTOCP, with 4 simulated tumors present in the target image.

Figure 8.1d and e show the unwarped images for the manual and automatic case. Figure 8.5a shows the absolute difference image between the reference and target. As can be seen, the tumors cannot be differentiated from the background registration noise. Figure 8.5b and c show the absolute difference images between the target and unwarped images for the manual and automatic case respectively, while Figure 8.5d shows the actual tumor image. As can be seen, the four tumors are very clearly visible for the manual cases. The two large tumors and one small tumor are clearly discernible for the automatic case. The other tumor is partially discernible. This is because there is still some misregistration error remaining after the unwarping process. However, a large amount of registration has taken place.

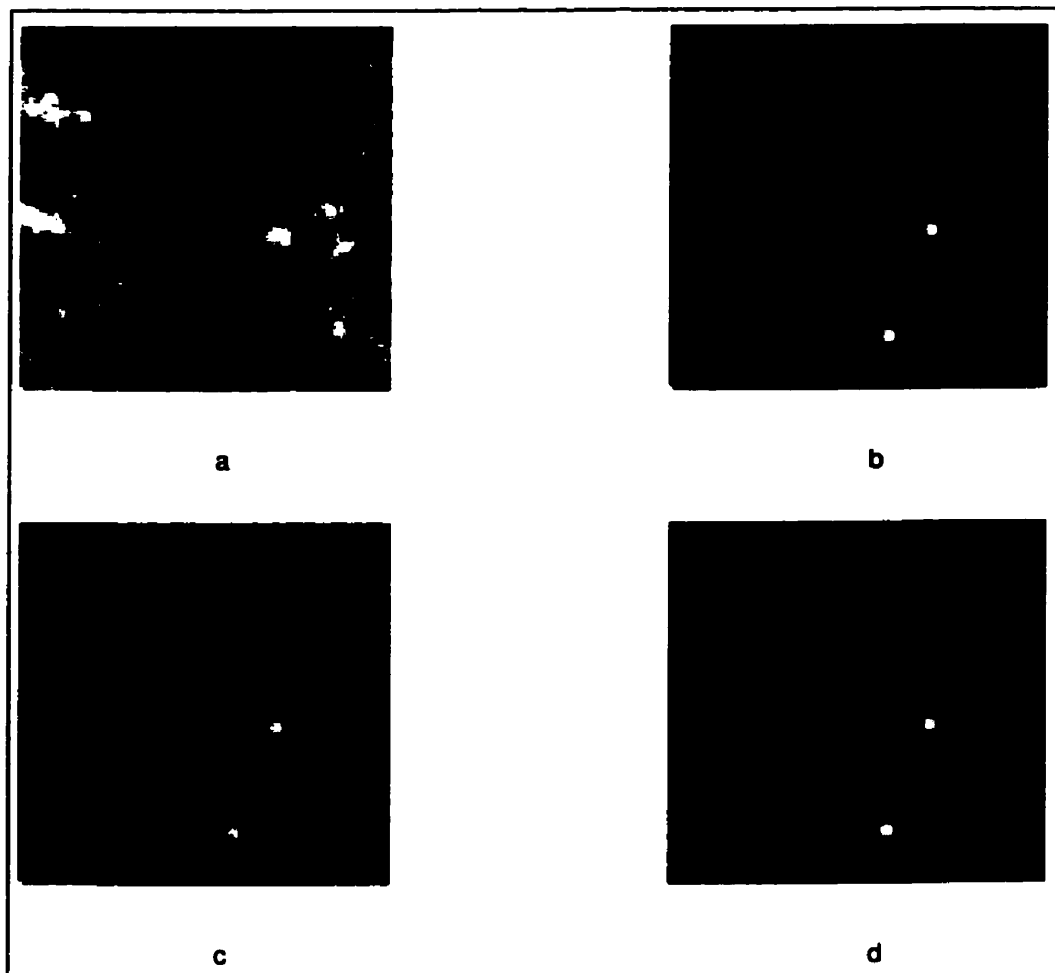


Figure 8.5: *a*: Absolute difference image between the reference and target for the bilinear distortion. *b* and *c*: Absolute difference images between the target and unwarped images for the manual and automatic cases. *d*: Actual tumor image for the bilinear distortion.

8.3 Sinusoidal distortion

Figures 8.6a and b show the reference and target images for the sinusoidal distortion. Figure 8.6c shows the target image with the tumors added. Figure 8.7a and b show the STs of the reference and target images using a 9 x 9 neighborhood size.

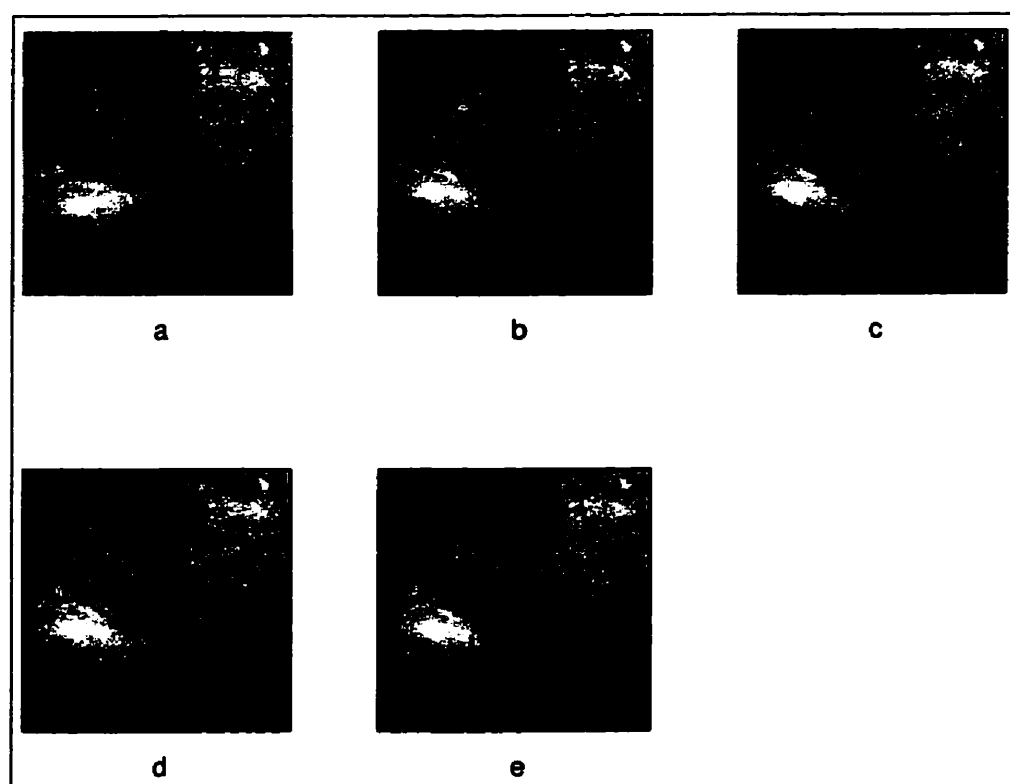


Figure 8.6: *a* and *b*: Reference and target images for the sinusoidal distortion. *c*: *b* with tumor image added. *d* and *e*: Unwarped images obtained using UICP and AUTOCP.

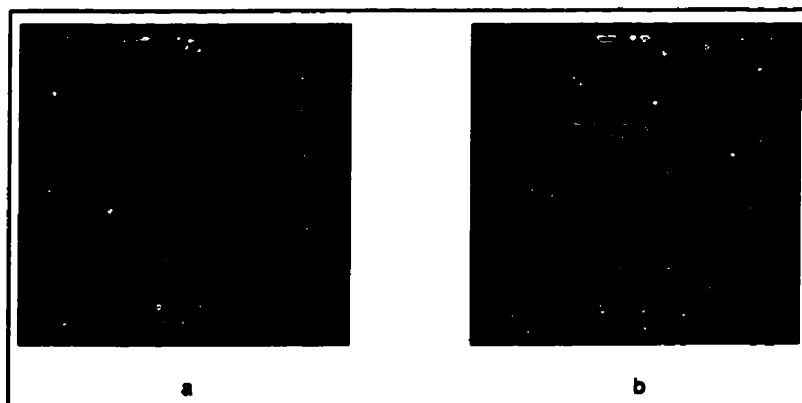


Figure 8.7: *a* and *b*: STs of the reference and target images for the sinusoidal distortion using a 9 x 9 neighborhood size.

Figure 8.8a and b, and Figure 8.9a and b, show the **D** and **E** diagrams for the manual and automatic case. For the automatic case, 11 iterations were required to get this result.

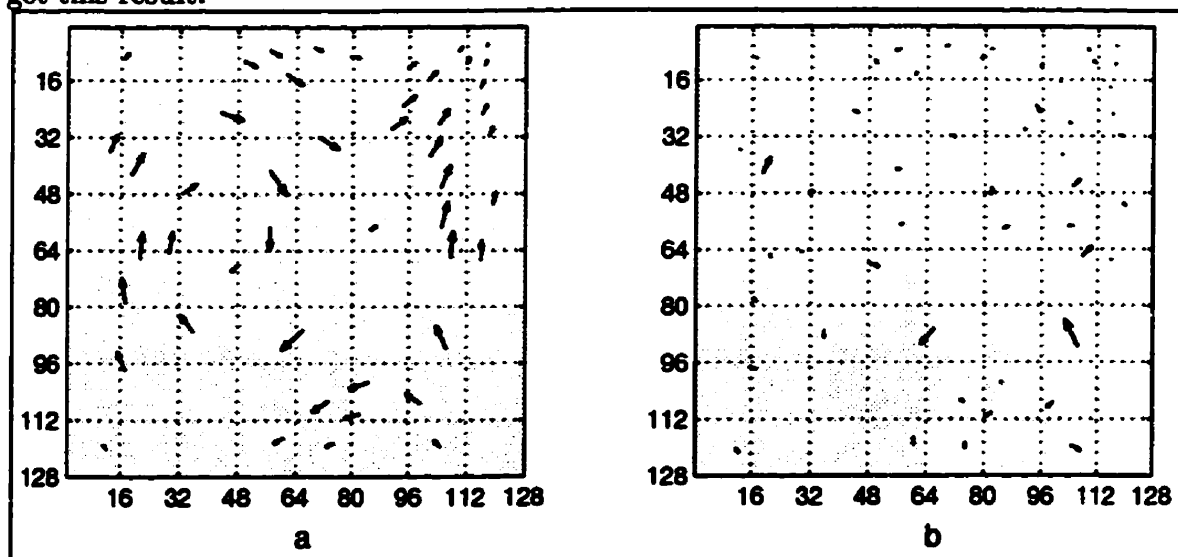


Figure 8.8: *a* and *b*: **D** and **E** vector diagrams for the sinusoidal distortion obtained using UICP, with 4 simulated tumors present in the target image.

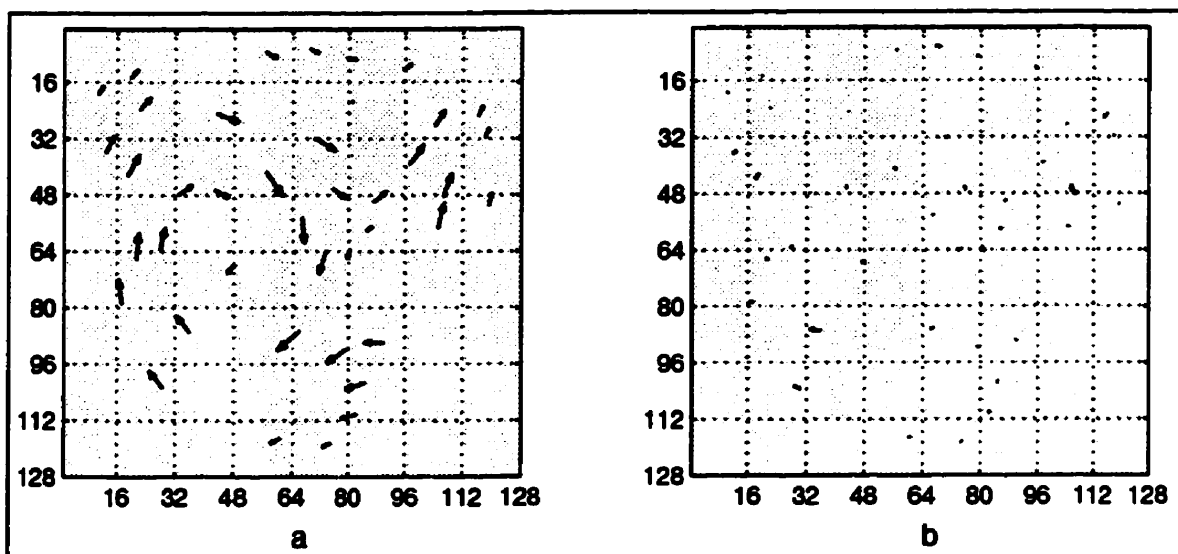


Figure 8.9: *a* and *b*: **D** and **E** vector diagrams for the sinusoidal distortion obtained using AUTOCP, with 4 simulated tumors present in the target image.

Figure 8.6e and f show the unwarped images for the manual and automatic case. Figure 8.10a shows the absolute difference image between the reference and target. As can be seen, the tumors cannot be differentiated from the background registration noise. Figure 8.10b and c show the absolute difference images between the target and unwarped images for the manual and automatic cases respectively, while Figure 8.10d shows the actual tumor image. As before, the four tumors are clearly visible for the manual case. Again, as before, the two large tumors and one small tumor are clearly discernible for the automatic case, with the other being partially discernible, because of a small amount of remnant misregistration error after the unwarping process. As in the case for the bilinear distortion, a significant amount of registration has taken place.

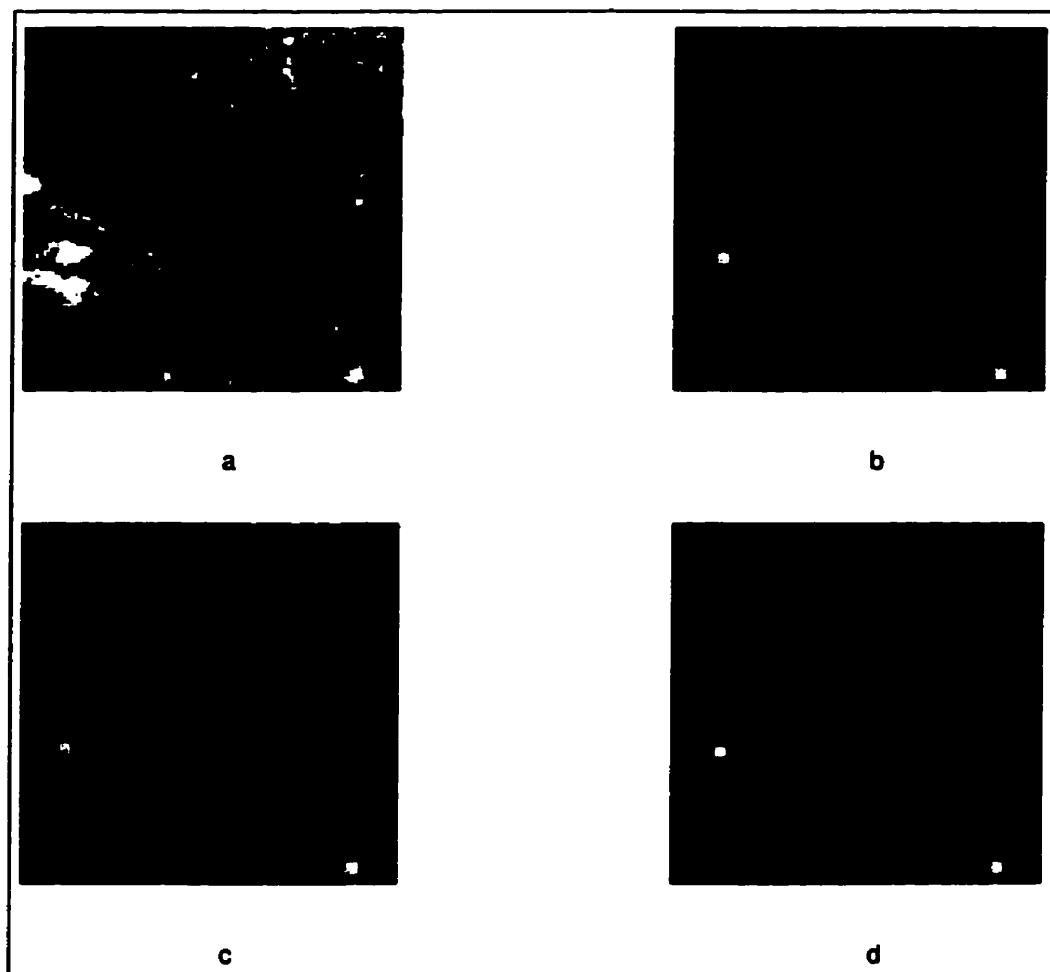


Figure 8.10: *a*: Absolute difference image between the reference and target for the sinusoidal distortion. *b* and *c*: Absolute difference images between the target and unwarped images for the manual and automatic cases. *d*: Actual tumor image for the sinusoidal distortion.

8.4 Conclusions

The performance of the starbyte transformation was studied when simulated tumors were added to the target image. The study was performed for two distortions, namely the bilinear and sinusoidal distortion. A tumor image created with four tumors with different densities and radii at different locations, was added to the target images in each case. It was found that both AUTOCP and UICP were successful in registering the given pairs of images, in spite of the presence of the tumors. While some misregistration error remained after the unwarping in the automatic case, a significant amount of registration was performed and the tumors were discernible in the difference images.

Up to now, we have studied the image registration problem using the starbyte transformation for simulated distortions. Successful image registration depends upon successful extraction of accurate control points. In all cases, extraction of accurate control points was entirely possible only because of the use of the starbyte transformation, which has thus greatly aided the matching process. Matching of the original images directly in each case would have been clearly difficult, if not impossible, because of their low contrast and lack of sufficient landmarks. In the next chapter, we present preliminary results for matching two pairs of approximately corresponding MRI slices. These were taken from 3D data sets of two volunteers who were imaged twice. The next chapter also addresses the issue of 3D breast imaging.

Chapter 9

REGISTERING REAL IMAGES: THE NEED FOR 3D IMAGING

9.1 Introduction

Up to now, the image registration problem using the starbyte transformation was studied using simulated distortions. It was found that accurate control points could be extracted for complicated distortions relating the reference and target images. Control points could also be extracted when noise was present in the original images. The algorithm was also shown to work when artificial tumors were added to the target image. However, the entire work presented was for 2D images. While the results are important and show that the starbyte registration algorithm has potential, more work remains to be done before the method can be used in actual clinical breast imaging.

Traditional mammography has serious limitations because of its 2D nature, so that registering two temporal mammograms can only have limited success, because of the large amounts of superimposed tissue which has all been projected together on a 2D surface. Hence future research needs to be directed towards 3D breast imaging. We discuss the need for 3D breast imaging in this chapter. We also present a “negative” result of registering two approximately corresponding portions of temporal mammograms. Due to the fact that the algorithm is still 2D in nature, and in order to demonstrate its working on pairs of real images, we present the results of registering two slices taken from MRI (magnetic resonance imaging) 3D data sets of

the same volunteer. We also demonstrate that results would be much more inferior had projections of these 3D MRI images been registered instead of slices.

9.2 3D breast imaging

Currently, mammography is the first-line imaging technique for the detection and diagnosis of breast lesions. However it still misses about 5 to 15 % of all tumors [4], [83], [103]. This is because of its two-dimensional (2D) nature. A tumor is primarily missed because tissue present above and below it is superimposed together on a 2D projection in mammography. The problem is especially more serious when imaging women with dense breasts . In particular, this excludes most women under age 40 whose breasts are mammographically dense [59]. Due to all these problems, it is necessary to image the breast in 3D, so that no tissue overlap can occlude the visibility of an underlying tumor. Three important modalities have to be investigated in connection with 3D breast imaging namely computed tomography (CT), MRI, breast sonography and electrical impedance tomography (EIT). EIT for breast is still in its infancy and is only now being developed [85].

Earlier attempts at CT mammography in the late 1970s for breast did not succeed because of various reasons [124]. First of all, it was noticed that there was considerable overlap in the CT numbers of many breast cancers, as well as several types of benign breast lesions [17]. Only by imaging the breast twice, before and after intravenous iodide administration, could CT scanning provide any useful diagnostic tool [111], because of a radiological contrast enhancement by contrast agents which created a 5% increase in CT numbers for breast cancers compared to benign lesions [18], [39]. While clinical investigation of CT mammography was initially enthusiastic, it was soon found that there was only a marginal improvement in detection accuracy as

compared to standard mammography [18]. A dedicated CT scanner unit developed by General Electric showed no improvement over standard mammography and in fact, showed a higher false positive rate than standard mammography [39]. This was probably because of the low spatial resolution used ($1 \text{ voxel} = 1 \text{ mm} \times 1 \text{ mm} \times 1 \text{ cm}$). Also the cost of the examination, the need for intravenous iodide administration, and the high radiation dose needed, soon made CT mammography impractical for routine screening or diagnosis.

With the introduction of spiral or helical scanning which allows the imaging of the whole breast in less than a minute, recent studies have begun focusing once again on CT breast imaging [116]. Raptopoulos *et al* [95] assess the performance of high resolution CT on breast biopsy specimens following conventional X-ray specimen mammography. They conclude that with fatty tissue, CT and mammography performed equally well. However, for masses in dense tissue where mammography predominantly is ineffective, CT performed much better. This is because the masses are not occluded by the presence of overlapping tissue. However, because of the averaging effect of the large voxels used in CT, microcalcifications are very poorly detected. It may be possible to combine dual energy electronic imaging with CT to improve microcalcification detection [124], [62]. Raptopoulos *et al* conclude that CT may be selectively used for those patients with dense, difficult-to-evaluate breasts where mammography has proved ineffective.

In order for CT to be used on a regular basis for screening of asymptomatic women, the dose problem will have to be solved [41]. Muller *et al* [84] calculated the dose of CT mammography to be three to six times that of diagnostic X-ray mammography. The dose problem can be tackled in several ways [124]: In commercial X-ray CT machines, 180 or more views are taken. Hence we have the choice of either using very few photons per view or very few views. The former approach needs new extensions to

present CT algorithms so that the image can be reconstructed from noisy data. The latter approach has been used in nevoscopy [24], where images were reconstructed from only three views. Reconstruction of images from limited data has been studied extensively in the literature [42]. Another way of reducing the dose may be to image only a portion of the breast, and use CT algorithms that can reconstruct just local regions [113]. Dose reduction may also be possible by performing CT mammography with monochromatic X-rays [27], [28].

MRI is another important imaging modality which may prove very useful for 3D breast imaging. Earlier investigations with whole body coils [110], [100] had limited success. The development of dedicated surface coils resulted in the ability to image smaller lesions than is traditionally possible with standard mammography [21]. Since then many breast coils have been developed, partly as transmitter and receiver coils. Single breast coils as well as surface coils which imaged both breasts simultaneously were also developed [103], [80]. Recently a new high-resolution breast coil has been developed [19]. While MRI is relatively new and hence any clinical experience with breast MRI is still limited, MRI has already proved to be superior to mammography in differentiating solid from cystic lesions, and shows an advantage over mammography in clearly differentiating between dense and fatty tissue. However, MRI too has its shortcomings. Microcalcifications are not well imaged with MRI [21]. Also, while masses embedded in fatty tissue are easily imaged, masses adjacent to fibroglandular tissue are not well imaged. Also the high cost of each examination, as well as the time needed for signal collection have prevented MRI thus far from being a useful tool in routine breast cancer screening, although it has been used as an adjunct to mammography for diagnosis. The nearly standard use of gadolinium contrast agents which are mildly neurotoxic [102], makes MRI of breast an invasive procedure. However, due to its merits, the usefulness of MRI for 3D breast imaging

should continue to be investigated.

Breast sonography has been used for a long time as a helpful adjunct to mammography in the evaluation of abnormalities of the breast [11], [31]. While microcalcifications are not well imaged with sonography and the resolution of the images is poor compared to mammography, it is presently the most reliable technique to differentiate cysts from solid masses with accuracy rates of 96 to 100 %. Other areas where breast sonography can be used include ([124], [103]) imaging dense breasts, the application of Doppler techniques to make benign-malignant differentiations of sonographically detectable masses (by imaging the vasculature), ultrasound-guided aspiration biopsy of nonpalpable lesions etc. Recent developments in breast sonography include the computer-aided analysis of B-mode images as well as a new technique for breast lesion detection by measuring changes in velocity and attenuation from a projection image taken similar to an X-ray projection image [97]. Sabel and Aichinger in [103] say that "ultrasonic methods of reconstructive tomography, pattern recognition and 3D imaging will remain topics of scientific research during the coming years."

For a complete and comprehensive review on recent developments in breast imaging, see [103].

Even if 3D breast imaging became a reality, image registration would still remain a major problem. This is because, in order to "solve" the breast cancer problem, we need to detect extremely small tumors. Hence, even if there is no overlap of tissue, many of the structures found in the breast can be of the size of the tumors we are actually searching for. Since an important property of a tumor is its ability to grow (and this is especially true for "fast" tumors which have a higher metastatic potential) [56], we would still need to take two 3D images of a patient, so that any change can be detected by subtracting the two 3D images. However, because of the loose and non-rigid nature of breast tissue, these two 3D images would be misaligned

and hence would have to be first aligned before they can be subtracted. Hence image registration will continue to be a major problem even if 3D breast imaging becomes routinely available.

Up to now, image registration research in connection with breast cancer has been 2D in nature [125], [78], [104]. However registering longitudinal mammograms of the same patient will not give us any meaningful results. This can be simply expressed in the following way:

Let $F(x, y, z)$ and $G(x, y, z)$ represent two volume breast images of the same patient taken at different times. Projections $f(x, y)$ and $g(x, y)$ taken along the z axis are given by:

$$\begin{aligned} f(x, y) &= \int_z F(x, y, z) dz \\ g(x, y) &= \int_z G(x, y, z) dz \end{aligned} \tag{9.1}$$

Registering $f(x, y)$ and $g(x, y)$ will make sense only if the initial distortion T relating $F(x, y, z)$ and $G(x, y, z)$ is independent of z . However, in reality, when a woman's breast is placed at different times in the machine, this condition is not met, partly because of the non-rigid nature of tissue and also because of other changes in the breast. Hence the distortion T is a function of x , y , and z and image registration has to be done in 3D if any useful results are to be obtained. We now present below a "negative" result obtained by trying to register two approximately corresponding portions of two mammograms of the same patient taken about 2 years apart. This data was from [86].

9.3 Results with approximately corresponding portions of a pair of temporal mammograms

Figure 9.1a and b show approximately corresponding portions picked out by eye from temporal mammograms of Patient # 18 in [86]. These portions also contain microcalcifications. Portions from the original mammograms were downsampled by a linear factor of 4 from 600 x 600 pixels to 150 x 150 pixels to obtain the images shown in Figure 9.1a and b. The regions containing microcalcifications in these portions have been identified by drawing a white circle around them in the reference and target images. As can be seen, the region containing microcalcifications is larger in the target. Hence, ideally, if these two portions could be registered accurately and subtracted, the extra microcalcifications which have arisen in the time interval between the two images, should be visible in the difference image. Figure 9.2a and b show the STs of the original images using a neighborhood size of 15 x 15, with protocol 1.

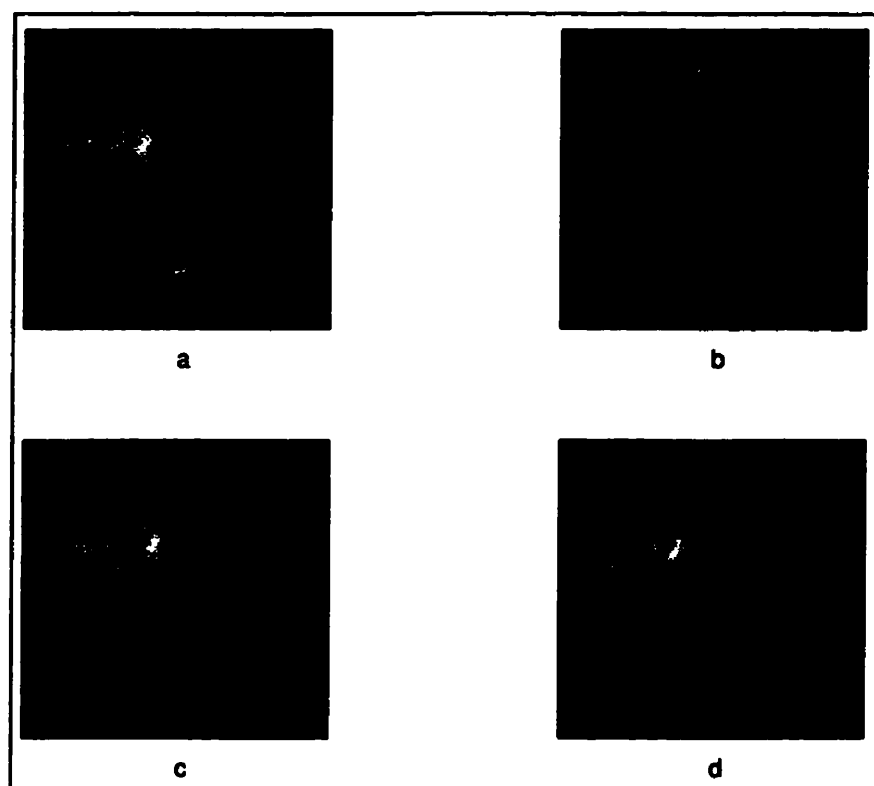


Figure 9.1: *a* and *b*: Portions of actual temporal mammograms of the same patient. *c* and *d*: Unwarped images obtained with and without the starbyte transformation.

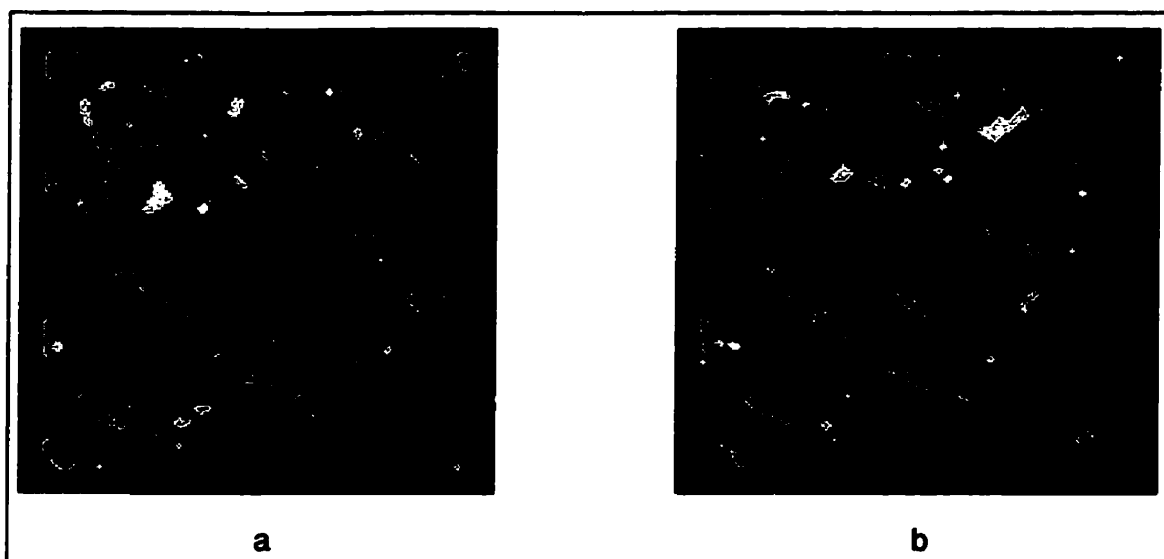


Figure 9.2: *a* and *b*:STs of the original images.

UICP alone was run on these images. A total of 40 control points was returned by UICP out of which 25 good control points were selected for this experiment. This was done so that we could truly evaluate if, using good control points, any useful results could be obtained by registering the given portions of the temporal mammograms.

Figure 9.3a and b show these 25 control points superimposed on the reference images. It should be noted that these control points were those returned by UICP. In order to also evaluate direct selection of control points, as against using the starbyte transformation, control points were also picked by eye directly from the original images. A total of 14 control points was thus obtained. In performing the experiment with the original images directly, an effort was made to select as many points as possible. However, because of the large differences between the reference and target, only 14 points could be obtained. Figure 9.4a and b show these 14 points superimposed on the original images.

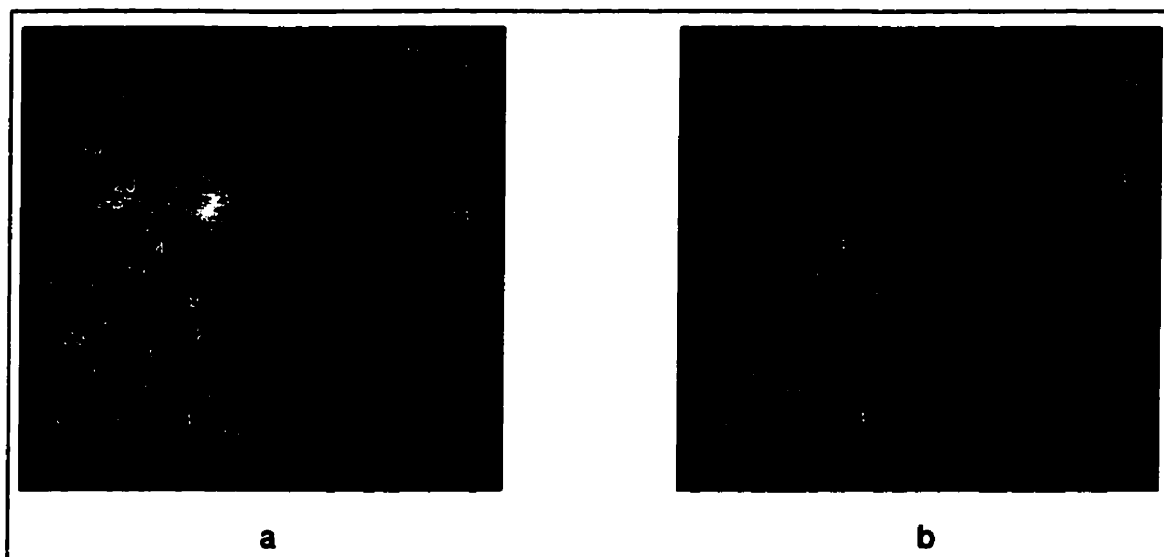


Figure 9.3: *a* and *b*: Reference and target images with 25 control points obtained using UICP superimposed.

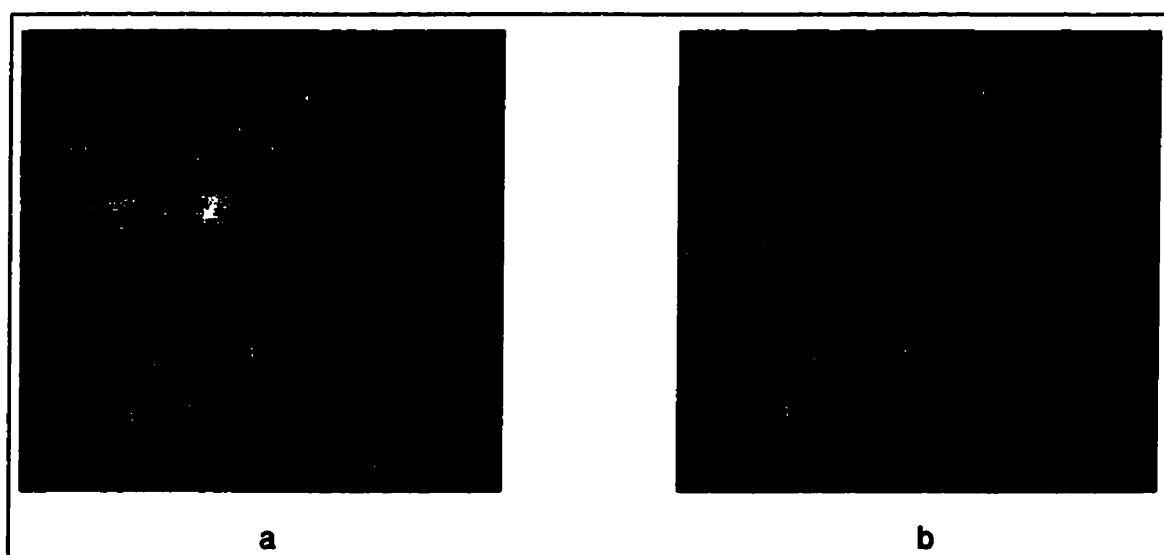


Figure 9.4: *a* and *b*: Reference and target images with 14 control points obtained directly from the original images, superimposed.

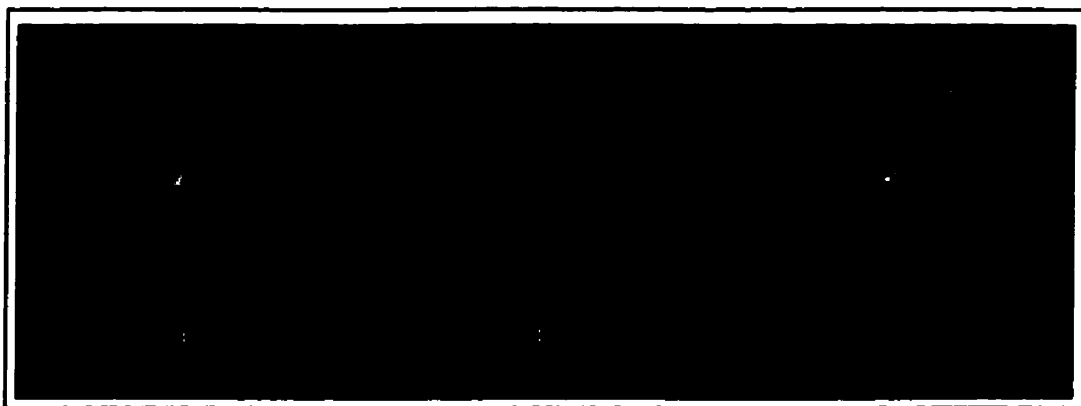


Figure 9.5: *a*: Absolute difference image between the reference and target . *b* and *c*: Absolute difference images between the target and unwarped images obtained with and without the starbyte transformation. Circles on *b* and *c* show the location where microcalcifications are present in the target. Darker regions in the difference image correspond to lower difference values.

Figure 9.5a shows the absolute difference image between the reference and target. As can be seen, there is a large amount of misregistration error between the two. Figure 9.5b and c show the absolute difference images between the target and unwarped images obtained with and without the starbyte transformation. The circles drawn on Figure 9.5b and c show the region where we should see some microcalcifications if true registration had taken place. The microcalcifications, which are in fact, visible in the target image, are hardly visible in the difference images against the large amount of background misregistration noise. Also G , the root mean square error values per pixel between the target and reference, and the target and the unwarped images obtained with and without the starbyte transformation are 0.0023, 0.0026 and 0.0028 respectively. These numbers show that the misregistration error has increased after the unwarping process.

For the limited purpose of this thesis, and in order to demonstrate the applicability

of the starbyte image registration algorithm to real images, we now register pairs of MRI slices of two volunteers in the next section. Such an operation could be repeated for all slice pairs producing a rough 3D transformation. While imaging slices still is 2D in nature, it is better than registering projections. Since T depends on z as well, two slices taken from 3D data sets can never be truly corresponding because information present in one slice may not all be present in the other.

9.4 Results with slices of MRI 3D breast data sets

Two women were imaged twice using the MRI machine at the Institute of Biodiagnostics, National Research Council, Winnipeg. The MRI images were taken with a specialized MRI breast coil [19]. The two women were 35 and 36 years old. Images were taken with a 3 Tesla whole body Bruker system. The imaging parameters for the first volunteer were:

pulse sequence: MSME, FOV (Field of view): $14cm^2$, T_E (echo time):20ms, T_R (repetition time):1500ms, slice thickness:3mm, image size: 256 x 256.

The imaging parameters for the second volunteer were:

FLASH (Fast low angle single shot), FOV: $14cm^2$, T_E :7.4ms, T_R :1050ms, slice thickness:3mm, image size:256 x 256.

The women stepped into the MRI machine and a 3D image was taken. The women were then asked to step out of the machine and step back in and a second 3D image was taken. For the first woman, each 3D data set consisted of 15 slices. For the second woman, each data set consisted of 20 slices. The 256 x 256 image slices were downsampled to 128 x 128. A pair of roughly corresponding slices was selected by eye for each volunteer. An observation of the corresponding slices showed that they were misaligned. As has been pointed out earlier, this is because of the

non-rigid nature of breast tissue. While in this case, the two 3D images for each volunteer were taken within minutes of each other, in a real imaging situation, where the screening time between two images is typically of the order of several months or a year, the amount of misalignment between images would be even larger, because other than mere positional misalignment, there could also be changes in breast tissue, for example, from changes during the menstrual cycle, aging, etc. This experiment thus demonstrates the clear need for solving the image registration problem for breast imaging.

In the examples below, a 9×9 neighborhood size was used for UICP while a 15×15 neighborhood size on filtered original images was used for AUTOCP. This was because a higher neighborhood size renders larger regions which are more easily identifiable by an automatic program. While the same higher neighborhood ST could have been used for UICP too, a 9×9 neighborhood size was enough, as corresponding regions could be identified by eye. A moving average filter of 5×5 was used to filter the original images before computing the ST for AUTOCP. Protocol 3 was used in all cases.

9.4.1 Results with the MRI slice pair from the first volunteer

Figures 9.6a and b show approximately corresponding slices from the 3D data sets from the first volunteer. It is seen that the actual signal portion of the image is present only in the second half of the images. A histogram equalization of the entire image (not shown) revealed that the first half contained noise and echo. The signal portion of the images also contain a large amount of noise.

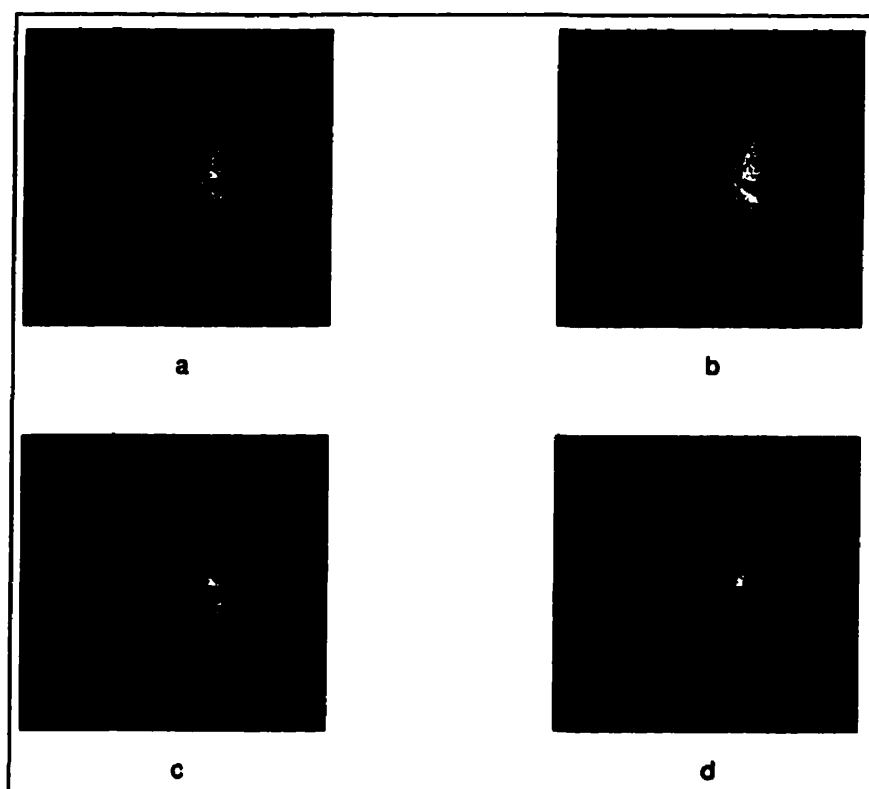


Figure 9.6: *a* and *b*: Corresponding slices taken from 3D MRI data sets of first volunteer. *c* and *d*: Unwarped images obtained with UICP and AUTOCP.

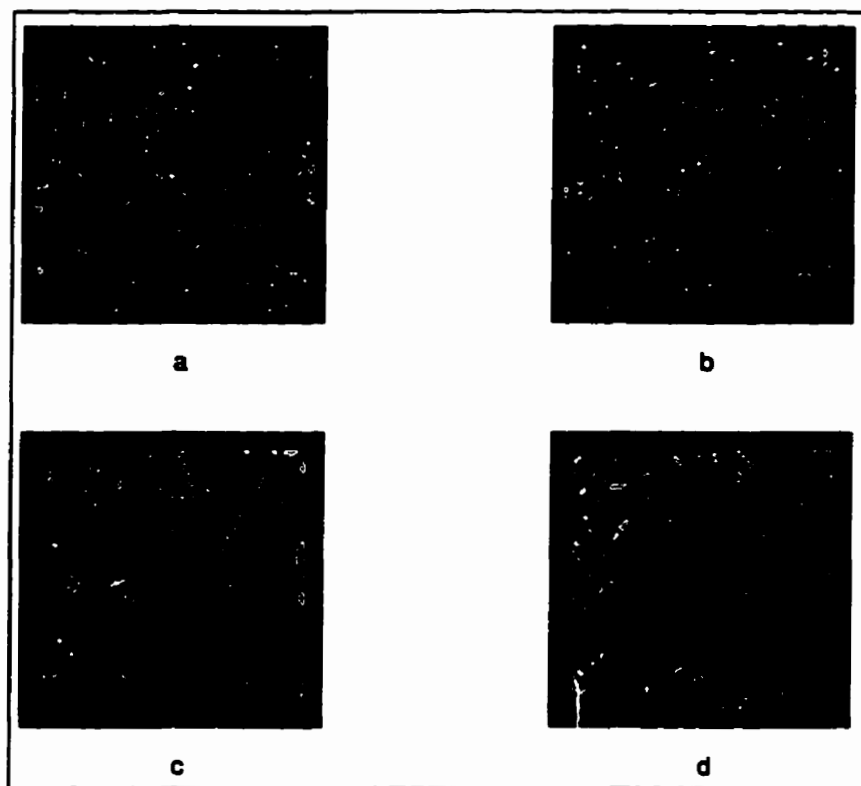


Figure 9.7: *a* and *b*: STs of the MRI slices from the first volunteer using a 9×9 neighborhood size. *c* and *d*: STs of filtered reference and target images using a 15×15 neighborhood size. The echo is apparent.

Figure 9.7a and b show the starbyte transformation for the original images for a 9 x 9 neighborhood size while Figure 9.7c and d show 15 x 15 neighborhood size STs for filtered versions of the original images. The noise in the original images is revealed in Figure 9.7a and b, where small regions have been formed corresponding to the noise in the original images in the first half of the STs while larger regions have been formed in the second half corresponding to the signal portion of the original images. Several of these smaller regions have been smoothened out and joined together in Figure 9.7c and d where a 15 x 15 ST was used on filtered original images. The regions on the right side of the image are also much larger.

UICP and one iteration of AUTOCP were run on the images. Figure 9.8a and b show 17 control points obtained using UICP superimposed on the original images, while Figure 9.9a and b show 6 control points obtained using AUTOCP superimposed on the original images. As can be seen visually, most of the points in both cases are accurate, but since we do not know the actual transformation relating the original images, we cannot present D and E diagrams here. Figure 9.6c and d show the unwarped images obtained using TPS-unwarping for the manual and automatic case.

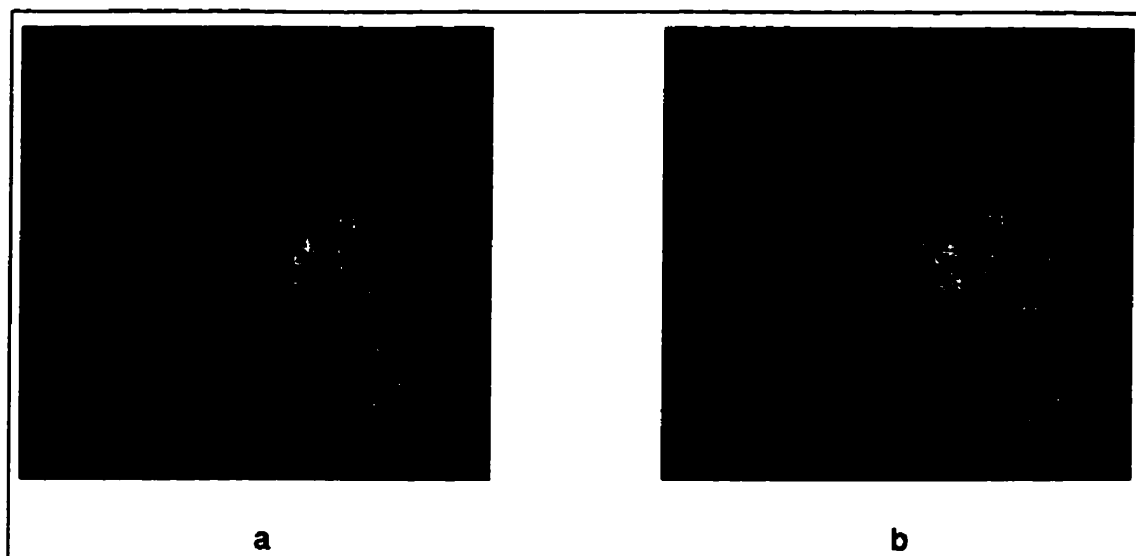


Figure 9.8: *a* and *b*: Reference and target images (corresponding MRI slices of first volunteer) with 17 control points obtained using UICP superimposed.

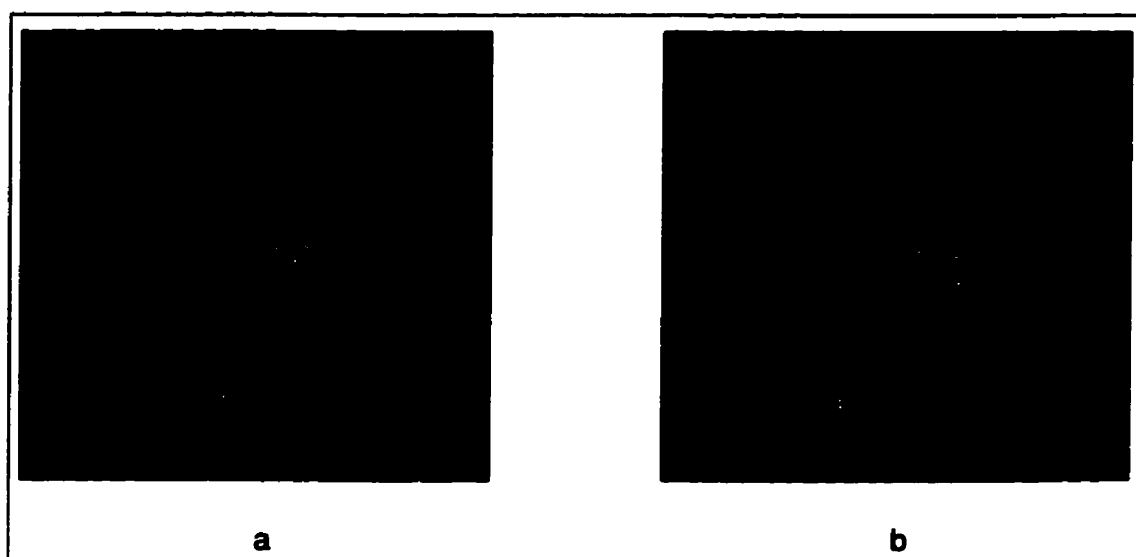


Figure 9.9: *a* and *b*: Reference and target images (corresponding MRI slices of first volunteer) with 6 control points obtained using AUTOCP superimposed.

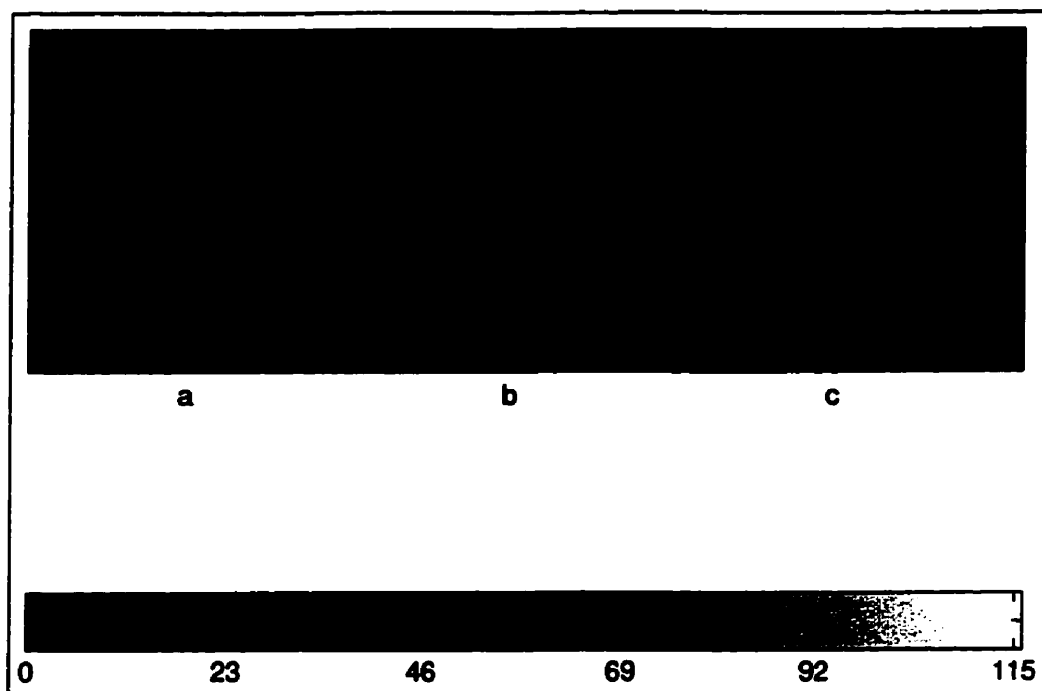


Figure 9.10: *a*: Absolute difference image between the reference and target for the MRI slice pair from the first volunteer. *b* and *c*: Absolute difference images between the target and unwrapped images for the manual and automatic cases.

Figure 9.10a shows the absolute difference image of the reference and target, while Figures 9.10b and c show the absolute difference image between the target and unwrapped images. It is clear that the registration error present in Figure 9.10a is largely removed after unwarping. However some misregistration error at the edge of the breast is still present.

9.4.2 Results with the MRI slice pair from the second volunteer

Figures 9.11a and b show approximately corresponding slices from the 3D data sets corresponding to the second volunteer. In this case, the signal portion of the image covers most of the image area. Even here, a large amount of noise is present. A close observation of the two images shows that some structures present in one are not present in the other. Figure 9.12a and b show the starbyte transformation for the original images for a 9 x 9 neighborhood size while Figure 9.12c and d show 15 x 15 neighborhood size STs for filtered versions of the original images.

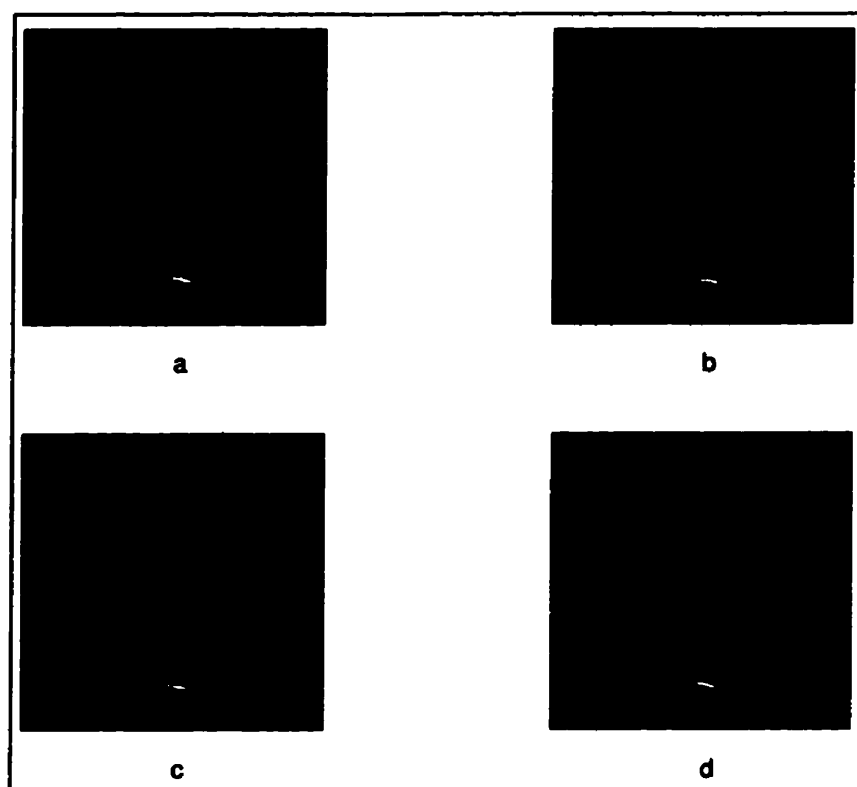


Figure 9.11: *a* and *b*: Corresponding slices taken from 3D MRI data sets of second volunteer. *c* and *d*: Unwarped images obtained with UICP and AUTOCP.

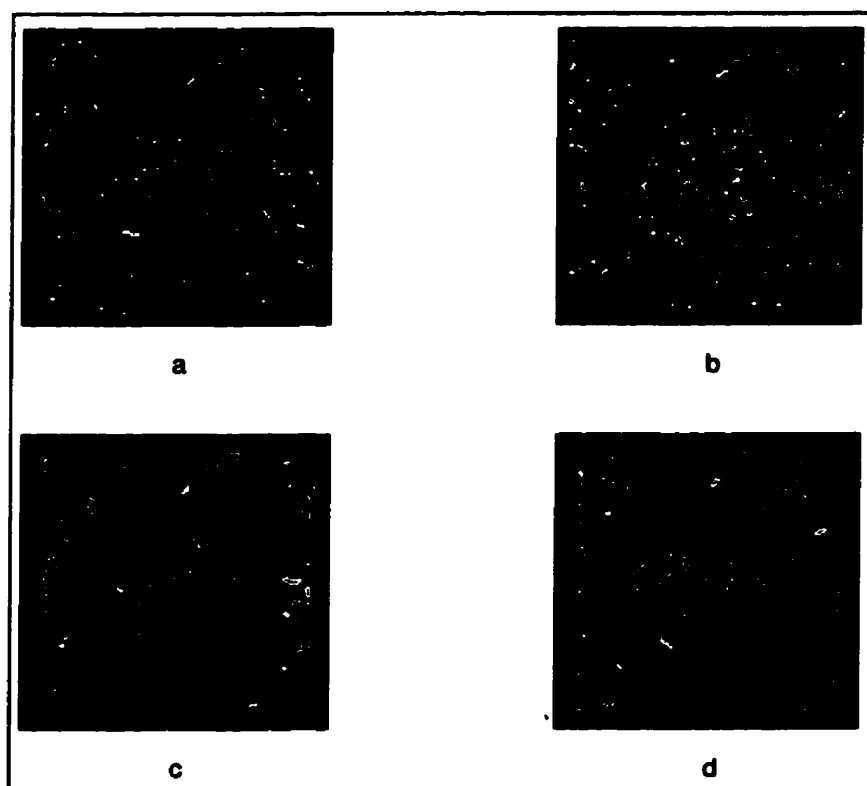


Figure 9.12: *a* and *b*: STs of the MRI slices from the second volunteer using a 9 x 9 neighborhood size. *c* and *d*: STs of filtered slices using a 15 x 15 neighborhood size.

UICP and one iteration of AUTOCP were run on the images. Figure 9.13a and b, and Figure 9.14a and b show 16 and 6 control points obtained using UICP and AUTOCP respectively, superimposed on the original images. As before, most of the control points in both cases look visually accurate.

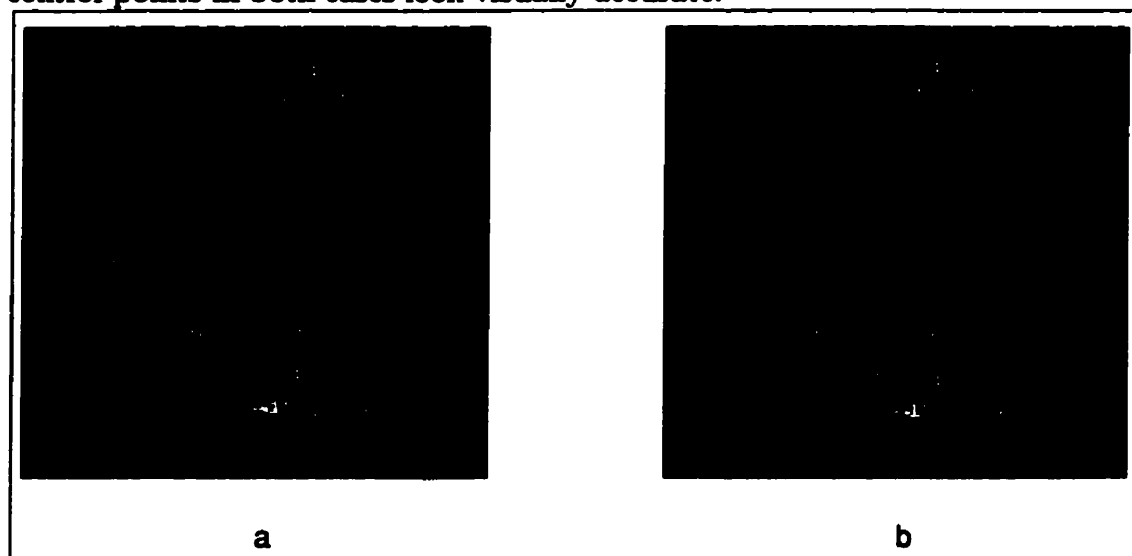


Figure 9.13: *a* and *b*: Reference and target images (corresponding MRI slices of second volunteer) with 16 control points obtained using UICP superimposed.

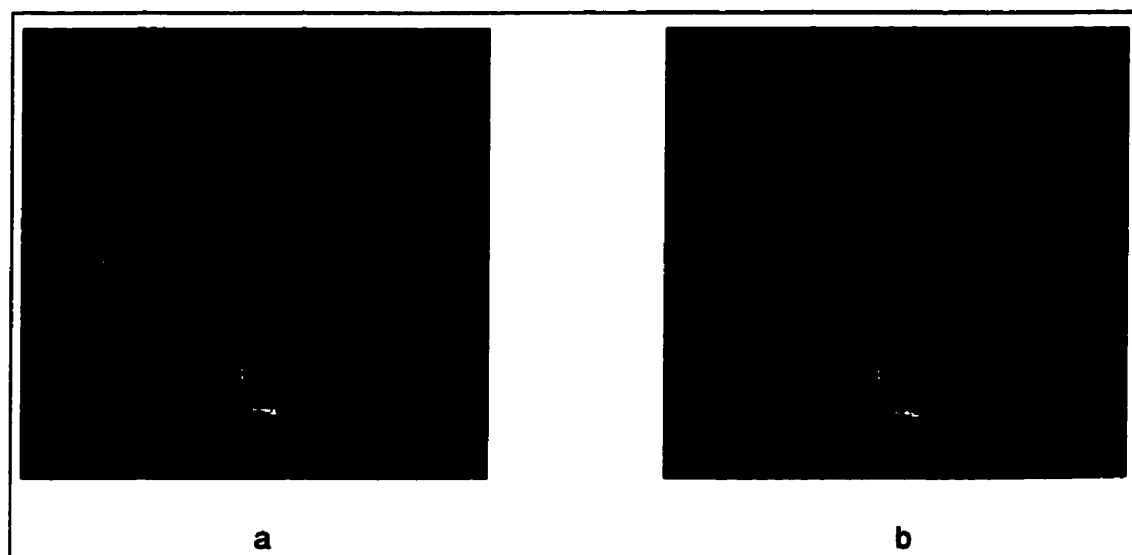


Figure 9.14: *a* and *b*: Reference and target images (corresponding MRI slices of second volunteer) with 5 control points obtained using AUTOCP superimposed.

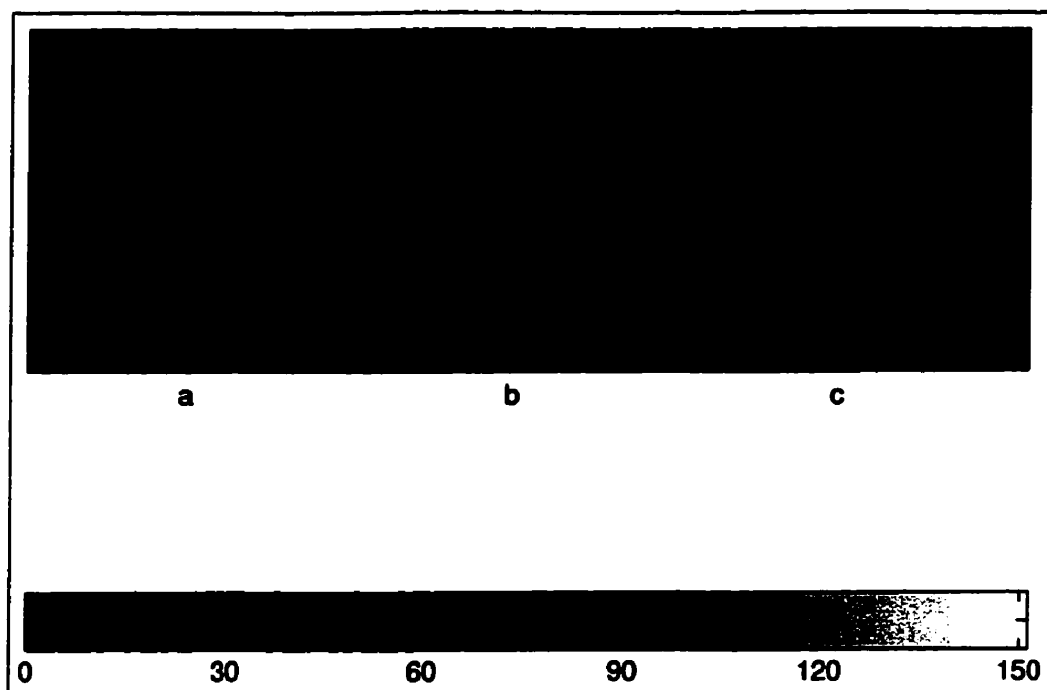


Figure 9.15: *a*: Absolute difference image between the reference and target for the MRI slice pair from the second volunteer. *b* and *c*: Absolute difference images between the target and unwarped images for the manual and automatic cases.

Figure 9.11c and d show the unwarped images obtained using TPS-unwarping for the manual and automatic case. Figure 9.15a shows the absolute difference image between the reference and target, while Figures 9.15b and c show the absolute difference image between the target and unwarped images. As in the previous case, a significant amount of misregistration error in Figure 9.15a is removed after unwarping.

9.5 Calculation of projection images

In order to show that matching projections rather than slices results in more inaccurate results because 3D information is superimposed on 2D, we have created artificial “projections” from the 3D data sets by adding corresponding pixels on all the slices in the data sets for the two volunteers. This is not a very accurate way of creating a projection image because the slice profile used in obtaining each slice was not strictly rectangular and profiles of consecutive slices overlap. However, the profile was approximately rectangular and is the inverse Fourier transform of a sinc pulse with 3 lobes, the slice thickness being the halfwidth of the sinc pulse. The pulses overlap at the half-width position. The effect of having a non-rectangular pulse can be expressed as follows: Assuming that projections are taken along the z direction, let $\hat{p}(x, y)$ be the projection value at the location (x, y) computed by simply adding corresponding values in all the slices together. If N slices have been taken, each of width w , then $\hat{p}(x, y)$ is given by

$$\hat{p}(x, y) = \sum_{n=1}^N \int_{(n-1)w}^{nw} P(x, y, z) S(x, y, z) dz \quad (9.2)$$

where $P(x, y, z)$ is the actual value at location (x, y, z) and $S(x, y, z)$ is a function related to the slice profile. If every slice has been taken using the same profile for the pulse, then $S(x, y, z)$ would be a function of only x and y . However, what we would ideally like to know is $p(x, y)$ which is given by

$$\begin{aligned} p(x, y) &= \sum_{n=1}^N \int_{(n-1)w}^{nw} P(x, y, z) dz \\ &= \int_0^{Nw} P(x, y, z) dz \end{aligned} \quad (9.3)$$

Knowing $\hat{p}(x, y)$ does not give us a direct way of estimating $p(x, y)$. However, since $S(x, y, z)$ is nearly rectangular, the error committed in computing $\hat{p}(x, y)$ as against $p(x, y)$ is assumed to be small.

9.5.1 Results with projection images from the first volunteer

Figures 9.16a and b show projection images corresponding to the first volunteer. Figure 9.17a and b show the starbyte transformation for the original images for a 9 x 9 neighborhood size while Figure 9.17c and d show 15 x 15 neighborhood size STs for filtered versions of the original images.

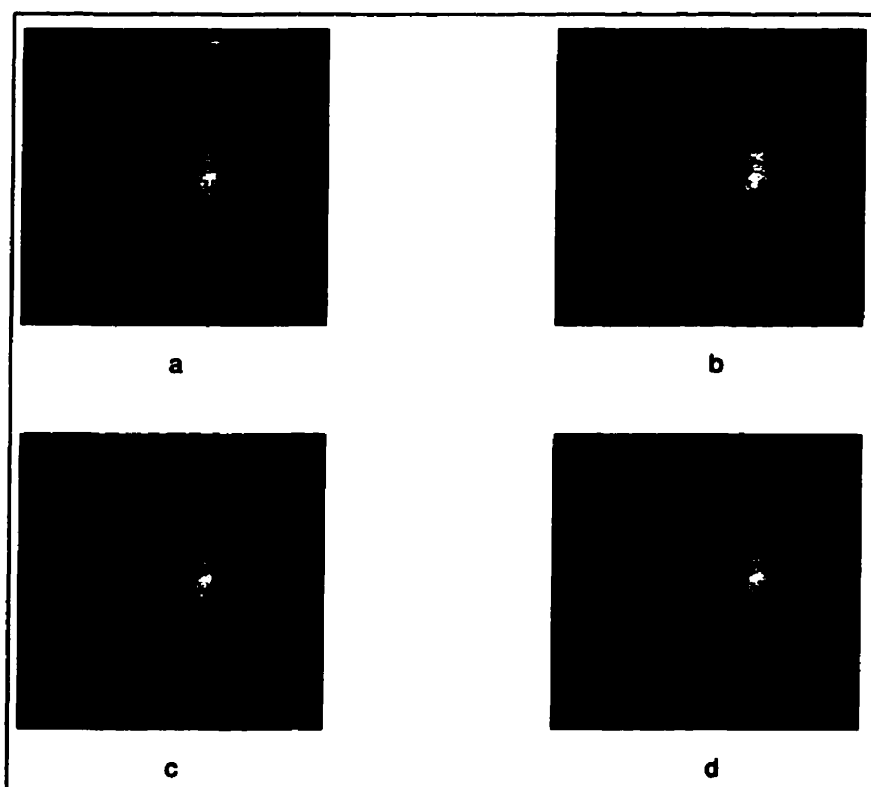


Figure 9.16: *a* and *b*: Projection images for the first volunteer. *c* and *d*: Unwarped images obtained with UICP and AUTOCP.

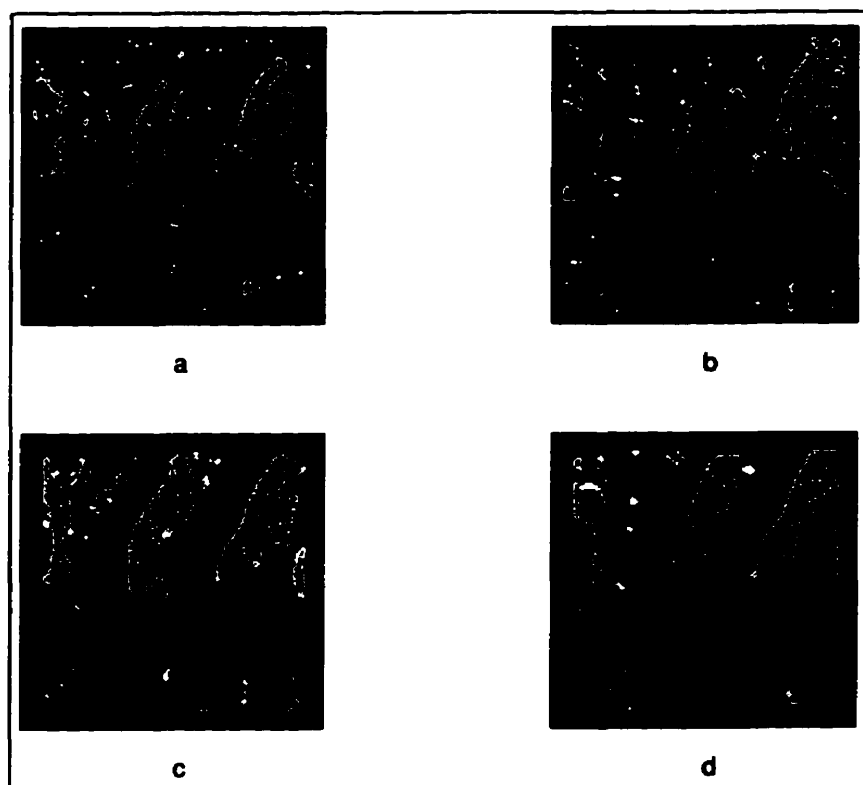


Figure 9.17: *a* and *b*: STs of the projections for the first volunteer using a 9 x 9 neighborhood size. *c* and *d*: STs of filtered projections using a 15 x 15 neighborhood size. Echos are again apparent.

UICP and one iteration of AUTOCP were run on the images. Figure 9.18a and b show 11 control points obtained using UICP superimposed on the original images, while Figure 9.19a and b show 3 control points obtained using AUTOCP superimposed on the original images. It is clear that a large number of points have been selected in the echo region. Figure 9.16c and d show the unwarped images obtained using TPS-unwarping for the manual and automatic case.

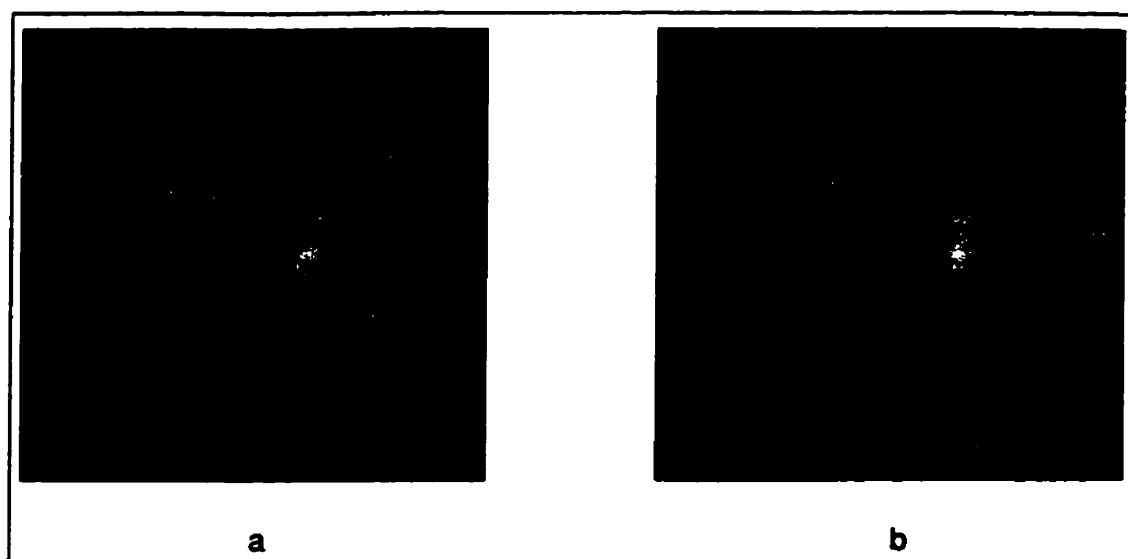


Figure 9.18: *a* and *b*: Projection images of first volunteer with 11 control points obtained using UICP superimposed.

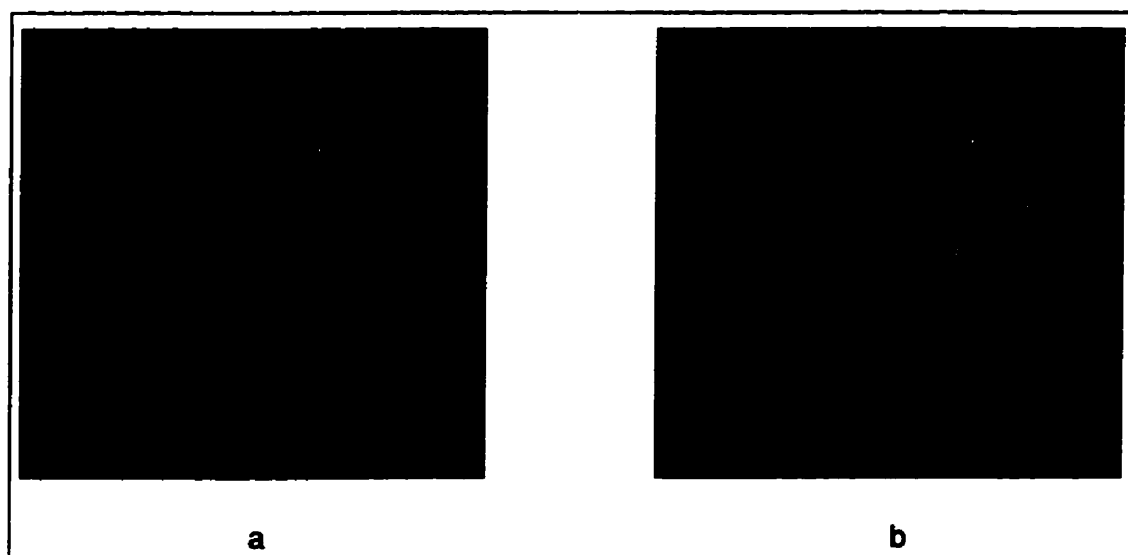


Figure 9.19: *a* and *b*: Projection images of first volunteer with 3 control points obtained using AUTOCP superimposed.

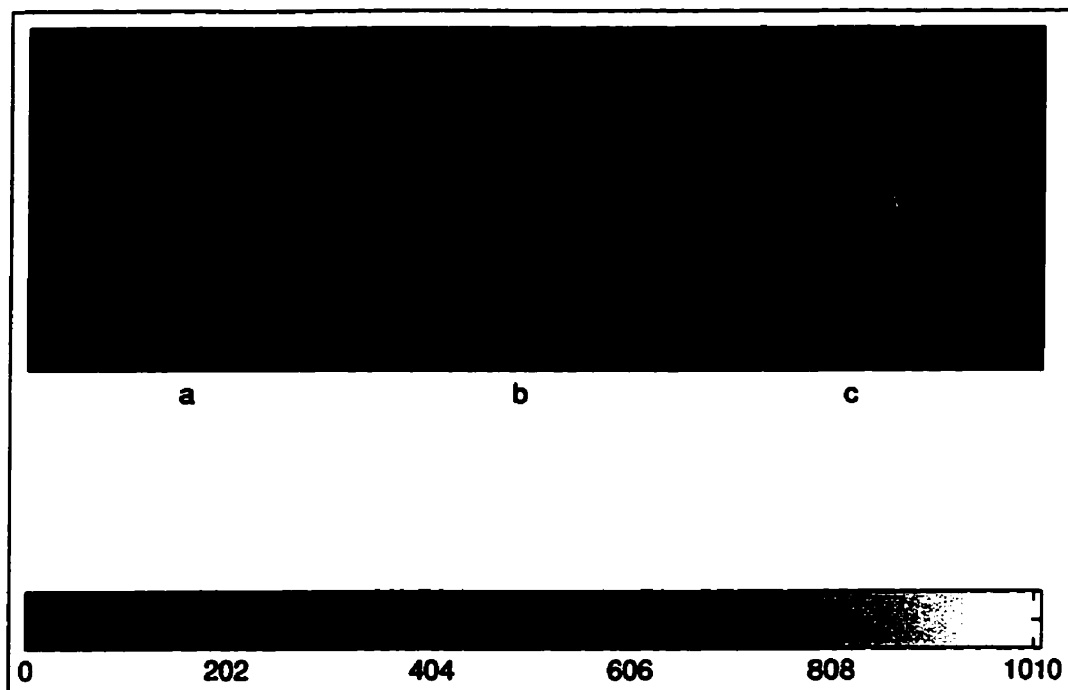


Figure 9.20: *a*: Absolute difference image between the reference and target image (projection images for the first volunteer). *b* and *c*: Absolute difference images between the target and unwarped images for the manual and automatic cases.

Figure 9.20a shows the absolute difference image of the reference and target. It is clear when comparing Figure 9.10a and Figure 9.20a that there is a larger amount of noise in Figure 9.20a. Figures 9.20b and c show the absolute difference image between the target and unwarped images. When comparing these difference images to the corresponding ones for the slice case, we see that it is clear that it was more difficult to register the projections because the initial registration error was much larger to start with. Also while the matching process has removed some of the misregistration error, there is still more misregistration error remaining after the unwarping process as compared to the slice case.

9.5.2 Results with projection images from the second volunteer

Figures 9.21a and b show projection images corresponding to the first volunteer. Figure 9.22a and b show the starbyte transformation for the original images for a 9 x 9 neighborhood size while Figure 9.22c and d show 15 x 15 neighborhood size STs for filtered versions of the original images.

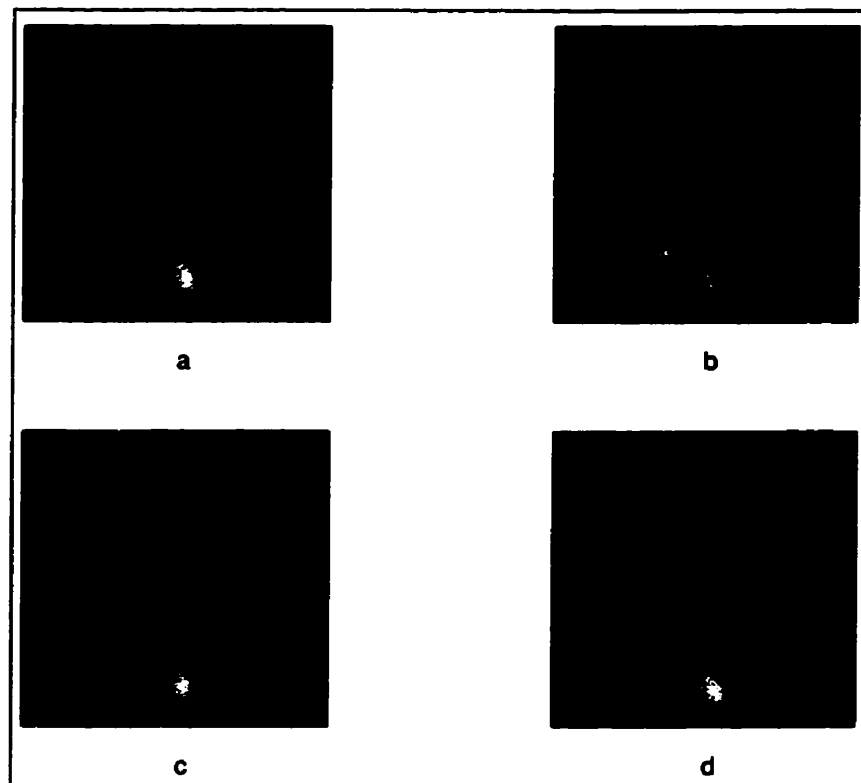


Figure 9.21: *a* and *b*: Projection images for the second volunteer. *c* and *d*: Unwarped images obtained with UICP and AUTOCP.

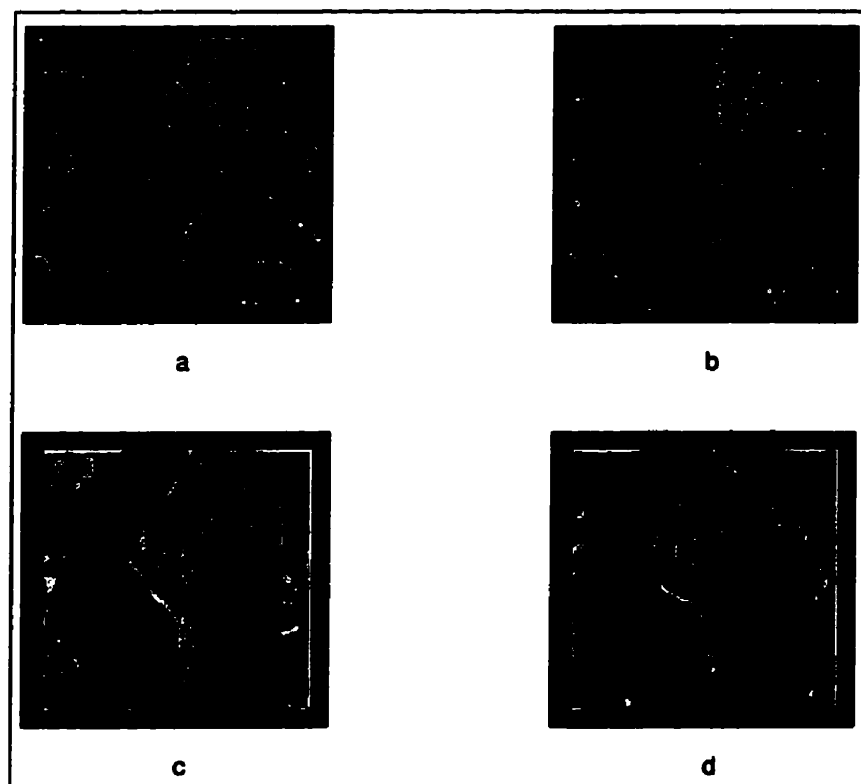


Figure 9.22: *a* and *b*: STs of the projections for the second volunteer using a 9×9 neighborhood size. *c* and *d*: STs of filtered projections using a 15×15 neighborhood size.

UICP and one iteration of AUTOCP were run on the images. Figures 9.23a and b show 13 control points obtained using UICP superimposed on the original images, while Figures 9.24a and b show 6 control points obtained using AUTOCP superimposed on the original images. Figures 9.21c and d show the unwarped images obtained using TPS-unwarping for the manual and automatic case.

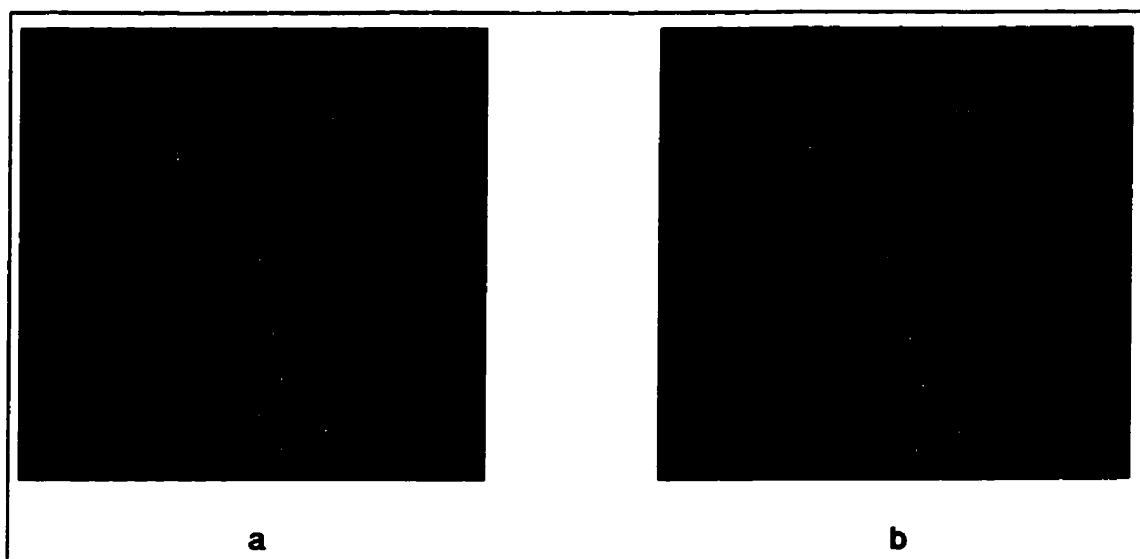


Figure 9.23: *a* and *b*: Projection images of second volunteer with 13 control points obtained using UICP superimposed.

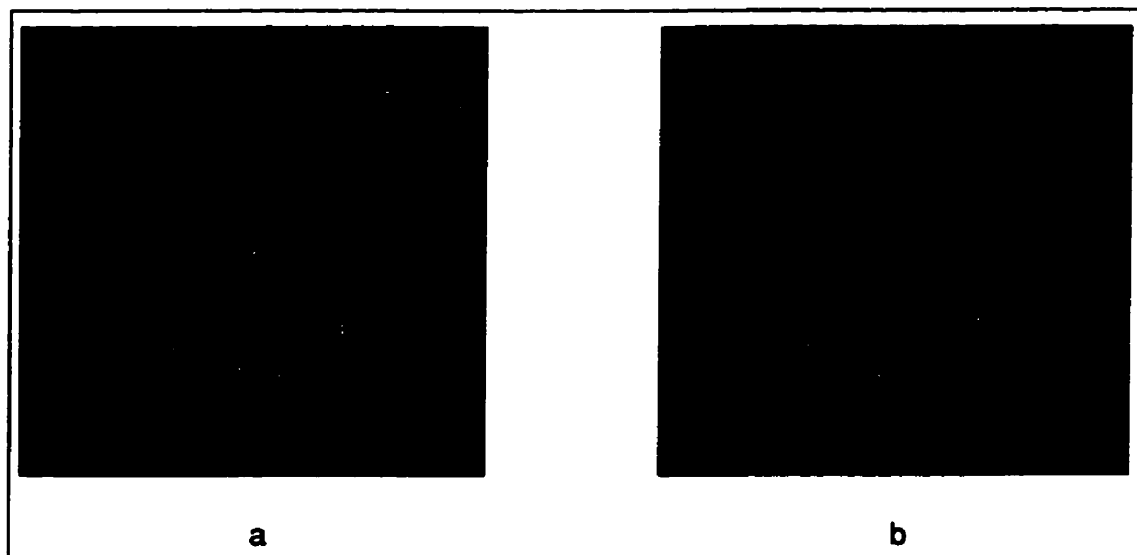


Figure 9.24: *a* and *b*: Projection images of second volunteer with 6 control points obtained using AUTOCPP superimposed.

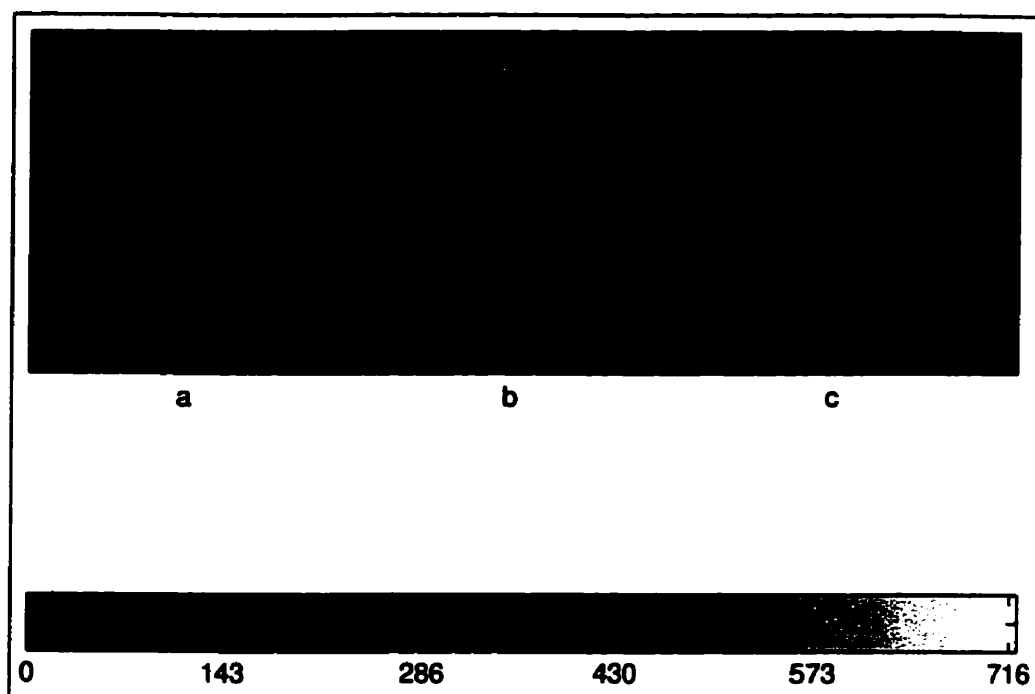


Figure 9.25: *a*: Absolute difference image between the reference and target image (projection images for the second volunteer). *b* and *c*: Absolute difference images between the target and unwarped images for the manual and automatic cases.

Figure 9.25a shows the absolute difference image between the reference and target. As before the amount of misregistration error is larger and hence these two images are more difficult to register than the slices. As before, when comparing these difference images to the corresponding ones for the slice case, it is clear that registration was better in the slice case.

Volunteer #	Slice			Projection		
	U	M	A	U	M	A
1	0.0058	0.0043	0.0050	0.0035	0.0038	0.0044
2	0.0074	0.0041	0.0051	0.0029	0.0017	0.0025

Table 9.1: *Column 1:* Volunteer number. *Column 2:* root mean square error per pixel for the slices for the unregistered case (U), using UICP (M) and AUTOCP (A). *Column 3:* root mean square error per pixel for the projections for the unregistered case (U), using UICP (M) and AUTOCP (A).

G values were calculated for the slices and the projections and are presented in Table (9.1). For both volunteers, the error values have decreased after unwarping for the slice case showing that registration has taken place. Since the images contain a large amount of noise, and since the slices are only approximately corresponding (i.e. the transformation between them is not fully planar), G values after unwarping cannot be significantly lower than the corresponding values before unwarping. For the first volunteer, this is also because of some remnant misregistration error at the boundary of the breast which increases the G value after unwarping. For the projections, it is seen that G increases after unwarping for the first volunteer. While G is lower after unwarping for the second volunteer, the decrease is not as good as for the corresponding slice case. This demonstrates that the slices were registered better than the corresponding projections.

However, it should be noted that while the slice registration experiments have fared better than the projection registration experiments, in a real situation one would expect the difference in performance to be much bigger than what was shown. The reasons for this are :

1. Positioning of the pendant breast in general is more accurate and reproducible than positioning of a compressed breast as is done in standard mammography.

Since in these experiments, the women were asked to step back into the machine almost immediately for the second image, the amount of distortion T itself is less (because factors such as a change in menstrual cycle, aging, etc. won't play a part here). Hence the amount of distortion along the z axis is also less. Thus every slice in this experiment undergoes approximately the same transformation. In a true registration experiment, where a volunteer returns for the second image after a long time interval, z would be larger and hence the results obtained by projection registration will be poorer than what was observed here.

2. The “projection” images used in these experiments are not true projections images and are only approximations. For example, in this case, adding all the slices has averaged out the noise to some extent. In a true projection, this may not be the case.

9.6 Conclusions

The need for 3D breast imaging was discussed in this chapter. We also showed by a “negative” result that registering pairs of mammograms can only have limited success because of the superimposition of 3D information onto 2D. In order to show the working of the starbyte registration algorithm on real images, we have also registered two pairs of approximately corresponding slices taken from 3D MRI data sets of two volunteers. We also showed that registration would have been poorer had projections been registered instead of slices. The next chapter outlines future work which can be done based on the work in this thesis.

Chapter 10

CONCLUSIONS AND FUTURE WORK

10.1 Conclusions

Image registration is an important problem in breast imaging. Matching of breast images directly is almost impossible because of the low contrast and noisy nature of the images and in standard mammography, because of the projection of 3D information onto a 2D plane.

This thesis is the solution of the image registration problem (with particular emphasis on breast images) using a textural transformation called the *starbyte transformation*. The starbyte transformation of a given image is generated using local neighborhood properties of pixels in the image. In this thesis, a textural value at a pixel was obtained as a bit string using binary comparisons of average grey values over different sectors in the local neighborhood of a pixel. However, the textural transformation can be done in several different ways. When applied to breast images, the starbyte transformation converts them to texture maps with clearly demarcable regions. Accurate control points can be easily identified as the centroid of these regions.

A complete registration algorithm was presented in the thesis, which consisted of first starbyte-transforming the original images, extracting control points, and performing unwarping using thin plate splines. The use of thin plate splines ensures that no assumptions are made on the type of distortion relating the original images.

A major portion of the thesis consisted in systematically studying the performance of the starbyte registration algorithm for six different distortions. Once the starbyte-transformed images are generated, control points can be identified manually as well as automatically. A manual and an automatic algorithm for identification of control points have been presented in detail with several examples. Limited studies with noise and artificial tumors have also been carried out.

The thesis also discusses an important issue, namely, the need for 3D breast imaging. Controlled preliminary studies with pairs of roughly corresponding slices taken from MRI 3D data sets of volunteers have been presented. We have also demonstrated that matching of temporal mammograms cannot give us any useful results because of the large amount of tissue overlap present in them.

An important feature of the image registration algorithm presented in this thesis is that it can be easily extended to the third dimension.

We outline below future work which can be done based on the work in this thesis.

10.2 Future Work

Future work can be done at several levels. We outline some of the possible extensions to the work in this thesis.

10.2.1 Generalization of the starbyte transformation concept

As discussed in chapter 2, the starbyte transformation can be generalized by considering other properties of the local neighborhood of a pixel. These properties can be other spatial properties using special spatial masks, frequency domain properties [40], textural properties [55] etc. Whether other textural transformations will generate useful regions for matching remains to be seen. An explanation of why some starbyte

transformations have this property is yet to be discovered.

10.2.2 Mathematical proof of the commutativity property of the starbyte transformation

The starbyte transformation is still heuristic in its present form, and hence its properties have been studied and learnt only through a large number of examples. Although, the intuitive explanation for why the starbyte transformation works, is provided in the Introduction chapter as well as in the chapter "The Starbyte Transformation", it would be interesting to see if the commutativity property of the transformation can be established mathematically. Since it is obvious that the commutativity property does not hold for all possible distortions (for e.g. the property breaks down if the image is rotated by 90 deg or more), it would be worthwhile to investigate for what range of distortions, this property holds, as well as to investigate how factors like image continuity affect this property.

10.2.3 Alternate methods of registration using the starbyte transformation

Since starbyte transformations result in texture maps having regions with easily identifiable boundaries, registration could also be performed by extracting the boundaries of these regions and matching them using active contours [22], [23] or local linear stretching [63], [104]. This could result in a larger number of control points. However, errors in the boundary profile could instead result in errors in the control points. The advantage of centroids, which we have used, is that they are insensitive to small errors in the region boundaries. The starbyte-transformations would still have to be cleaned with a morphological operation before control point selection to smooth the boundaries.

10.2.4 Use with other types of images

Future work should also involve testing the starbyte registration algorithm on other medical and non-medical images to see if the properties of the transformation are still preserved.

10.2.5 Improvements in the automatic procedure

The automatic procedure presented cannot be used for extremely large distortions. This is because of the use of the X and Y locations of the centroids, as well as the number of pixels in the parameter list. Improvements should be made to the automatic procedure so that more general shape descriptors can be used. This will allow the algorithm to be used for larger and more severe distortions. Standard image processing books like [40] describe several shape descriptors. In the present implementation, matches between regions are decided based only on properties of the particular regions. However, in the manual case, when clicking on matching regions, we inherently make use of neighboring regions to select matches. Hence the automatic procedure can also be modified so that neighboring regions can influence the selection process.

10.2.6 More extensive studies with complicated distortions, noise and artificial tumors

While studies have been performed in this thesis with a range of distortions, as well as with noise and artificial tumors, more extensive studies are required. Future study should particularly be directed towards the relationship between the commutativity property and the level of distortion. It would be worthwhile studying how commutation breaks down with a severe distortion. Extensive studies with breast

images of different textural characteristics should also be carried out.

In chapter 7, we said that apart from a reasonable model for the noise, we also have to obtain a realistic spatial correlation of the noise for x-ray images. While we performed noise studies with portions of mammograms, noise studies with breast images from other modalities will also have to be performed, in which case noise models appropriate to the particular modality have to be used. Apart from these simulation noise studies, other noise studies should include imaging the same object twice in the same modality, for example taking two X-ray images of breast tissue samples or two MRI images of the same person without any other changes like movement, etc., so that the only difference in the images is because of noise. The parameters of the imaging procedure can also be varied so that for a series of images, the noise in the images alone will be varied. This will enable us to study to what extent the starbyte registration algorithm works under different noise conditions.

Abnormalities were modeled as simulated tumors which were represented as changes in density level. Also the tumors were assumed to be symmetrical. This corresponds closest to a circumscribed mass. However, breast abnormalities can manifest themselves in many other different ways. Some of these include microcalcifications, ill-defined masses, architectural distortions, etc. Lefebvre [70] describe a way of adding simulated microcalcifications to mammograms. Mazur [78] has performed some registration studies with a small amount of architectural distortion. Hence, the performance of the starbyte registration algorithm should be evaluated with other types of breast abnormalities. Even with masses, it would be useful to do a systematic study by varying the radius r and the density level d of the tumor, to check at which point the algorithm breaks down. This will give us a quantitative idea about the limits of the algorithm. The limits of detection of small tumors need to be ascertained, and

compared to the sizes that need to be found for high cure rates [112]. Noise and abnormalities were studied separately. It would be useful to evaluate how the algorithm performs when noise and abnormalities are present together in the images.

10.2.7 Use with a parallel machine

After the starbyte-transformed images are generated, each density slice in the image pair is essentially treated independently. Hence the algorithm lends itself to use with a parallel machine. After the starbyte-transformed images are obtained, the morphological OC cleaning of individual density slices can be done in parallel. Also individual slice pairs can be dealt with in parallel to extract control points. However, this can obviously be done only for the automatic procedure. Still if the method is to be used for very large images, this could result in a significant reduction in time.

10.2.8 Improvements in the unwarping procedure

The unwarped images were calculated using TPS directly. However, this is computationally intensive. Evaluation algorithms for TPS are available which have dramatically reduced their computational complexity [91], [9], [8]. Powell [91] has developed an algorithm which tabulates TPS values on a very fine rectangular grid using smoothness and differentiability properties of the kernel function in the equation. Complete algorithmic details are given in [91].

Secondly, a property of the TPS is that corresponding points in the two images are mapped on exactly to each other. This can sometimes prove to be an undesirable property because errors in control points will translate to errors in unwarping. However it is possible to regularize the TPS equation so that the interpolation scheme in the original TPS equations can be replaced by an approximation scheme [98]. In this way control points with large errors will not unduly spoil the unwarping process.

The use of TPSs as mapping functions has an inherent assumption that the TPS equations model distortions in breast tissue well. While the TPS is based on concepts from mechanics, ideally the unwarping function should be obtained by building a constitutive model of breast tissue that contains its actual mechanics. This model should take into account the elasticity and viscosity of the various tissues present in the breast, and include terms to describe how these tissues deform under compression pressure and gravity, and during growth (weight gain or loss, lactation, and aging). This requires an extensive research program which involves empirically estimating the properties of breast tissue. There is no guarantee that the unwarping function obtained using this procedure would be significantly better than using TPS mapping functions. On the other hand, it may be well worthwhile to try it once to see how closely both transforms match one another.

10.2.9 Extension to 3D

We outlined the need for a 3D breast image registration algorithm in the previous chapter. Extension of the starbyte transformation procedure to 3D is not difficult. Since the starbyte values at each pixel location are based on averages taken over 2D neighborhoods, in the 3D case, the averages will be taken over 3D neighborhoods. Extension of the automatic procedure to 3D will not pose any major changes. Since each slice in 2D will be a volume image in 3D and every region in 2D will now be a three-dimensional region, suitable visualization software can be used, so that corresponding 3D regions in the two volume images can be selected by the user for the manual procedure. Image registration using 3D TPS mapping functions is already part of many applications [99].

Thus extension of the entire starbyte registration algorithm to 3D is possible. In the previous chapter, we have outlined the various options for 3D breast imaging. It

is clear that while some technical difficulties will have to be overcome for the present, 3D breast imaging is practicable. The breast cancer problem can be effectively cured, if we can routinely image small breast abnormalities with 3D imaging and detect them successfully using a robust registration algorithm like the starbyte algorithm.

References

- [1] J. Adams, C. Patton, C. Reader and D. Zamora, "Hardware for geometric warping," *Electronic Imaging*, April, 1984.
- [2] H. Akima, "On estimating partial derivatives for bivariate interpolation of scattered data" in *Rocky Mountain J. Math*, Vol. 14, pp. 41-52, 1984.
- [3] Y. Amit, "A non-linear variational problem for image matching," *SIAM J. Sci. Comput.*, vol. 15, no. 1, pp. 207-224, 1994.
- [4] L. H. Baker, Breast cancer detection demonstration project: five year summary report, *CA Cancer J. Clin* vol 42, pp. 194-225, 1982.
- [5] D. I. Barnea and H. F. Silverman, "A class of algorithms for fast digital registration," *IEEE Trans. Comput.*, Vol. C-21, pp. 179-186, 1972.
- [6] R. E. Barnhill, "Representation and approximation of surfaces," in *Mathematical Software III*, J. R. Rice (Ed.), Academic Press, New York, pp. 69-120, 1977.
- [7] H. H. Barrett and W. Swindell, "Radiological imaging, the theory of image formation, detection and processing," *Academic Press*, vol. 2, 1981.
- [8] I. Barrodale, D. Skea, M. Berkley, R. Kuwahara and R. Poecker, Warping digital images using thin plate splines, *Pattern Recognition*, vol. 26, pp. 375-376, 1993.
- [9] R. K. Beatson and G. N. Newsam, "Fast evaluation of radial basis functions: I," *Comp. Math. Applications*, Vol. 24, Iss. 12, pp. 7-19, 1992.

- [10] M. Bern and D. Eppstein, "Mesh generation and optimal triangulation," *Technical Report CSL-92-1*, Xerox PARC, 1992.
- [11] M. Bohm-Velez and E. B. Mendelson, "Computed tomography, duplex Doppler ultrasound, and magnetic resonance imaging in evaluating the breast," *Semin. Ultrasound CT MR*, vol. 10, pp. 171-176, 1989.
- [12] F. L. Bookstein, "Principal warps: thin-plate splines and the decomposition of deformations", *IEEE Trans. Pattern Analysis Mach. Intell.*, vol. 11, pp. 567-585, 1989.
- [13] L. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, Vol. 24, no. 4, pp. 325-376, 1992.
- [14] P. C. Bunch, K. E. Huff and R. Van Metter, "Sources of noise in high-resolution screen-film radiography," *Proc. SPIE*, vol. 626, pp. 63-71, 1986.
- [15] Canadian Cancer Statistics 1992. National Cancer Institute of Canada, Toronto, Canada.
- [16] H. P. Chan, C. J. Vyborny, H. MacMahon, C. E. Metz, K. Doi, E. A. Sickles, "Digital mammography: ROC studies of the effects of pixel size and unsharp-mask filtering on the detection of subtle microcalcifications," *Invest. Radiol.*, vol. 22, pp. 581-589, 1987.
- [17] C. H. J. Chang, J. L. Sibala, J. H. Gallagher, R. C. Riley, A. W. Templeton, P. V. Beasley, and R. A. Porte, "Computed tomography of the breast: a preliminary report," *Radiology*, vol. 124, pp. 827-829, 1977.
- [18] C. H. J. Chang, J. L. Sibala, S. L. Fritz, S. J. Dwyer III, A. W. Templeton, F. Lin, and W. R. Jewell, "Computed tomography in detection and diagnosis of

- breast cancer," *Cancer*, vol. 46, pp. 939-946, 1980.
- [19] X. Chen, "A Shielded MRI Breast-Coaxial Radio-Frequency Coil," M.Sc. Thesis, Department of Physics, University of Manitoba, Winnipeg, Canada, 1997.
 - [20] Coreldraw Version 3.0, Corel Corporation, Ottawa, Ontario, Canada.
 - [21] N. Dash, A. R. Lupetin, R. H. Daffner, Z. L. Deeb, R. J. Sefczek, and R. L. Schapiro, "Magnetic resonance imaging in the diagnosis of breast diseases," *Am. J. Roentgenol.*, vol. 146, pp. 119-125, 1986.
 - [22] C. A. Davatzikos and J.L. Prince, "An active contour model for mapping the cortex," *IEEE Trans. Med. Imaging*, vol. 14, pp. 65-80, 1995.
 - [23] C. Davatzikos, J. L. Prince, R. N. Bryan, "Image registration based on boundary mapping," *IEEE Trans Med. Imaging*, vol. 15, pp. 112-115, 1996.
 - [24] A. P. Dhawan, R. Gordon, and R. M. Rangayyan, "Nevoscopy: three dimensional computed tomography for nevi and melanomas in situ by trasniluumination," *IEEE Trans. Med. Imaging*, vol. MI-3, pp. 54-61, 1984.
 - [25] A. P. Dhawan, G. Buelloni and R. Gordon, "Enhancement of mammographic features by optimal adaptive neighborhood image processing," *IEEE Trans. Medical Imaging MI-5(1)*, vol. 120, pp. 8-15, 1986.
 - [26] Dictionary of Computing, E. L. Glaser, I. C. Pyle (Consultant Editors), V. Illingworth (General Editor), Second Edition, Oxford University Press, New York, 1986.
 - [27] F. A. Dilmanian, "Computed tomography with monochromatic x rays," *Amer. J. Physiol. Imaging*, vol. 3/4, pp. 175-193, 1992.

- [28] F. A. Dilmanian, X. Y. Wu, E. C. Parsons, B. Ren, J. Kress, T. M. Button, L. D. Chapman, J. A. Coderre, F. Giron, D. Greenberg, D. J. Krus, Z. Liang, S. Marcovici, M. J. Petersen, C. T. Roque, M. Shleifer, D. N. Slatkin, W. C. Thomlinson, K. Yamamoto and Z. Zhong, "Single- and dual-energy CT with monochromatic synchrotron x-rays," *Phys. Med. Biol.*, vol. 42, pp. 371-387, 1997.
- [29] C. J. D'Orsi and D. B. Kopans, "Mammographic feature analysis," *Seminars in Roentgenology*, vol. XXVIII, pp. 204-230, 1993.
- [30] N. Dyn, D. Levin and S. Rippa, "Numerical procedures for surface fitting of scattered data by radial functions," *SIAM J. Sci. Comput.*, vol. 7, pp. 639-659, 1986.
- [31] R. L. Egan, M. B. McSweeney and F. B. Murphy, "Breast sonography and the detection of cancer," *Recent Results Cancer Res.*, vol. 90, 90, 1984.
- [32] J. Flusser, "An adaptive method for image registration," *Pattern Recognition*, vol. 25, pp. 45-54, 1992.
- [33] J. Flusser and T. Suk, "A moment-based approach to registration of images with affine geometric distortion," *IEEE Trans. Geoscience & Remote Sensing*, Vol. 32, No. 2, pp. 382-387, March 1994.
- [34] G. E. Forsythe, M. A. Malcom and C. B. Moler, *Computer methods for mathematical computations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.
- [35] R. Franke, "A critical comparison of some methods for interpolation of scattered data," *Technical Report NPS-53-79-003*, Naval Postgraduate School, Monterey, 1979.

- [36] R. Franke, "Smooth interpolation of scattered data by local thin plate splines," NPS-53-81-002, Naval Postgraduate School, Monterey, 1981.
- [37] R. Franke, "Smooth interpolation of scattered data by local thin plate splines," *Comp. & Math. with Appl.*, Vol. 8, no. 4, pp. 273-281, 1982.
- [38] C. R. Giardina and E. R. Dougherty, "Morphological Methods in Image and Signal Processsing," Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
- [39] J. J. Gisvold, D. F. Reese and P. R. Karsell, "Computed tomographic mammography (CTM), *Am. J. Roentgenol.*, vol. 133, pp.1143-1149, 1979.
- [40] R. C. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, Reading, Massachusetts, 1990.
- [41] R. Gordon, "Dose reduction in computerized tomography", (guest editorial), *Invest. Radiol.*, vol. 11, pp. 508-517, 1976.
- [42] R. Gordon and R. M. Rangayyan, "Geometric deconvolution: a meta-algorithm for limited view computed tomography," *IEEE Trans. Biomed. Eng.*, vol. BME-30, pp. 806-810, 1983.
- [43] R. Gordon, R. M. Rangayyan, "Feature enhancement of film mammograms using fixed and adaptive neighborhoods," *Applied Optics*, vol. 23, pp. 560-564, 1984.
- [44] R. Gordon, "Toward robotic x-ray vision: new directions for computed tomography," *Applied Optics*, vol. 24, pp. 4124-4133, 1985.
- [45] R. Gordon, "Evolution escapes rugged fitness landscapes by gene or genome doubling: the blessing of higher dimensionality", *Computers and Chemistry*, vol. 18, pp. 325-332, 1994.

- [46] A. Goshtasby, "Piecewise linear mapping functions for image registration," *Pattern Recognition*, vol. 19, pp. 459-466, 1986.
- [47] A. Goshtasby, G. C. Stockman and C. V. Page, "A region-based approach to digital image registration with subpixel accuracy," *IEEE Trans. Geosci. Remote Sensing*, vol. 24, pp. 390-399, 1986.
- [48] A. Goshtasby, "Piecewise cubic mapping functions for image registration," *Pattern Recognition*, vol. 20, pp. 525-533, 1987.
- [49] A. Goshtasby, "Registration of images with geometric distortions," *IEEE Trans. Geosci. Remote Sensing*, vol. 26, pp. 60-64, 1988.
- [50] A. Goshtasby, "Image registration by local approximation," *Image Vision Computing*, Vol. 6, Iss. 4, pp. 255-261, 1988.
- [51] R. L. Grimsdale, F. H. Sumner, C. J. Tunis and T. Kilburn, "A system for the automatic recognition of patterns," *Proc. IEE*, vol. 106B, pp. 210-221, 1959.
- [52] H. Guan and R. Gordon, "A projection access order for speedy convergence of ART (Algebraic Reconstruction Technique): a multilevel scheme for computed tomography," *Phys. Med. Biol.*, vol. 39, pp. 2005-2022, 1994.
- [53] R. W. Hamming, "Coding and information theory," Prentice-Hall, Englewood Cliffs, New Jersey, 1980.
- [54] R. M. Haralick, S. R. Sternberg and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. PAMI*, vol. PAMI-9, no. 4, pp. 532-550, 1987.
- [55] R. M. Haralick and L. G. Shapiro, *Computer and robot vision*, Addison-Wesley, Reading, Massachusetts, 1992.

- [56] L. S. Heuser, J. S. Spratt, J. G. Kuhns, A. F. C. Chang, H. C. Polk, J. B. Buchanan, "The association of pathologic and mammographic characteristics of primary human breast cancers with "slow" and "fast" growth rates and with axillary lymph node metastases," *Cancer*, vol. 53, pp. 96-98, 1984.
- [57] M. K. Hu, "Pattern recognition by moment invariants," *Proc. IRE (Correspondence)*, vol. 49, pp. 1428, Sept. 1961.
- [58] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179-187, 1962.
- [59] V. P. Jackson, R. E. Hendrick, S. A. Feig and D. B. Kopans, "Imaging of the radiographically dense breast," *Radiology*, vol. 188, pp. 297-301, 1993.
- [60] G. James and R. C. James, *Mathematics Dictionary*, 4th ed., Van Nostrand Reinhold Co., New York, 1976.
- [61] N. S. Jayant, *Digital coding of waveforms: principles and applications to speech and video*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.
- [62] P. C. Johns, D. J. Drost, M. J. Yaffe and A. Fenster, "Investigation of dual-energy radiography for breast cancer detection," *Med. Phys.*, vol. 11, 391, 1984.
- [63] C. Kambhamettu, "Curvature-based approach to point correspondence in recovery in nonrigid motion," Master's thesis, Department of Computer Science and Engineering, University of South Florida, 1991.
- [64] N. Karssemeijer, Department of Radiology, University of Nijmegen, the Netherlands, private communication.
- [65] Kauffman, S A *Origins of Order: Self-Organization and Selection in Evolution*, New York: Oxford University Press, 1994.

- [66] D. B. Kopans, "What is a useful adjunct to mammography?" *Radiology*, vol. 161, 560, 1986.
- [67] K. I. Laws, "Rapid texture identification," *Proceedings of the SPIE Conference on Image Processing for Missile Guidance*, San Diego, pp. 376-380, 1980.
- [68] C. L. Lawson, "Software for C^1 surface interpolation," In *Mathematical Software III*, J. R. Rice (Ed.), New York, Academic Press, pp. 161-194, 1977.
- [69] D. G. Leckie, "Use of polynomial transformations for registration of airborne digital line scan image," *Proc. Fourteenth Int. Sym. Remote Sensing Environment*, pp. 635-641, 1980.
- [70] F. Lefebvre, H. Benali, R. Gilles and R. D. Paola, "A simulation model for clustered breast microcalcifications," *Med. Phys.*, vol. 21, iss 12, pp. 1865-874, 1994.
- [71] S. Maitra, "Moment invariants," *Proc. of the IEEE*, Vol. 67, No. 4, pp. 697-699, 1979.
- [72] FrameMaker Version 4, Frame Technology Corporation, San Jose, California.
- [73] P. Maragos and R. Shaffer, "Morphological systems for multidimensional signal processing," *Proc. IEEE*, Vol. 78, no. 4, April 1990.
- [74] J. Martin, "Telecommunications and the computer," Second Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.
- [75] J. E. Martin, "Breast imaging techniques: mammography, ultrasonography, computed tomography, thermography and transillumination," *Radiol. Clin. North Am.*, vol. 21, 149, 1983.

- [76] Matlab Version 4.2c, The Mathworks Inc., Natick, Massachusetts.
- [77] Mathematica Version 3.0, Wolfram Research, Champaign, Illinois.
- [78] A. K. Mazur, *Image correlation technique for recovering deformation fields from pictures*, Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 1992.
- [79] A. K. Mazur, E. J. Mazur and R. Gordon, "Digital differential radiography (DDR): a new diagnostic procedure for locating neoplasms, such as breast cancers, in soft, deformable tissues," *SPIE*, vol. 1905, pp. 443-455, 1993.
- [80] P. W. Mcowan and T. W. Redpath, "A specialized receiver coil for NMR imaging of female breasts," *Phys. Med. Biol.*, vol. 32, pp. 259-263, 1987.
- [81] Morph version 4.0 (Image morphology program) by Richard Alan Peters II, Department of Electrical and Computer Engineering, Vanderbilt University School of Engineering, Nashville, anonymous ftp site: 129.59.100.16 (directory: /pub).
- [82] W. M. Morrow, R. B. Paranjape, R. M. Rangayyan, J. E. L. Desautels, "Region-based contrast enhancement of mammograms," *IEEE Trans. Med. Imaging*, vol. 11, pp. 392-406, 1992.
- [83] M. Moskowitz, Breast Imaging, *Cancer of the breast 4th ed.*, W. B. Saunders Company (Editors: W. L. Donegan, J. S. Spratt), 1995.
- [84] J. W. Muller, P. F. Van Waes, P. R. Koehler, "Computed tomography of breast lesions: comparison with x-ray mammography," *J. Computer Assisted Tomography*, vol. 7, pp. 650-654, 1983.
- [85] R. M. Murugan, "The limits of resolution of 3d electrical impedance tomography, with applications to joint effusions and breast cancer," Ph.D. Thesis, Department

of Electrical and Computer Engineering, University of Manitoba, Winnipeg, in preparation.

- [86] Nijmegen Digital Mammogram Database, anonymous ftp. site: figment.csee.usf.edu (Directory: pub/mammograms/nijmegen-images).
- [87] R. Nishikawa, Department of Radiology, University of Chicago, personal communication.
- [88] A. Papoulis, *Probability, random variables, and stochastic processes*, McGraw-Hill, New York, 1965.
- [89] R. B. Paranjape, R. M. Rangayyan, and W. M. Morrow, "Adaptive neighborhood mean and median image filtering," *J. Electronic Imaging*, Vol. 3, no. 4, pp. 360-367, 1994.
- [90] Richard Alan Peters II, Department of Electrical and Computer Engineering, Vanderbilt University School of Engineering, personal communication.
- [91] M. J. D. Powell, "Tabulation of thin plate splines on a very fine two-dimensional grid," *Report NL DAMTP 1992/NA2*, University of Cambridge, Cambridge, UK, 1992.
- [92] W. K. Pratt, *Digital image processing*, John Wiley and sons, 1992.
- [93] R. J. Prokop and A. P. Reeves, "A survey of moment-based techniques for unoccluded object representation and recognition", *CVGIP: Graphical Models and Image Processing*, Vol. 54, no. 5, pp. 438-460, 1992.
- [94] R. M. Rangayyan, L. Shen, R. B. Paranjape, J. E. L. Desautels, J. H. MacGregor, H. F. Morrish, P. Burrowes, S. Share, F.R. MacDonald, "An ROC evaluation

- of adaptive neighborhood contrast enhancement for digitized mammography," *Digital Mammography*, Elsevier Science B. V. (Editors: A. G. Gale, S. M. Astley, D. R. Dance, A. Y. Cairns), Amsterdam, The Netherlands, pp. pp. 307-313, 1994.
- [95] V. Raptopoulos, J. K. Baum, M. Hochman, A. Karellas, M-J Houlihan and C. J. D'Orsi, "High resolution CT mammography of surgical biopsy specimens," *J. Computer Assisted Tomography*, vol. 20, Iss. 2, pp. 179-184, 1996.
- [96] G. Revesz, H. L. Kundel and M. A. Graber, "The Influence of Structured Noise on the Detection of Radiologic Abnormalities," *Invest. Radiol.* vol. 9, 474, 1974.
- [97] K. Richter, "Clinical amplitude/velocity reconstructive imaging (CARI) - a new sonographic method for detecting breast lesions," *Br. J. Radiol.*, vol. 68, pp. 375-384, 1995.
- [98] K. Rohr, H. S. Stiehl, R. Sprengel, W. Beil, R. M. Buzug, J. Weese and M. H. Kuhn, "Point-based elastic registration of medical image data using approximating thin-plate splines," *Proc. 4th Internat. Conf. Visualization in Biomedical Computing (VBC '96)*, Hamburg, Germany, *Lecture Notes in Computer Science* 1131, K. H. Höhne and R. Kikinis (Eds.), Springer Berlin Heidelberg, pp. 297-306, 1996.
- [99] K. Rohr, H. S. Stiehl, R. Sprengel, W. Beil, R. M. Buzug, J. Weese and M. H. Kuhn, "Landmark-based elastic matching of tomographic images," *Proc. Freiburger Workshop 1997: Digitale Bildverarbeitung in der Medizin*, Freiburg, Germany, B. Arnolds, H. Müller, D. Saupe and T. Tolxdorff (Eds.), Albert-Ludwigs-Universitat Freiburg, pp. 163-168, March 10-11 1997.
- [100] R. J. Ross, J. S. Thompson, K. Kim and R. A. Bailey, "Nuclear magnetic

- resonance imaging and evaluation of human breast tissue: preliminary clinical trials," *Radiology*, vol. 143, pp. 195-205, 1982.
- [101] A. Rosenfeld and A. C. Kak, *Digital Picture Processing. Vol I and II.*, Academic Press, Orlando, Florida, 1982.
- [102] V. M. Runge, J. A. Clanton, C. M. Lukehart, C. L. Partain and A. E. James, Jr., "Paramagnetic agents for contrast-enhanced NMR imaging: a review," *Am. J. Roentgenol.*, vol. 141, 1209, 1983.
- [103] M. Sabel and H. Aichinger, "Recent developments in breast imaging," *Phys. in Med. and Biol.*, vol. 41, pp. 315-368, 1996.
- [104] M. Sallam, "Image unwarping and difference analysis: a technique for detecting abnormalities in mammograms," Ph.D. Thesis, Department of Computer Science and Engineering, University of South Florida, 1997.
- [105] L. L. Schumaker, "Fitting surfaces to scattered data," in *Approximation Theory II*, edited by G. G. Lorentz, C. K. Chui and L. L. Schumaker, Academic Press, New York, pp. 203-268, 1976.
- [106] J. Serra, *Image Analysis and mathematical morphology*, Academic Press, London, 1982.
- [107] L. Shen, Y. Shen, R. M. Rangayyan, J. E. L. Desautels, H. Bryant, T. J. Terry, N. Horeczko, "Earlier detection of interval breast cancers with adaptive neighborhood contrast enhancement of mammograms," *SPIE*, vol. 2710, 940, 1996.
- [108] D. Shepard, "A two-dimensional interpolation function for irregularly spaced data," *Proceedings ACM National Conference*, pp. 517-524, 1964.

- [109] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator", First Workshop on Applied Computational Geometry, ACM, pp. 124-133, PA, 1996.
- [110] E. A. Sickles, P. L. Davis and L. E. Crooks, "NMR characteristics of benign and malignant breast tissues: preliminary report," *Radiology*, vol. 149, 92, 1983.
- [111] E. A. Sickles, "Computed tomography scanning, transillumination and magnetic resonance imaging of the breast," *Recent Results Cancer Res.*, vol. 105, 31, 1987.
- [112] R. Sivaramakrishna and R. Gordon, "Detection of breast cancer at a smaller size can reduce the likelihood of metastatic spread. A quantitative analysis," *Academic Radiology*, vol. 4, pp. 8-12, 1997.
- [113] K. T. Smith and F. Keinert, "Mathematical foundations of computed tomography," *Appl. Opt.*, vol. 24, 3950, 1985.
- [114] D. Steiner and M. E. Kirby, "Geometrical referencing of Landsat images by affine transformation and overlaying of map data," *Photogrammetria*, vol. 33, pp. 41-75, 1977.
- [115] G. C. Stockman, S. Kopstein and S. Benett, "Matching images to models for registration and object detection via clustering," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 4, pp. 229-241, 1982.
- [116] A. Teifke, F. Schweden, H. Cagil, H. U. Kauczor, W. Mohr and M. Thelen, "Spiral-Computertomographie der Mamma," *Fortschr. Rontgenstr*, vol. 161, pp. 495-500, 1994.

- [117] Triangle Version 1.3, Developed by J. R. Shewchuk,
<http://www.cs.cmu.edu/~quake/triangle.html>.
- [118] P. A. van den Elsen, E-J. D. Pol and M. A. Viergever, "Medical image matching: a review with classification," *IEEE Engineering in Medicine and Biology*, vol. 12, pp. 26-39, 1993.
- [119] P. Van Wie and M. Stein, "A LANDSAT digital image rectification system," *IEEE Trans. Geosci. Electr. GE-15*, vol. 3, pp. 130-137, 1977.
- [120] J. C. Weinreb and G. Newstead, "Controversies in breast MRI", *Magn. Reson. Q*, vol. 10, pp. 67-83, 1994.
- [121] P. Wingo, T. Tong, S. Bolden, "Cancer statistics, 1995," *CA Cancer J. Clin.*, vol. 45, pp. 8-30, 1995.
- [122] G. Wolberg, *Digital image warping*, IEEE Computer Society Press Monograph, Los Alamos, California, 1990.
- [123] XV Version 3.10a by John Bradley, Bryn Mawr, Pennsylvania.
- [124] X. Zhou, R. Gordon, "Detection of early breast cancer: an overview and future prospects," *CRC Critic. Rev. Biomed. Eng.*, vol. 17, pp. 203-225, 1989.
- [125] X. Zhou, *Digital Subtraction Mammography via geometric unwarping for detection of early breast cancer*, Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 1991.
- [126] S. W. Zucker, "Region growing: childhood and adolescence," *Comput. Graphics Image Processing*, vol. 5, pp. 382-399, 1976.

Appendix

The appendix contains copies of the following three papers:

1. The first paper titled "Detection of breast cancer at a smaller size can reduce the likelihood of metastatic spread: a quantitative analysis" is reprinted from *Academic Radiology*, vol. 4, pp. 8-12, 1997.
2. The second paper titled "The rough landscape of the image registration problem" is a manuscript in preparation.
3. The third paper titled "Hu's moment invariants: How invariant are they under skew and perspective transformations?" is reprinted from *Proc. IEEE Wescanex Conference*, pp.292-295, June 1997.

The appendix also contains the C and Matlab source code used to perform image registration using the starbyte transformation.

Detection of Breast Cancer at a Smaller Size Can Reduce the Likelihood of Metastatic Spread: A Quantitative Analysis

Radhika Sivaramakrishna, MEng¹, Richard Gordon, PhD^{1,2,3}

Rationale and Objectives. The authors extrapolated the lognormal relationship between size of tumor and probability of metastasis to include small tumors.

Methods. Extrapolation was performed by using linear weighted regression analysis techniques to estimate prediction intervals for the predicted probabilities.

Results. Tumors detected at 1 cm in diameter had a 7.31% probability of metastasis (95% prediction interval [PI], 4.36% to 11.6%). Tumors detected at 5 mm in diameter had a 1.23% probability of metastasis (95% PI, 0.45% to 3.0%). Tumors detected at 2 mm had a 0.049% probability of metastasis (95% PI, 0.00705% to 0.267%).

Conclusion. This analysis shows a major reduction in metastasis probability when tumors are detected at small sizes. These results suggest that detection of very early tumors can substantially reduce the likelihood of metastatic spread.

Key Words: Breast cancer, prognostic indicators, tumor size, metastasis, probit analysis.

Currently, one in nine women in the United States and one in 10 women in Canada can be expected to develop breast cancer over a life span of 85 years. Breast cancer is the second leading cause of cancer mortality in American women and the most common cancer among American women [1] and among Canadian women [2]. The means to prevent breast cancer have not yet been found. Therefore, early detection and cure are very important. From an imaging point of view, if we are to screen asymptomatic women, tumor size is the most important prognostic indicator.

Although studies have indicated a relationship between tumor size and the probability of metastasis [3-7], to our knowledge no study has shown the implications of such a relationship for small tumors, which was the focus of this study. There has been a growing debate about whether it is really necessary to detect tumors at an early stage [8,9]. This study shows a

From the Departments of ¹Electrical and Computer Engineering, ²Radiology, and ³Physics, University of Manitoba, Winnipeg, Manitoba, Canada.

This work was supported by a studentship to R.S. from the Cancer Research Society in Montreal and by Nicholas Anthonisen, Dean of Medicine, University of Manitoba.

Address reprint requests to R. Gordon, PhD, Department of Radiology, University of Manitoba, Winnipeg, Manitoba, Canada R3A 1R9.

Received October 16, 1995, and accepted for publication after revision September 26, 1996.

Acad Radiol 1997;4:8-12

©1997, Association of University Radiologists

TABLE 1: Actual and Calculated Proportion of Metastases as a Function of Tumor Diameter and Volume at Time of Detection

Tumor Class	Tumor Diameter Range (cm)	Tumor Volume Range (mL)	No. of Patients	Proportion of Metastases (%)	
				From Actuarial Curves	From Lognormal Model
1	1-2.5	0.52-8.18	317	27.1	24.0
2	2.5-3.5	8.18-22.4	496	42.0	45.0
3	3.5-4.5	22.4-47.71	544	56.7	57.2
4	4.5-5.5	47.71-87.71	422	66.5	66.4
5	5.5-6.5	87.11-143.79	329	72.8	73.5
6	6.5-7.5	143.79-220.89	192	83.8	78.9
7	7.5-8.5	220.89-321.55	136	81.3	82.9
8	>8.5	>321.55	212	92.0	90.3

Source.—Koscielny et al [3].

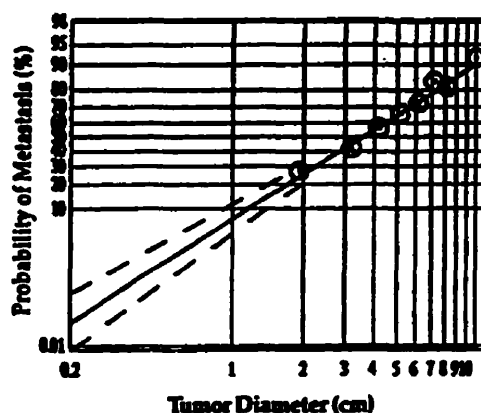


FIGURE 1. The regression line from the study by Koscielny et al [3] is depicted with the actual data points (actuarial values) shown as dots with circles around them. The graph illustrates that the tumor volume at detection is lognormally related to the probability of metastatic dissemination.

clear reduction in metastasis probability when tumors are detected at small sizes.

MATERIALS AND METHODS

Koscielny et al [3] showed a lognormal relationship between the probability of metastatic dissemination and the size of the primary tumor at detection. They used data from 2,648 patients with breast cancer treated at the Institut Gustave Roussy between 1954 and 1972. Eight classes of tumor size were defined according to clinical diameter of tumor at the time of detection. The patients were followed up for 25 years, and the number of patients in each class with metasta-

sis by the end of 25 years was cumulatively added to determine the cumulative proportion of metastasis for each size class. Assuming that after 25 years practically all distant metastases associated with primary tumor had become manifest, they concluded that the cumulative proportion of patients with metastasis at the end of this period was equal to the dissemination probability in that particular subgroup of patients. Table 1 gives the number of patients with each of the eight classes of tumor. The column with the heading "From Actuarial Curves" in Table 1 gives the cumulative proportion of patients with metastasis for each class. Koscielny et al [3] performed a weighted linear regression analysis between the probits of the proportion of metastases at long term and the mean value of the logarithms of the tumor volumes in each class. Although the numbers of patients were not directly taken into account, a weight equal to the reciprocal of the variance of the actuarial proportion of metastases was used. Their straight line (Fig 1, solid line) has the equation $Y = 0.3817X + 3.7939$, where the probit values are the Y values, and mean log [mL] volume values in each class are the X values. Their model fit the data very well.

The column "From Lognormal Model" in Table 1 indicates the values obtained by using the lognormal model. Because the model represents the data well, the corresponding values in the two columns for each tumor class are very close. Also, Koscielny et al [3] obtained a χ^2 value of 1.56. The number of degrees of freedom is 5 (8 data points minus 3); therefore, the idea of mean tumor volume being lognormally related to the probability of metastasis is reasonable.

We used Koscielny's data to illustrate how detection

of a smaller tumor volume would affect the dissemination probability. Because Koscielny's line extends down to tumors only 2 cm in diameter, we extrapolated the line into the tumor diameter range of 2 cm down to 0.2 cm. The volumes (V) corresponding to these tumor diameters (D) can be calculated with the formula for a sphere: $V = (\pi D^3)/6$ [3]. We then used the previous equation to calculate the probit that corresponded to the logarithm of the volumes obtained. From the probit, the actual percentages could be obtained by using the detailed tables of Fisher and Yates [10] or mathematic software like Mathematica (Wolfram Research, Champaign, IL) or Matlab (Mathworks Inc, Natick, MA).

The 95% prediction intervals for the predicted probabilities were calculated by using a standard formula from Myers [11] (their Equation 2.26, with the variance of the X 's replaced by a modified formula with weights and the residual sum of squares replaced by the weighted residual sum of squares). The 95% prediction intervals are shown as long dashes in Figure 1 along with the extrapolation. Because variance values for the data were not available, prediction intervals were calculated by using weights proportional to the number of patients.

RESULTS

Table 2 gives the predicted values for the extrapolation as well as the 95% prediction intervals for tumor diameters of 2, 1.8, 1.6, . . . 0.2 cm. From Table 2 it is clear that even for a 1 cm tumor, the 95% upper bound of the probability of metastasis is as low as 11.6%. As the diameter of the tumor at detection decreases, the probability of metastasis declines sharply; a 0.2 cm tumor has a 95% upper bound as low as 0.267%.

DISCUSSION

It could be argued that the linear relationship between the probit of the proportion of metastases in the long term and the logarithm of tumor volumes may not be relevant for smaller tumors. Because there are few patients in whom tumors are detected at such small sizes and even fewer who are followed up for a very long period, there may be no way of directly proving that the linear relationship is applicable to smaller tumors. Also, even if the linear relationship is not completely accurate, we would like to be reassured that

TABLE 2: Extrapolated Metastasis Probability

Diameter (cm)	Volume (mL)	Metastasis Probability (%)	95% Prediction Interval (%)
2	4.19	25.5	20.8–30.7
1.8	3.05	21.8	17.1–27.1
1.6	2.14	18.0	13.5–23.3
1.4	1.44	14.3	10.1–19.5
1.2	0.905	10.7	7.02–15.5
1	0.524	7.31	4.36–11.6
0.8	0.268	4.38	2.27–7.83
0.6	0.113	2.08	0.878–4.45
0.5	0.0654	1.23	0.45–3.0
0.4	0.0335	0.617	0.185–1.78
0.2	0.00419	0.0490	0.00705–0.267

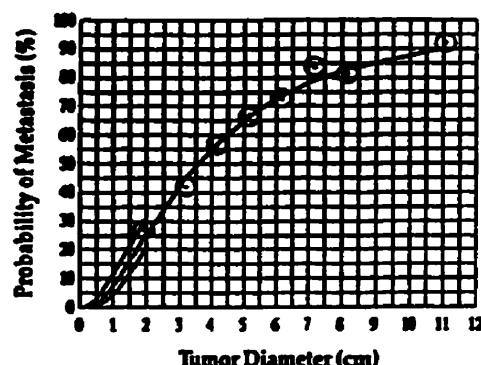


FIGURE 2. Figure 1 is shown replotted on a graph linear in probability on the y axis and linear in tumor diameter on the x axis. The extrapolated portion and the 95% prediction limits are all shown with short dashes.

this relationship represents a worst-case situation or an upper bound, so that probabilities would in fact be lower than the values predicted by the linear relationship.

There are two reasons favoring a linear relationship or a linear upper bound. First, when Figure 1 is replotted with linear probability on the y axis and linear tumor diameter (Fig 2) on the x axis, the smooth extrapolation through the origin is apparent. At zero tumor diameter or volume (ie, no tumor), it is obvious that there is no metastasis, so that the origin (0,0) must start the curve. Note that the curves for the prediction interval also must go through the origin. Second, there is no physical or biologic reason for the line to bend up; hence it is simplest to assume that the linear relationship does apply to smaller tumors or, in the worst case, that it represents an upper bound.

An analogous problem with extrapolation presents

itself in the well-studied radiation dose-response curve, for which models have been proposed to extrapolate high-dose curves to low-dose regions [12,13]. Concerns that animal models may not be applicable to humans have led to the use of the linear model with the understanding that it represents a worst-case situation [14].

Any extrapolation of data is speculative, because new phenomena may enter at unexplored ranges of the parameters (tumor diameter, in our case). Extrapolation is at the heart of prediction in science, so our calculations are predictions of how metastasis rates would be reduced if smaller tumors were detected regularly. Development of high-resolution breast imaging could help test our prediction, and we hope that our findings will motivate research in this direction.

Traditionally, breast-tumor volumes are calculated by using the volume for an oblate spheroid [15] because the predominant geometric shape of small primary breast cancers is spheroidal, with a long axis following the direction of the ducts [16,17]. However, for the data analyzed here, only one diameter was available, so a spherical shape was assumed [3]. Also, the clinical techniques used by Koscielny et al [3] to measure tumor diameter were approximate, and they differed for two different time periods in the study but were well correlated. A monotonic relationship between pathologic and clinical diameter was found (Fig 4 in the study by Koscielny et al [3]); for large tumors, however, there is the caveat that measurements were made after preoperative radiation therapy, and for small tumors the curve extrapolates to nonzero pathologic diameter for zero clinical diameter. The cause of the latter discrepancy is not clear. Thus we have confined our analysis to an extrapolation of the clinical diameters; these diameters were used in their Figure 2, which is the basis for our Figures 1 and 2. Also, because the diameters used to generate the lognormal graph (their Fig 2) were the mean log volumes of all tumors in a particular diameter range, exact tumor diameters were not required to generate the graph. And as we pointed out, the bound estimates we have calculated will also partly allow for possible errors in tumor-diameter measurements, as well as a spherical shape assumption for the tumors. The major source of error in tumor diameter was the grouping of tumors into classes according to size (Table 1). Although Koscielny's categorization of tumors by size is not as fine as we would have liked, their analysis of metastasis rate versus tumor size is the most quantitative of all those we found in the literature.

The following points are relevant to our analysis. First, although the extrapolation is to very small tumors, we have not indicated how these tumors might be detected. Mammography, which is currently used for screening, cannot routinely detect tumors as small as 0.2 cm, but other imaging modalities such as magnetic resonance imaging can detect such tumors in some cases [18]. Second, Koscielny and others [3,6,7] have demonstrated that the relationship between the size of the primary tumor at detection and the probability of metastasis is affected by other prognostic indicators such as the number of axillary nodes that have been invaded and the histologic grade. Third, in addition to a mass, there are other direct and indirect signs of breast cancer with different prognoses [19,20]. Tumor shape, by itself, appears to be an important prognostic indicator [21]. Fourth, tumor growth rate [22], pathologic type of carcinoma [23], patient age [15], mammographic visibility [24], and other factors are relevant to the analysis. These factors are interrelated because tumors with a high growth rate are associated with a poorer prognosis and occur more frequently in premenopausal women [25]; also, the density of the breast parenchyma, which influences the size at which a tumor is detected, is age dependent [26]. There are many such relationships amongst these factors that have not been taken into account in the extrapolation because they are not clearly quantifiable. Also, Koscielny's data are not differentiated with respect to tumor growth rate, pathologic type of carcinoma, and patient age.

Several researchers have expressed concern that an increase in the use of mammography for screening asymptomatic women will create an overdiagnosis bias because more small tumors will be detected that have a good prognosis, but no noticeable change in the mortality rate will be created [8,9]. For example, Miller [9] says "it re-emphasizes a truism that it is unnecessary to detect cancers as early as possible to obtain a benefit, it is only necessary to detect them early enough." We do not agree with this statement. The extrapolation shows that the probability of metastasis rises sharply in association with tumor diameter. For example, a tumor detected at 0.5 cm in diameter has a mean probability of metastasis of 1.23%, but a tumor detected at 2 cm has a mean probability of metastasis of 25.5%. These results show that if tumors could be detected when they were very small, then the probability of metastasis would be very low. As in prostate cancer [27], it is true that some

breast tumors will never result in a life-threatening disease. The problem would then be differentiating these tumors from those that would metastasize. This is a technical problem that we hope can be addressed with improved imaging techniques [28].

In conclusion, the size of a breast tumor is logarithmically related to the probability of metastasis. We have extrapolated this relationship to include small tumors and predict that detecting tumors at small sizes would greatly reduce the probability of metastasis. The extrapolation does not consider the influence of factors such as axillary node involvement, histologic grade, tumor shape, tumor growth rate, mammographic visibility, pathologic type of carcinoma, and patient age on the size-probability relationship. Nevertheless, this analysis clearly shows a major reduction in the metastasis probability when tumors are detected at small sizes.

ACKNOWLEDGMENTS

We thank the reviewers and Michael W. Vannier, MD, whose comments helped improve the quality of the article. We thank N. S. Shashidhar, PhD, for help with generating the graphs and Serge Koscielny, MD, for all his help.

REFERENCES

- Wingo PA, Tong T, Bolden S. Cancer statistics, 1995. *CA Cancer J Clin* 1995;45:8-30.
- Canadian Cancer Statistics 1992. Toronto, Canada: National Cancer Institute of Canada, 1992.
- Koscielny S, Tubiana M, Lee MG, et al. Breast cancer: relationship between the size of the primary tumor and the probability of metastatic dissemination. *Br J Cancer* 1984;49:709-715.
- Koscielny S, Tubiana M, Valleron AJ. A simulation model of the natural history of human breast cancer. *Br J Cancer* 1985;52:515-524.
- Tubiana M, Koscielny S. Natural history of human breast cancer: recent data and clinical implications. *Breast Cancer Res Treatment* 1991;18:125-140.
- Atkinson EN, Brown BW, Montague ED. Tumor volume, nodal status, and metastasis in breast cancer in women. *J Natl Cancer Inst* 1986;76:171-178.
- Carter CL, Allen C, Henson DE. Relation of tumor size, lymph node status, and survival in 24,740 breast cancer cases. *Cancer* 1989;63:181-187.
- Black WC, Welch GH. Advances in diagnostic imaging and overestimation of disease prevalence and the benefits of therapy. *N Engl J Med* 1993;328:1237-1243.
- Miller AB. Screening for cancer: is it time for a paradigm shift? *Ann RCPSC* 1994;27:353-355.
- Fisher RA, Yates F. Statistical tables for biological, agricultural and medical research. 6th ed. Edinburgh, Scotland: Oliver & Boyd, 1964.
- Myers RH. Classical and modern regression with applications. 2nd ed. Boston, MA: PWS-KENT, 1990.
- Feig SA. Radiation risk from mammography: is it clinically significant? *AJR* 1984;143:469-475.
- Webster EW. On the question of cancer induction by small x-ray doses. *AJR* 1981;137:647-666.
- Upton AC, Beebe GW, Brown JM, Quimby EH, Shellabarger C. Report of the NCI Ad Hoc Working Group on the risks associated with mammography in mass screening for the detection of breast cancer. *JNCI* 1977;58:481-493.
- Peer PGM, van Dijk JAAM, Hendriks JHCL, Holland R, Verbeek ALM. Age-dependent growth rate of primary breast cancer. *Cancer* 1993;71:3547-3551.
- Heuser L, Spratt JS, Polk HC. Growth rates of primary breast cancer. *Cancer* 1979;43:1888-1894.
- Heuser L, Spratt JS, Polk HC, Buchanan J. Relation between mammary cancer growth kinetics and the intervals between screenings. *Cancer* 1978;43:857-862.
- Orsi SG, Troupin RH. Nonmammographic imaging of the breast: current issues and future prospects. *Semin Roentgenol* 1993;28:231-241.
- D'Orsi CJ, Kopans DB. Mammographic feature analysis. *Semin Roentgenol* 1993;28:204-230.
- Martin JE, Moskowitz M, Milbrath JR. Breast cancer missed by mammography. *AJR* 1979;132:737-739.
- Giardina C, Ricco R, Lettini T, et al. Relation between primary tumor shape and biological behavior in breast cancer. *Tumori* 1989;75:117-122.
- Spratt JS, Spratt JA. Growth rates. In: Donegan WL, Spratt JS, eds. *Cancer of the breast*. Philadelphia, PA: W. B. Saunders, 1995;317-345.
- Gallager HS. Pathologic types of breast cancer: their prognoses. *Cancer* 1984;53:623-629.
- Holland R, Hendriks JHCL, Mravunac M. Mammographically occult breast cancer: a pathologic and radiologic study. *Cancer* 1983;52:1810-1819.
- Heuser LS, Spratt JS, Kuhns JG, Chang AFC, Polk HC, Buchanan JB. The association of pathologic and mammographic characteristics of primary human breast cancers with "slow" and "fast" growth rates and with axillary lymph node metastases. *Cancer* 1984;53:96-98.
- Wolfe JN. Breast parenchymal patterns and their changes with age. *Radiology* 1976;121:545-552.
- Slawin KM, Ohori M, Dilliglugli O, Scardino PT. Screening for prostate cancer: an analysis of the early experience. *CA Cancer J Clin* 1995;45:134-147.
- Zhou X, Gordon R. Detection of early breast cancer: an overview and future prospects. *CRC Crit Rev Biomed Eng* 1989;17:203-225.

THE ROUGH LANDSCAPE OF THE IMAGE REGISTRATION PROBLEM

**N. S. Shashidhar⁺, Radhika Sivaramakrishna^{*} and Richard Gordon^o
Departments of Electrical and Computer Engineering^{*o},
Biosystems Engineering⁺ and Radiology^o**

Author's address for correspondence

**Department of Radiology
University of Manitoba
ON-104, 820 Sherbrook Street
Winnipeg, Manitoba
Canada R3A 1R9
Tel: (204)-789-3828
Fax: (204)-787-2080
e-mail: radhika@cc.umanitoba.ca**

Abstract

Applications which require the registration of a pair of images frequently involve finding the global minimum of a rough multidimensional objective function. The aim of this paper is (1) to demonstrate the extent of this roughness in the landscape of the objective function for some image pairs and, thereby, attempt to relate the roughness in the objective function to the roughness in the original images, and (2) to demonstrate that this roughness can in some instances be smoothed by simple means without essentially changing the location of the global minimum.

Keywords

Image, registration, optimization.

Running Headline

The rough landscape of the image registration problem.

INTRODUCTION

The image registration problem consists of finding a transformation which maps a given reference image $G(x, y)$ to a given target image $F(x, y)$. The knowledge of the transformation is significant in a wide range of applications where features which are present in the reference image need to be identified in the target image or in applications requiring the detection of new features

which appear in the target image. The new features can be highlighted by showing a difference image formed by subtracting the correctly transformed reference image from the target image¹⁻³. In fact, presenting a difference image might be the only way of making very subtle changes in small features apparent to the eye. The application of this concept to the detection of breast cancer in its early stages is of special interest to the authors⁴⁻⁷.

In this paper, we focus our attention on image registration problems that involve solving the following functional equation to find the transformation $T(x, y)$ of the xy -plane into itself

$$G(x, y) = F(T(x, y)) \quad (1)$$

Obviously, the image registration problem need not be solvable for an arbitrarily chosen image pair or there may exist many transformations satisfying equation (1). Most of them would be highly discontinuous and obviously could not have arisen from a practical registration problem. However, if the transformation $T(x, y)$ is a smooth diffeomorphism of the xy -plane (for example, a one-to-one continuously differentiable mapping of the xy -plane into itself for which the inverse mapping is also continuously differentiable), then the visually identifiable features in G reappear in F , perhaps in displaced locations and with slight changes in appearance. In other words, the graphs of G and F considered as landscapes would have similar hills, valleys and plains. Conversely, if G and F are similar in this sense, it should be possible to find a smooth, locally non-degenerate transformation $T(x, y)$ such that

$F(T(x, y))$ is close to $G(x, y)$ in some sense⁸.

In a practical situation (arising, for example, in the problem of registering two computed tomographic mammograms of the same subject taken one year apart⁷) the two images are known to be related and, hence, the problem of finding a transformation becomes meaningful. Again, in a practical situation it becomes necessary to pose image registration as an optimization problem since we cannot hope to find a transformation which exactly satisfies equation (1). Accordingly, we seek a transformation of the xy -plane which *optimally* maps G to F , the transformation being chosen from an appropriately designed transformation space.

A numerical scheme to search for such an optimal transformation generally calls for an iterative procedure which starts with an initial trial transformation T_0 which is successively refined to generate a sequence of transformations $\{T_1, T_2, T_3 \dots\}$. Each term in the sequence is generated by modifying the transformation used in the previous step so that the transformed image approaches the given target image as monitored by the fall in value of an appropriately defined objective function.

The emphasis in this paper is not to solve any particular registration problem but to dwell on the fundamental issue of understanding the structure of the spaces and the behavior of the objective function associated with the registration procedure.

It is well known that for reasonably textured images, the optimization

problem associated with image registration involves a “rough” objective function – one which has numerous local minima. For this reason, even when the transformation between the reference and target image is a rigid transformation, the associated optimization problem frequently does not permit a simple solution and usually necessitates the development of iterative search techniques which can be computationally expensive^{9–11}. For more general deformations between the reference and target images, the numerical solution of the registration problem becomes even more involved^{7,12,13}. Hence, the optimality of the solution is often compromised in order to reduce computational complexity¹⁴.

The content of the paper is organized as follows: In the next section we briefly describe transformation spaces in general and also the specific transformation space we use in our numerical experiments. Next, we present a means of exploring the transformation space with a view to understanding the behavior of a registration algorithm traversing the space in search of a solution. In the following section we examine a simple means of reducing the roughness in the objective function associated with the registration procedure. This can lead to a simplification of the registration problem. A discussion of the results and concluding remarks follow in the last section.

THE TRANSFORMATION SPACE

In designing a transformation space within which the registration algorithm is to search for an optimal transformation, it is necessary to restrict the dimension of the transformation space to be finite so that each transformation of the xy -plane that can be applied to the reference image can be completely specified by writing out an ordered list of n numbers.

Further, the transformation space must be sufficiently constrained to exclude transformations which yield over-distorted images. At the same time, the transformation space must offer a sufficient degree of freedom to include transformations more general than elementary operations of scaling, translation and rotation.

To provide an example of a finite dimensional transformation space, we first consider mappings of the xy -plane (into itself) which are of the form

$$T(x, y) = (a_x + b_x x + c_x y + d_x xy, a_y + b_y x + c_y y + d_y xy). \quad (2)$$

with $a_x, b_x, \dots, d_y \in \mathbf{R}$.

Transformations of this form are called bilinear transformations¹⁵. It must be understood that the domain and range of T in equation (2) are appropriately restricted regions of the xy -plane so that T is one-to-one and onto. This ensures that well-defined programs can be written to transform (and, if necessary, untransform) the images during the registration procedure. We note that the transformation we have been calling T is frequently designated

in the literature as an inverse transformation (T^{-1}). The concepts introduced in this paper are more clearly appreciated using our convention and the variance with the usage in the literature should cause little confusion.

We can now define the transformation space \mathcal{T} as

$$\mathcal{T} = \left\{ T: \begin{array}{l} T \text{ is a mapping of the } xy\text{-plane} \\ \text{into itself and is of the form given in} \\ \text{equation (2).} \end{array} \right\} \quad (3)$$

\mathcal{T} , of course, contains an infinity of transformations that can be applied to a reference image. Each of these transformations is specified by supplying the values of the eight real numbers ($a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y$) in equation (2).

Figure 1 shows the effect of applying a transformation of the form in equation (2) to a square domain \mathcal{D} shown shaded in Figure 1(a). For the specific choice of the eight numbers we have used to define the transformation in this example, the domain \mathcal{D} is transformed to the shaded region \mathcal{R} shown in Figure 1(b). Notice that the range \mathcal{R} is not a rectangular region of the xy -plane. Straight lines in the domain \mathcal{D} parallel to one of the coordinate axes happen to be mapped by the transformation to straight lines in the range \mathcal{R} . However, the straightness of lines in other orientations (e.g. the diagonal of the domain \mathcal{D}) is not preserved under the transformation T .

We note that the n -tuple which specifies the transformation can be identified with a point in R^n . Hence, along the lines of many other optimization problems, image registration can be interpreted as a search of a region of R^n – the algorithm searches for a point in R^n which corresponds to that particular transformation that maps a given reference image G to an image which

is sufficiently close to the given target image F .

We now review an alternative (and visually appealing) way of specifying the polynomial forms in equation (2). Rather than specifying the eight-tuplet $(a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y)$, we specify four points in the domain \mathcal{D} and prescribe the four points in the range \mathcal{R} that these points should be mapped to by the transformation. The information provided by these “matching points” is equivalent to specifying the polynomial coefficients in equation (2).

Converting information given as a set of matching points to the form of equation (2) involves solving a set of linear equations: Let the four points in the domain \mathcal{D} be (x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (x_4, y_4) and their images under the transformation be (X_1, Y_1) , (X_2, Y_2) , (X_3, Y_3) and (X_4, Y_4) (a specific choice of these points has been shown in Figure 1). The coefficients of the polynomial forms in equation (2) are easily obtained by solving the following eight equations which are linear in $(a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y)$:

$$a_x + b_x x_1 + c_x y_1 + d_x x_1 y_1 = X_1$$

$$a_x + b_x x_2 + c_x y_2 + d_x x_2 y_2 = X_2$$

$$a_x + b_x x_3 + c_x y_3 + d_x x_3 y_3 = X_3$$

$$a_x + b_x x_4 + c_x y_4 + d_x x_4 y_4 = X_4$$

$$a_y + b_y x_1 + c_y y_1 + d_y x_1 y_1 = Y_1$$

$$a_y + b_y x_2 + c_y y_2 + d_y x_2 y_2 = Y_2$$

$$a_y + b_y x_3 + c_y y_3 + d_y x_3 y_3 = Y_3$$

$$a_y + b_y x_4 + c_y y_4 + d_y x_4 y_4 = Y_4 \quad (4)$$

When working with a practical registration problem, it is possible to estimate an approximate set of matching points ("by eye") by looking for specific features in the reference image which reappear in the target image. Hence, the visual appeal of the method of specifying the transformation by specifying matching points.

We clarify these concepts with specific examples. The image pairs we have chosen are shown in Figure 2. The textured reference image of Figure 2(a) was generated by drawing short, white line segments of random length and orientation on a black background. Figure 2(c) is a reference image derived by cropping a mammogram and Figure 2(e) is a familiar test image. The target images (Figures 2(b), (d), and (f)) were generated from the corresponding reference images using the bilinear transformation defined by

$$T_{GF}(x, y) = \begin{pmatrix} -9 & + & 0.95 & x & + & 0.125 & y & + & 0.00125 & xy, \\ 0 & + & 0.1 & x & + & 1.05 & y & - & 0.00125 & xy. \end{pmatrix} \quad (5)$$

By having derived the target images from the reference images in this particular way, we have guaranteed that the solution to the registration problem can indeed be found in the transformation space which in this case is the space of all one-to-one and onto mappings of the form given in equation (2).

The transformation $T_{GF}(x, y)$ given by equation (5) which deforms the reference images shown in Figures 2(a), (c) and (e) to the target images shown in Figures 2(b), (d) and (f) can be identified with the point

$(-9, 0.95, 0.125, 0.00125, 0, 0.1, 1.05, -0.00125)$ in R^8 . Alternatively, as explained previously in this section, this transformation can also be specified by prescribing matching points. We have chosen the following four points in the domain \mathcal{D} as the (x_i, y_i) appearing in equation (4)

$$(40, 40), (40, 80), (80, 40), (80, 80).$$

If we require these four points to go to the following set of points (the (X_i, Y_i) appearing in equation (4)) in the range \mathcal{R}

$$(36, 44), (43, 84), (76, 46), (85, 84),$$

we have effectively specified that transformation of the xy -plane given by equation (5) which exactly maps the given reference images to the given target images.

EXPLORING THE TRANSFORMATION SPACE

In order to explore the multidimensional transformation space (and, thereby, appreciate the working of potential registration algorithms) we need to limit our viewpoint to visualizable two-dimensional subspaces of the transform space. We also need to construct an objective function which rates transformations in a reasonable way.

Further, we need to ensure that the subspace we select will include both the identity transformation (which leaves the reference image unchanged)

and the transformation which exactly maps the reference to the target image. By doing this we will be able to present precisely what might happen as a registration algorithm attempts to iteratively transform the reference image to the target image.

We will now define a specific two-dimensional subspace of the eight-dimensional transformation space defined in equation (3) which we will use in our exploration. This subspace will contain the identity transformation, the transformation which exactly carries the points (x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (x_4, y_4) to the points (X_1, Y_1) , (X_2, Y_2) , (X_3, Y_3) and (X_4, Y_4) and additionally, an infinity of other transformations each of which will be specified by an ordered pair of real numbers (α, β) .

Once a value of (α, β) is fixed, the particular member of the subspace is identified as that transformation $T(x, y)$ which carries the points (x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (x_4, y_4) to their "new locations" given by the ordered pairs

$$\begin{aligned} T(x_1, y_1) &= ((1 - \alpha)x_1 + \alpha X_1, (1 - \beta)y_1 + \beta Y_1) \\ T(x_2, y_2) &= ((1 - \alpha)x_2 + \alpha X_2, (1 - \beta)y_2 + \beta Y_2) \\ T(x_3, y_3) &= ((1 - \alpha)x_3 + \alpha X_3, (1 - \beta)y_3 + \beta Y_3) \\ T(x_4, y_4) &= ((1 - \alpha)x_4 + \alpha X_4, (1 - \beta)y_4 + \beta Y_4) \end{aligned} \tag{6}$$

Having defined the subspace in this way, we can set $(\alpha, \beta) = (0, 0)$ to select the identity transformation and (for the choice of matching points given in the section which introduces the transformation space) we can set

$(\alpha, \beta) = (1, 1)$ to select the transformation which exactly maps the reference images in Figures 2(a), (c) and (e) to their targets. An iterative registration algorithm which successfully solves the registration problem in this subspace can be thought of as tracing a trajectory in the $\alpha\beta$ - plane starting from the origin and terminating at the point $(1,1)$.

We next introduce the concept of an objective function defined over this two-dimensional subspace by noting that for every trial transformation considered in the solution process, the registration algorithm needs some input on (1) how close to a solution it is and (2) how to steer the trajectory in the $\alpha\beta$ -plane so that the solution is approached. Both of these criteria are automatically met if we define an objective function whose minimization amounts to solving the registration problem. An obvious candidate for the objective function is the root-mean-squared error between the target image and the transformed reference image. This is defined as the square root of the sum-of-squares of the difference between the two images.

Thus, the objective function E which we will use in our numerical experiments is a mapping of the $\alpha\beta$ -plane into the real line. E attains the global minimum value of 0 when $(\alpha, \beta) = (1, 1)$.

An important means of gaining insight into the registration process is to visualize the roughness in the topography of the two-dimensional function $E(\alpha, \beta)$. The topography of E can be represented in several ways: E can be depicted as another image, a mesh-plot, a contour map etc. After some initial

experimentation, we realized that the spread in values of E can be so large that, in any of these representations, the fine structure of E is completely obscured.

Therefore, we choose to depict the roughness in the objective function by drawing a vector field plot of the gradient of $-E$. The result of a numerical experiment (starting with the image pair of Figures 2(a) and (b) with the subspace as defined in equation (6)) is shown in Figure 3(a). Here, we have shown a region of interest of the $\alpha\beta$ -plane (α and β both in the interval $[0, 2]$) and arrows showing the vector field $(-\frac{\partial E}{\partial \alpha}, -\frac{\partial E}{\partial \beta})$ computed for a grid of values of (α, β) . Figure 3(b) shows a detail of the vector field over a smaller region (α and β both in the interval $[0, 0.4]$). In these and other vector field plots, vectors falling short of a certain threshold in length have been regarded as zero vectors and have been shown as dots.

Figures 3(c), (d), (e) and (f) show the vector field plots associated with the other two image pairs (Figures 2(c), (d), (e) and (f)).

We provide an interpretation of these results in the last section.

SMOOTHING THE OBJECTIVE FUNCTION

In this section we use the method of visualizing the transform subspace described above to examine the effect of systematically removing detail from the original image pairs by filtering. Figure 4 shows three filtered versions of

the image pair of Figures 2(a) and (b) generated using first-order low-pass Butterworth filters of successively smaller pass-band width w .

The transfer function of the Butterworth filter¹⁶ used was

$$H(u, v) = \frac{1}{1 + \left(\frac{u^2 + v^2}{w^2}\right)}$$

where u and v are the coordinates in the frequency domain and w is the pass-band width. The values of w used were 6000 (Figure 4(a) and (b)), 2000 (Figures 4(c) and (d)), and 100 (Figures 4(e) and (f)).

The vector field plots corresponding to these images are shown in Figure 5. The vector fields associated with identically filtered versions of the other two image pairs (Figures 2(c), (d), (e), and (f)) are shown in Figure 6 and Figure 7. (The filtered versions of these image pairs themselves have not been shown in this paper).

DISCUSSION AND CONCLUSIONS

We begin the discussion with an interpretation of the vector field plots. An "easy" registration problem can be thought of as one for which the vector field plot shows well-defined arrows all converging to (or near) the point (1, 1) in the $\alpha\beta$ -plane. To obtain a solution, it is also important that there exist possible trajectories in the $\alpha\beta$ -plane which do not pass through regions containing zero vectors or vectors pointing in "wrong" directions (i.e. away from the point (1, 1)). The vector field plot permits a convenient visualization of possible trajectories which correspond to chains of head-to-tail linked arrows.

A gradient based optimization algorithm (arguably the simplest of optimization algorithms¹⁷) cannot be used to locate the minimum of an objective function which has a vector field as non-converging as the one shown in Figures 3(a) and (b). Such an algorithm would fail because of the presence of local minima near the origin (as revealed by the zero vectors in Figure 3(b)) and also because there exist regions containing vectors pointing in "wrong" directions which the algorithm has to traverse.

An examination of the vector field plots corresponding to the other two image pairs reveals that a registration algorithm (if based on a gradient-descent method of optimization) will do better at registering the image pair of Figures 2(c) and (d) and will find the task of registering the image pair of Figures 2(e) and (f) to be the "easiest" of the three.

Our usage of the term "easy" must be interpreted in relation to the run-time speed of the optimization algorithm and also in relation to the complexity of the descent method of optimization used. This variety of complexity generally takes the form of refinements which must be made in the choice of descent direction and refinements in line searches.

Having examined the vector field plots corresponding to the three entirely unrelated image pairs of Figure 2, we already begin to appreciate the potential of the vector field plots, both for visualizing transformation spaces and also for evaluating the registerability of image pairs. We have, of course, confined our exploration of the transformation space over one specific two-

dimensional subspace. For the images considered in this paper, no particular differences were noted in vector field plots corresponding to other possible two-dimensional subspaces.

We next draw attention to the fact that a systematic reduction in detail contained in the image pair can lead to a simplification of the registration problem. This is revealed in the vector field plots of Figure 5. It is important to note that the corresponding image pairs (Figure 4) exhibit features which can be matched “by eye” even after considerable blurring has been introduced by the filters.

The connection between the amount of filtering introduced and the convergence of the arrows in the vector field plot seems to break down if the blurring introduced by the filter is so excessive that the features in the image pair can no longer be visually matched. This fact is revealed in the vector field plots of Figure 6. In fact, the original image pairs used to generate the vector field plots of Figures 6(c), (d), (e) and (f) end up being so overfiltered that the image pairs appear to the eye as blank images.

The third image pair (Figures 2(e) and (f)) happens to contain enough detail spanning a broader range of spatial scales. Consequently, even after passing through a Butterworth filter with $w = 100$ this image pair continues to be visually registerable. We also note that this image pair contains little texture compared to the other two image pairs. These facts are consistent with our finding that the vector field plots for this image pair remain well-

converging for all filter widths we have considered here.

The Butterworth filter which we have used to reduce the amount of detail in the image pairs is definitely not the best choice of filter for all image types. Alternative filters which locally act on individual features are more appropriate for image pairs like that in Figures 2(c) and (d). In fact, it might be profitable to absorb the locally acting filter into the definition of the objective function itself so that the registration algorithm is selectively encouraged to match certain types of features (these might be radiologically significant, for example).

Other researchers¹³ have used a hierarchical structure for image registration, where large features are matched at the coarsest resolution. The solution at the coarsest level is interpolated and used as the first approximation to the next finer level. But no systematic study of the effect of the relation between the reduction in detail in the images to the ease with which registration can be performed has been presented in the literature.

It seems intuitively obvious that reducing the amount of detail in the image pairs should make the objective function less rough. However, if we are interested in registering the two original images we need to be certain that the same transformation which minimizes the smoothed objective function also minimizes the original objective function. In other words, we need to be certain that the solution to the derived "smooth registration problem" is also an acceptable solution to the original "rough registration problem". We

find this to be the case for the examples considered here.

In this paper, we have considered very simple transformations and taken sufficient care to ensure that the images are registerable. Image pairs originating from a practical registration problem need not have this pleasant property. We need to experiment with a sequence of progressively less registerable image pairs to learn more about the limits to which a simple transformation space can capture deformations outside the space. Even the presence of noise in an otherwise registerable image pair can cause complications in the registration process which are difficult to understand in the absence of a systematic study. Also, the image pairs may be related not only by a spatial transformation but also by an intensity transformation¹⁸.

This paper illustrates that it is possible to develop registration algorithms based on simpler optimization techniques (by simple pre-processing of the original images), in place of over-generalized and time consuming optimization methods which are frequently used to register images. This paper is to be regarded as the first in a sequence of studies directed towards a better understanding of the registration process.

Acknowledgements

The authors are grateful to Dr. Martin H. Reed for providing insightful discussions. Fellowships to the second author from the Cancer Research Society, Montreal and the University of Manitoba, Winnipeg are gratefully acknowledged.

REFERENCES

1. Rosenfeld, A and Kak, A C *Digital Picture Processing*, Vol. I and II, Academic Press, Orlando, Florida (1982)
2. Brown, L G 'A survey of image registration techniques', *ACM Computing Surveys*, Vol 24 No 4 (1992) pp 325-376
3. van den Elsen, P A, Pol, E-J D and Viergever, M A 'Medical image matching: a review with classification', *IEEE Engineering in Medicine and Biology*, Vol 12 No 1 (1993) pp 26-39
4. Gordon, R 'Toward robotic "x-ray vision": new directions for computed tomography', *Applied Optics*, Vol 24 No 23 (1985) pp 4124-4133
5. Zhou, X and Gordon, R 'Detection of early breast cancer: an overview and future prospects', *CRC Crit. Rev. Biomed. Eng.*, Vol 17 No 3 (1989) pp 203-255
6. Sivaramakrishna, R and Gordon, R 'Detection of breast cancer at a smaller size can reduce the likelihood of metastatic spread: a quantitative analysis', *Acad. Radiol.*, submitted
7. Mazur, A K, Mazur, E J and Gordon R 'Digital differential radiography (DDR): a new diagnostic procedure for locating neoplasms, such as breast cancers, in soft, deformable tissues', *SPIE*, Vol 1905 (1993) pp 443-455

8. Amit Y 'A non-linear variational problem for image matching', *SIAM J. Sci. Computing*, Vol 15 No 1 (1994) pp 207-24
9. Unser, M, Thevenaz, P, Lee, C and Ruttimann, U E 'Registration and statistical analysis of PET images using the wavelet transform', *IEEE Engineering in Medicine and Biology*, Sept-Oct 1995 pp 603-611
10. Ranade, S and Rosenfeld, A 'Point pattern matching by relaxation', *Pattern Recognition*, Vol 12 (1980) pp 269-275
11. Stockman, G C, Kopstein, S and Benett S 'Matching images to models for registration and object detection via clustering', *IEEE Trans. Patt. Anal. machine Intell.*, Vol 12 No 2 (1982) pp 131-147
12. Broit, C 'Optimal registrations of deformed images', Ph. D. Dissertation, University of Pennsylvania, 1981.
13. Bajscy, R and Kovacic, S 'Multiresolution elastic matching," *Comput. Vision Graph. Image Process.*, Vol 46 (1989) pp 1-21
14. Goshtasby, A and Stockman, G C 'Point pattern matching using convex hull edges', *IEEE Trans. Syst. Man Cybernetics*, Vol SMC-15 (1985) pp 631-637
15. Wolberg, G 'Digital image warping', *IEEE Computer Society Press Monograph*, Los Alamos, California (1990)

16. Gonzalez, R C and Wintz P 'Digital Image Processing', Addison-Wesley Publishing Co, Reading, MA (1987)
17. Spang III, H A 'A review of minimization techniques for nonlinear functions', *SIAM Review*, vol 4 No 4 (1962) pp 343-365
18. Wong, R Y 'Sensor transformations', *IEEE Trans. Syst. Man Cybernetics*, Vol SMC-7 No 12 (1977) pp 836-840

FIGURE CAPTIONS

Figure 1: Showing the action of the bilinear transformation given by the equation $T(x, y) = (-9 + 0.95x + 0.125y + 0.00125xy, 0 + 0.1x + 1.05y - 0.00125xy)$. The square domain \mathcal{D} and the regular grid in (a) are mapped into the nonsquare range \mathcal{R} and the irregular grid in (b). The transformation can be completely specified by identifying the four points $(x_1, y_1), \dots, (x_4, y_4)$ in (a) and their matching points $(X_1, X_1), \dots, (X_4, X_4)$ in (b).

Figure 2: The image pairs participating in the numerical experiments described in this paper. The textured reference image (a) was generated by drawing short, white line segments of random length and orientation on a black background. (c) is a reference image derived by cropping a mammogram. (e) is a familiar test image. The corresponding target images (b), (d), and (f) were generated from the corresponding reference images using the bilinear transformation given by $T(x, y) = (-9 + 0.95x + 0.125y + 0.00125xy, 0 + 0.1x + 1.05y - 0.00125xy)$.

Figure 3: Vector field plots of the negative gradient of the root-mean-square error between the target and the transformed images of Figure 2. The vector field plots for each image pair have been shown at two levels of resolution to avoid cluttering. Points in the $\alpha\beta$ -plane correspond to bilinear transformations selected from a two-dimensional subspace described in the section on exploring the transformation space. Each arrow in the vector field plots indicates how the corresponding transformation must be modified to reduce the error in the match between the target and the transformed images.

Figure 4: Filtered versions of the textured image pairs of Figures 2(a) and (b). These were generated by passing the image pair through first order, low-pass Butterworth filters (see the section "Smoothing the objective function"). The pass-band widths used were 6000 ((a) and (b)), 2000 ((c) and (d)), and 100 ((e) and (f)).

Figure 5: Vector field plots of the negative gradient of the root-mean-square error between the target and the transformed images of Figure 4.

Figure 6: Vector field plots computed along the lines of those in Figure 5 except that the image pair in Figures 2(c) and (d) was used in place of Figures 2(a) and (b). All other settings remain the same.

Figure 7: Vector field plots computed along the lines of those in Figure 5 except that the image pair in Figures 2(e) and (f) was used in place of Figures 2(a) and (b). All other settings remain the same.

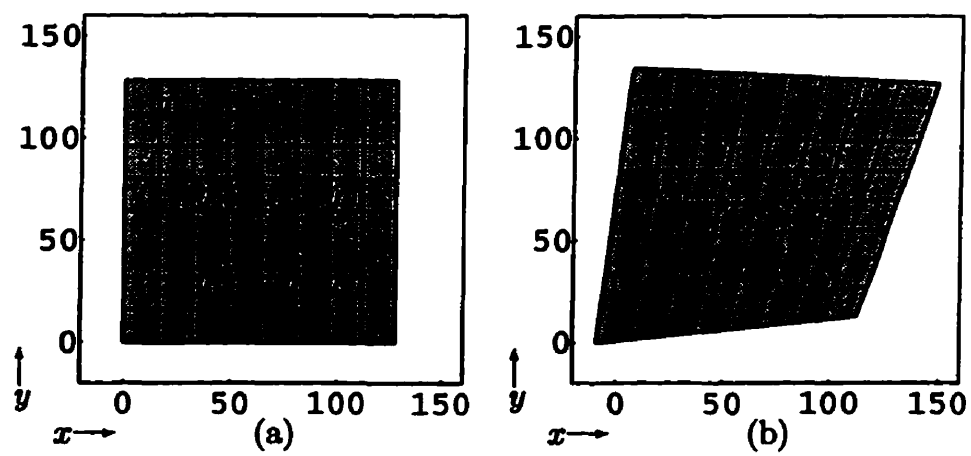
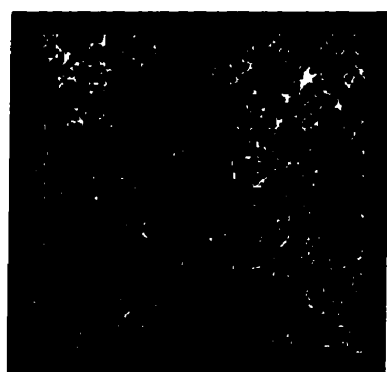


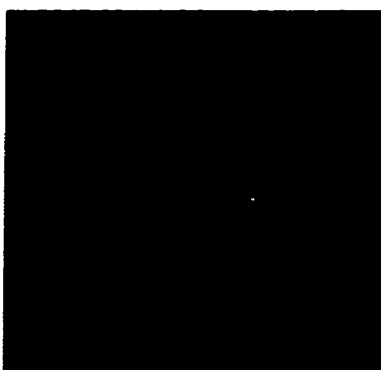
Figure 1



(a)



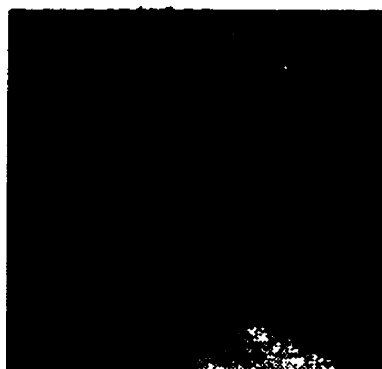
(b)



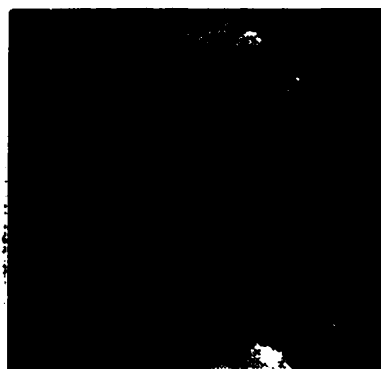
(c)



(d)



(e)



(f)

Figure 2

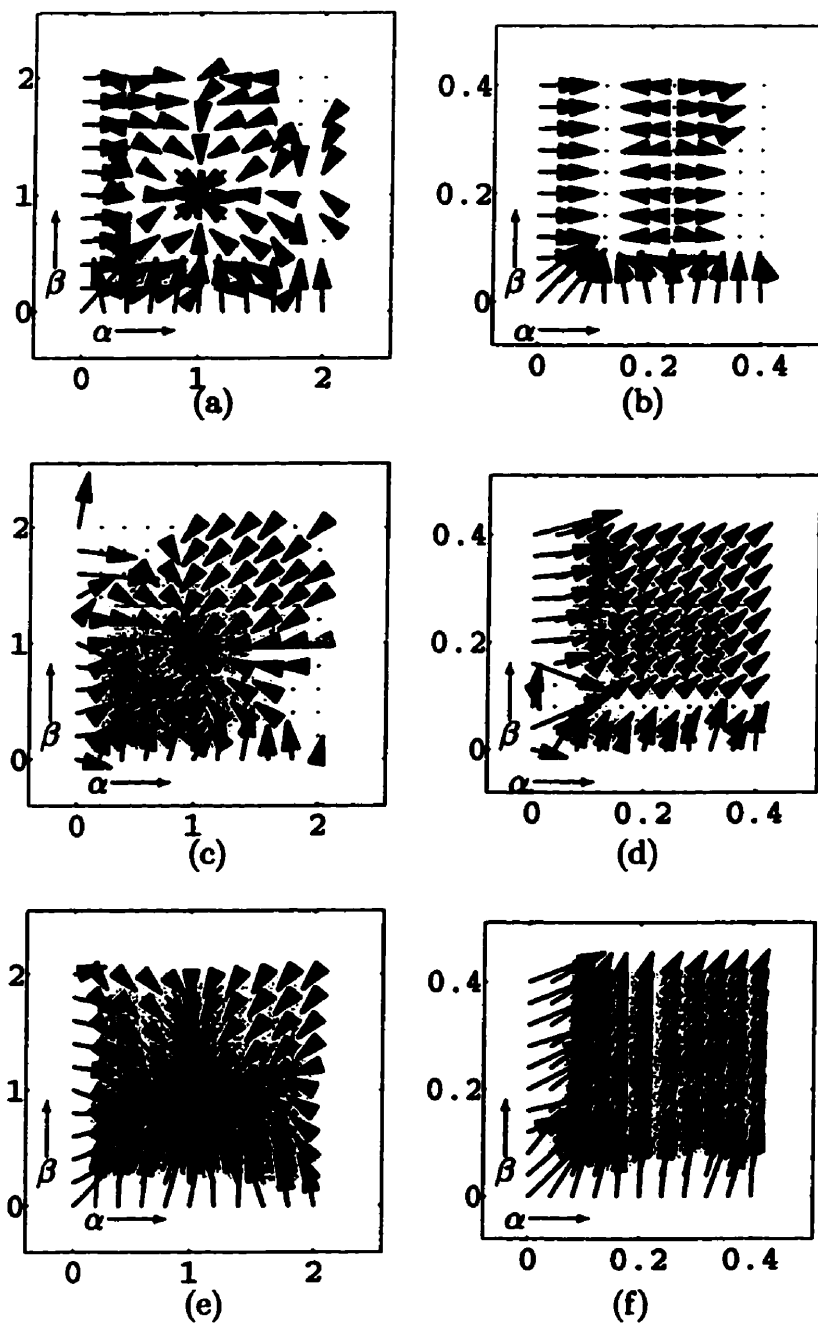


Figure 3



(a)



(b)



(c)



(d)



(e)



(f)

Figure 4

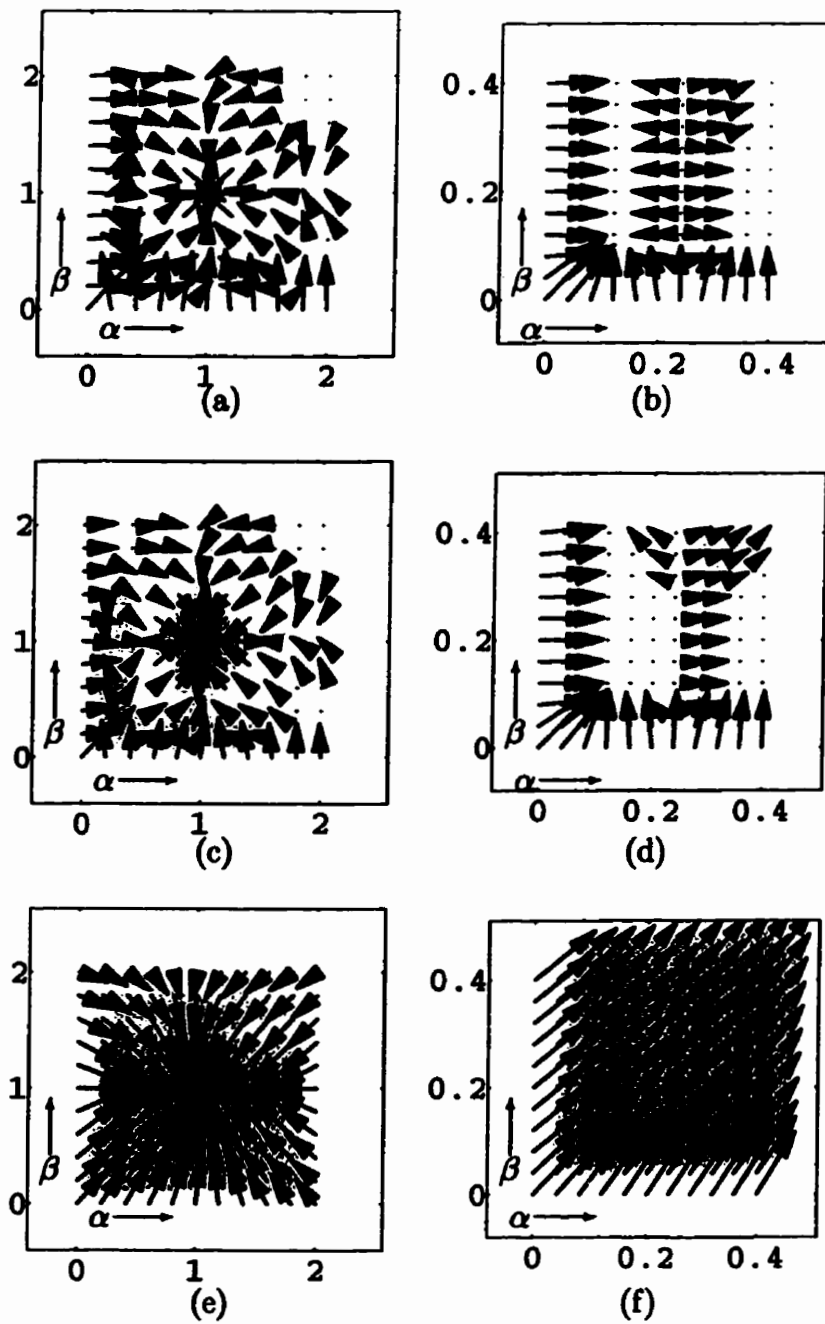


Figure 5

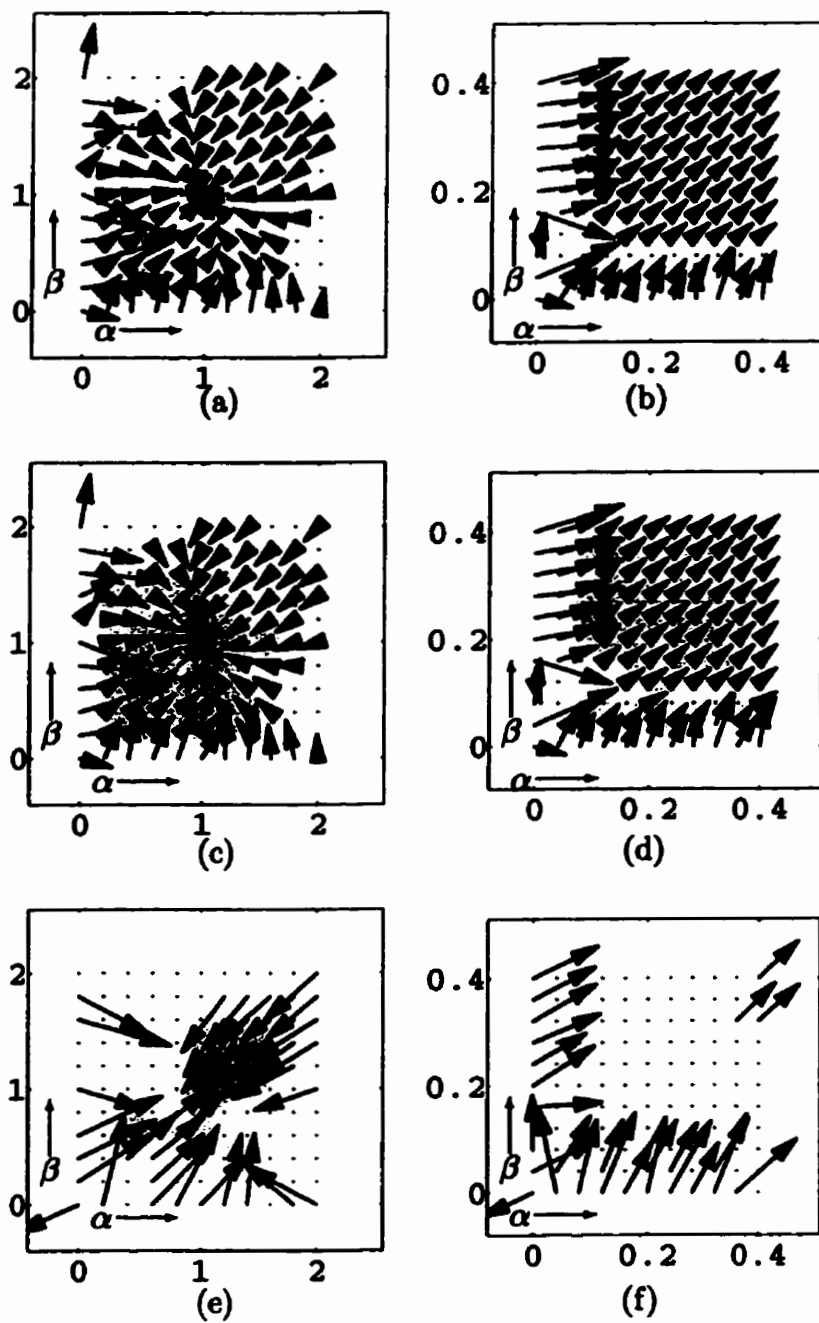


Figure 6

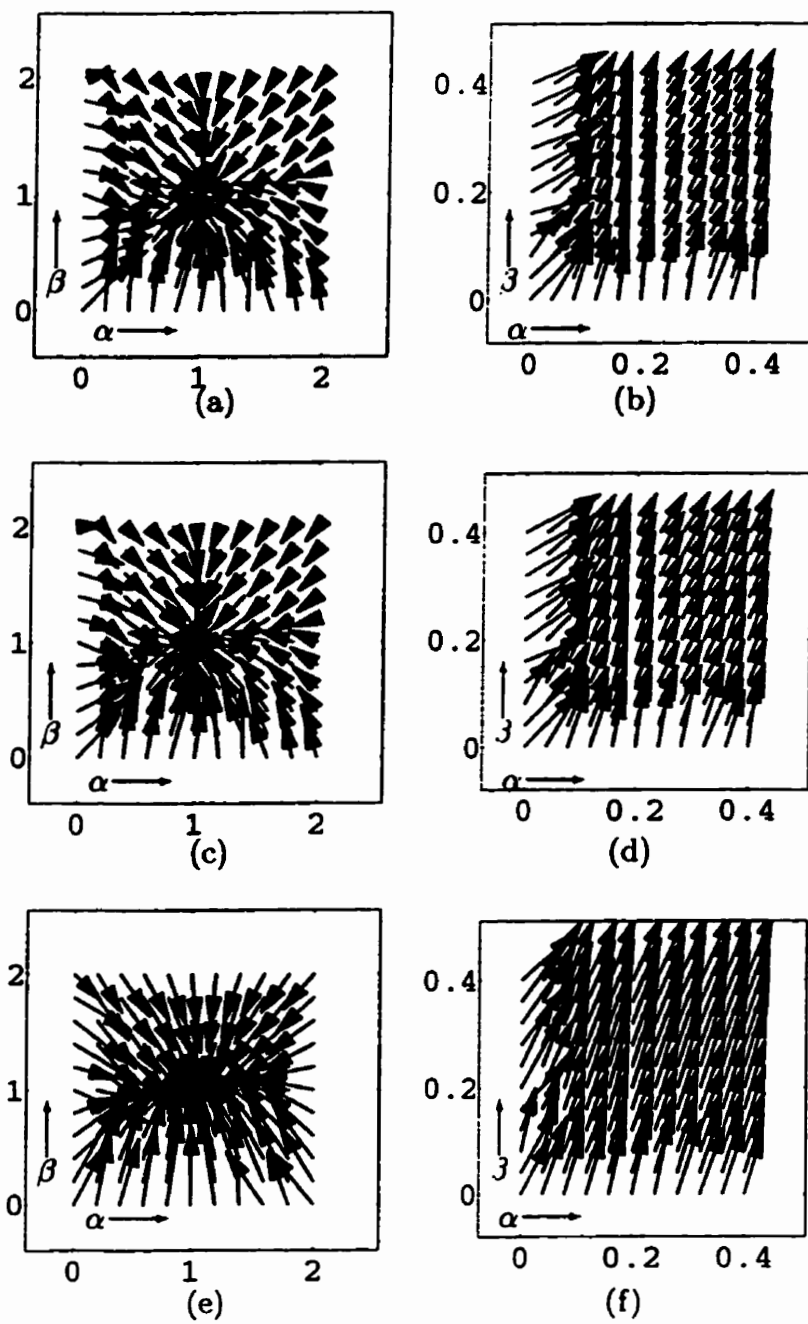


Figure 7

Hu's moment invariants: How invariant are they under skew and perspective transformations?

Radhika Sivaramakrishna*, *Student Member, IEEE* and N. S. Shashidhar†
 Department of Electrical & Computer Engineering* and Biosystems Engineering†
 University of Manitoba, Winnipeg, Manitoba, Canada.

Abstract— Hu's moment invariants are used in pattern recognition to provide a scale, orientation and position invariant characterization of the shape of a given object. In practical situations, objects may undergo other types of deformations, most commonly skew and perspective distortions. This paper explores the behaviour of Hu's invariants under these distortions.

Keywords— Hu's moment invariants, pattern recognition, shape matching, continuity.

I. INTRODUCTION

A typical pattern recognition task involves matching the shape of a given object in an image with one of a collection of standard objects in a library. This is accomplished by seeking a match between a suitable numerical characterization of the shape of the given object and the numerical characterizations of the objects in the library. The given object is then deemed to be closest to that object in the library whose numerical characterization is closest to its own. Ideally, two important properties of a shape characterization are (1) visually distinct objects should have distinct characterizations and (2) numerical similarity of the characterization of two objects should correspond to a visual similarity between them.

A number of moment-based techniques have been developed[3] for characterizing the shape of objects within an image. These techniques are useful in applications for classifying objects or for symbolically describing objects. For a given object in an image, Hu[1] defines a characterization consisting of an ordered seven-tuplet of real numbers listing seven moment invariants de-

rived from the first three central moments. This characterization is invariant to object scale, position and rotation. For a given object, finding a match in the library using Hu's characterization would be an easy task if the only transformation between the given object and the correct match in the library is a rigid transformation or scale change. However, in a practical situation, a given object may have undergone other types of deformations. The most common non-rigid type of distortions are skew and perspective transformations. The behavior of Hu's invariants has never been studied quantitatively for skew and perspective transformations. This is the focus of this paper in which we explore the limits of applicability of Hu's characterization under these more general transformations.

II. THE TEST OBJECTS

Figure 1 shows the images of the club, heart and diamond playing-card symbols (each about 50×50 pixels in size) and their skew- and perspective-transformed versions which were used as test objects.

Each object in Figure 1 was generated by rotating the corresponding base object (i.e. the club, heart or diamond symbol) in three-dimensions and projecting on to the xy -plane with respect to a viewpoint. The amount of rotation was specified by the Euler angles (ψ, θ, ϕ) given by

$$(\psi, \theta, \phi) = \frac{(i-1)}{11}(10, 10, -10),$$

where i is the column index which runs from 1 to 12. The viewpoint was held fixed at

∞ (10, 10, 10) to generate the skew-transformed sequences (i.e. the view direction was specified by the direction ratios (10, 10, 10)) and at (10, 10, 10) to generate the perspective-transformed sequences.

III. HU'S CHARACTERIZATION OF THE OBJECTS

Each object in Figure 1 has a characterization which is an ordered seven-tuplet of real numbers listing the seven Hu's moment invariants. The formulas to compute these numbers can be found in any of the references cited in this paper.

It must be noted that it is only for continuous images that Hu's characterization is strictly invariant to object scale, position and rotation. For real images, transformations such as these are, in general, approximate since every pixel location has to be rounded off in order to construct the transformed image. Hence, in order to make this study of Hu's characterization meaningful, we have taken care to retain all pixel positions exactly after applying the transformations. In other words, we have suppressed non-invariance arising due to discretization and, thereby, allowed the characterization to be affected by the transformations alone. The plots in Figure 2 show the seven Hu's moment invariants of all the objects in Figure 1.

We used an Euclidean distance function defined on the space of Hu's characterization to assess the extent of deformation which would yield the correct match for each object. Under the skew transformations considered, Hu's characterization pointed to the correct match in all cases. For example, the Hu's characterization of any of the skew-transformed club symbols was found to be closer to the Hu's characterization of the standard club symbol rather than to that of the heart or diamond symbols. Under the perspective transformations considered, Hu's characterization was able to pick up the correct match for the first 7 images for the club sequence, the first 11 images for the heart sequence and all the images for the diamond sequence.

IV. CONCLUSIONS

This brief study of the behavior of Hu's moment invariants under skew and perspective transformations show that (1) Hu's characterizations can be used for matching objects which have been transformed using skew and perspective transformations and that matches obtained using Hu's characterizations are almost as good as visually obtained matches (2) Continuous changes in the space of objects corresponds to continuous changes in the space of Hu's characterization.

References

- [1] M. K. Hu, "Visual pattern recognition by moment invariants", *IRE Trans. Inform. Theory*, IT-8, 1962, pp. 179-187.
- [2] W. K. Pratt, *Digital image processing*, John Wiley and sons, inc.
- [3] R. J. Prokop and A. P. Reeves, "A survey of moment-based techniques for unoccluded object representation and recognition", *CVGIP: Graphical Models and Image Processing*, Vol. 54, No. 5, September 1992, pp. 438-460.

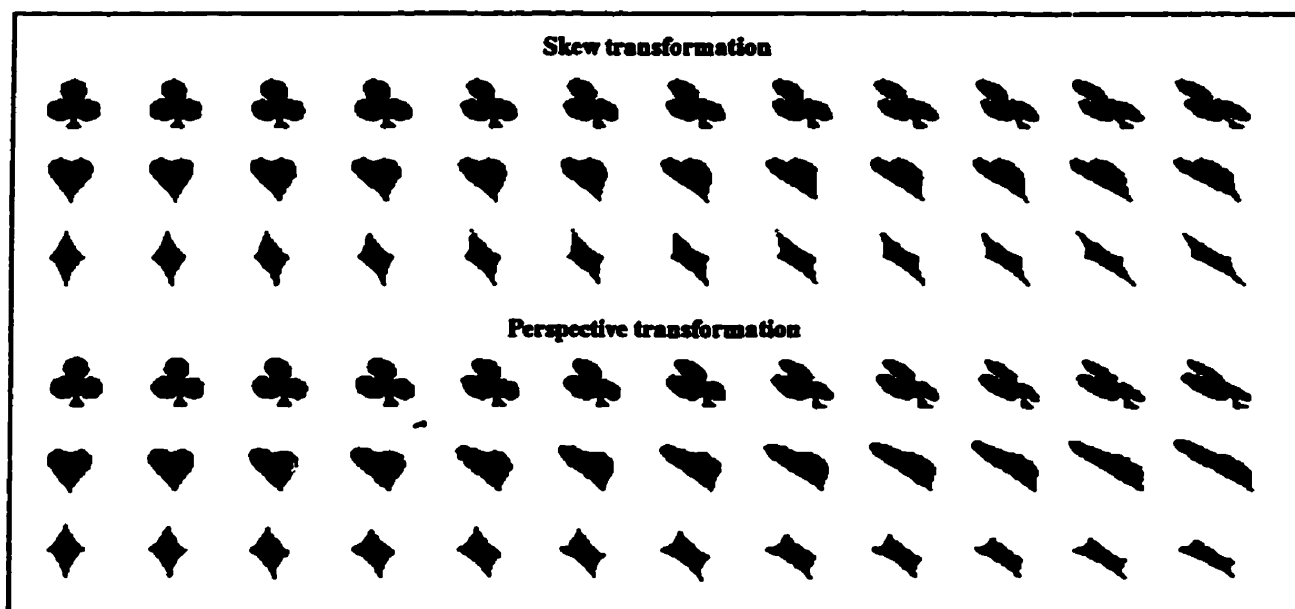


Figure 1: The array of test objects derived from the club, heart and diamond playing-card symbols. Reading from left to right, every row contains a sequence of gradually changing objects beginning with an undistorted object appearing in the first column. The objects in the first three rows have been subjected to a skew transformation and those in the last three rows, to a perspective transformation.

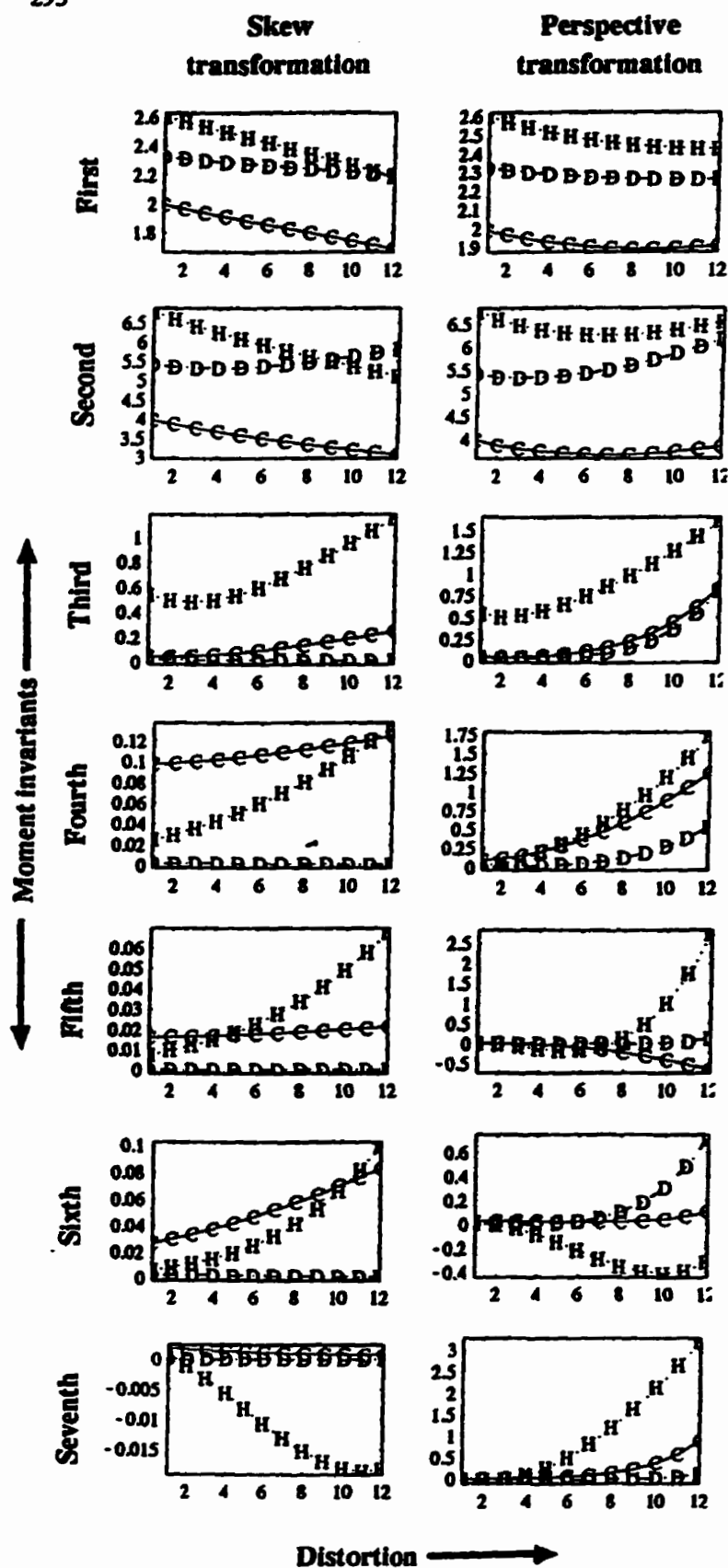


Figure 2: The set of seven Hu's moment invariants of the objects shown in Figure 1. Each sub-plot contains three separate labelled traces corresponding to the club (C), heart (H) and diamond (D) playing card symbols. The index used for the horizontal axis in each plot refers to the column number of the object in Figure 1.

DESCRIPTION OF THE SOURCE CODE

The source code used to perform image registration using the starbyte transformation is provided in this appendix.

The flow charts for UICP and AUTOCP in chapter 6 of the thesis show that there are various components comprising the process of performing image registration using the starbyte transformation. These various components have been written using C and Matlab code. These various C and Matlab programs have to be called in a certain sequence in order to finally obtain the unwarped image.

We describe below the sequence of calling these C and Matlab programs in order to perform image registration using UICP and AUTOCP. This code was written on a Unix platform. Hence, in order to run the code on other platforms, header files in the C programs may have to be changed. Also certain function calls may have to be changed in both the Matlab and C programs.

All C files have the extension “.c”, while all Matlab files have the extension “.m”. The description and function of every C and Matlab program is provided in the file containing the program. The description also provides the list and type of inputs expected by the program as well as the outputs of the program. All image files typically have the extension “.asc”, while data files have the extension “.dat”. The screen outputs of Matlab programs typically go into files with an extension “.out”. All C files have to be compiled to form executable versions before they can be used. The executable versions typically have the same file name as the source code file without the extension. For simplicity, all functions being used by a C program are included in the same file. Matlab programs like “uicp” have to be run from within the Matlab environment. However, Matlab programs which do not require graphics can be run as batch jobs from the shell tool command line using the shell program

`"matcomb"`.

Wherever a sequence of Matlab and C programs have to be called in sequence. Unix shell programs have been provided. The Unix shell program with the name `"filename"` can be run from the command line by typing the word `"filename"` at the command line followed by the inputs required by the shell program in the correct order separated by spaces. The inputs required for each shell program are described in the files containing the shell programs.

All executable programs can be maintained in one directory, whose path can be included in the main Unix path, so that these programs can be run from any other directory. In the case of C programs, this can be done, by suitably changing the `.login` file on Unix systems. To use Matlab functions from other directories, the Matlab path has to be changed. For the shell programs which call Matlab functions, the path to these functions can be included in the main Matlab path using the `"path"` command as follows:

`path(path,'Matlabfunctionpathname')`. In the shell programs provided, Matlab commands are typically `"echoed"` into a Matlab script file, which is then run using the command `"matcomb"`. When the above path command has also to be included in this script file, it is done as follows:

```
echo "path(path,'Matlabfunctionpathname')">> scriptfilename.m
```

Sequence of C and Matlab programs for image registration using UICP

Firstly the program `"starbyte"` is run on the original images to obtain the starbyte-transformed images (STs). The batch program `"starbytebatch"` can be used to run the starbyte program. The density slices of these STs have to be separated. Also the pixel values in each binary slice have to be either 0 or 255 before they can be cleaned

using the mathematical morphology package "morph".

For e.g., if a 4-bit protocol has been used, the starbyte transformation would return images having values between 0 and 15. The 16 slices making up the STs have to be separated out. The program "densityslice" returns a specified slice and converts each non-zero value in the slice to 255. For e.g. if the required slice was the slice # 12, then every "12" in the final slice is converted to the number "255".

Next the various density slices of the STs have to be cleaned using the freely distributable package "morph". The package "morph" has been written by Richard Alan Peters II, Department of Electrical Engineering, Vanderbilt University School of Engineering, Nashville, TN 37235. (email: rap2@vuse.vanderbilt.edu)

This software can be downloaded from the anonymous ftp site 129.59.100.16. The software is present in the compressed tar file *morph.tar.Z* in the */pub* directory. After downloading, the tar file can be uncompressed and untarred using standard Unix commands. Thence forward, the instructions in the README and makefiles can be followed to create the package.

After the morphological operation, every "255" in the slice is converted back to "12" using a C program "convertvalue". This is done for each of the 16 slices. After the 16 slices have been individually smoothened using the morphology package, they are added together using a Matlab program called "addimage.m". A small Unix shell program "clean" has been written which calls these programs in sequence. The various individual programs have been called through small batch programs. In the shell program, it is assumed that the executable version of each of these programs is available.

After the STs have been smoothened using "clean", matching regions are extracted from the STs using the Matlab program *uicp.m* (which has to be run from within Matlab). The outputs of this program (described in the file *uicp.m*), are inputs

to the C program "manualregion", which performs region aggregation and centroid calculation and returns the list of control points as a 4-column vector. This list of control points is loaded into Matlab, along with the reference image. The final unwarped image is then calculated using the Matlab function "splnewpimg.m", the list of control points and the reference image. The unwarped image is written into a specified file using the Matlab function "textwrite". The shell program "uicp" calls "manualregion", "splnewpimg" and "textwrite" in sequence.

In summary, in order to perform image registration using UICP, first the program "starbyte" has to be run on the original images, followed by "clean" to clean the STs, followed by uicp.m (within Matlab) to extract matching regions, followed by the Unix shell program "uicp" to finally obtain the unwarped image.

Sequence of C and Matlab programs for image registration using AUTOCP

In this case too, the programs "starbyte" and "clean" have to be run to obtain cleaned STs of the original reference and target images. To run AUTOCP in non-iterative mode, the Unix shell program "autocp" can be run to obtain the set of control points. The final unwarped image can be obtained by calling "splnewpimg" and "textwrite" from within Matlab, after loading in the reference image and the list of control points returned by "autocp". To run AUTOCP in iterative mode, the Unix shell program "iautocp" is run (after running "autocp"), once for every iteration. Again the final unwarped image can be obtained by calling "splnewpimg" and "textwrite" from within Matlab, after loading in the reference image and the list of control points.

/*

FILE NAME: AUTOREGION.C

WRITTEN BY: RADHIKA SIVARAMAKRISHNA

This program reads the cleaned STs of the reference and target image, density slices these images at each density level, performs region aggregation for every region in each of these slices for both images. For every region, aggregated, the centroid is calculated. The X and Y locations of the centroid, Hu's first invariant, as well as the number of pixels in the region are stored in a struct type variable. Parameters for regions having a size less than a threshold "t" specified by the user are not saved. The program then writes the 4-column list of regions corresponding to each slice of the target and reference image into two separate files. It also writes a count file where each row has two columns, the first column being the number of regions in the particular density slice in the target image and the second column being the number of regions in the same density slice in the second image.

THIS PROGRAM CAN BE COMPILED ON ANY KIND OF SYSTEM, THOUGH HOWEVER INCLUDE FILES MAY HAVE TO BE CHANGED FOR SYSTEMS OTHER THAN UNIX.

INPUTS TO THIS PROGRAM

THIS PROGRAM HAS 7 INPUTS

1. INPUT FILE CONTAINING THE TARGET IMAGE (HAS TO BE IN RAW TEXT FORMAT).
2. INPUT FILE CONTAINING THE REFERENCE IMAGE (HAS TO BE IN RAW TEXT FORMAT).
- 3 AND 4: OUTPUT FILES WHICH WILL CONTAIN THE 4-COLUMN LIST CORRESPONDING TO THE TARGET AND REFERENCE IMAGE.
5. THE OUTPUT FILE WHICH WILL CONTAIN THE NUMBER OF REGIONS IN EACH DENSITY SLICE OF THE TARGET AND REFERENCE IMAGE.
- 6 THRESHOLD (IN PIXELS) WHICH SPECIFIES THE MINIMUM SIZE OF EACH REGION. REGIONS BELOW THIS SIZE ARE DISCARDED.
7. IMAGE SIZE (ASSUMED TO BE SQUARE).

*/

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define REGION_SIZE 4000 /* Maximum region size */
#define NO_OF_REGIONS 200 /* Maximum number of regions */
#define NUMBER_FILES 15 /* Number of density levels */
```

/* rg_struct below is used to store the the number of pixels in the region, the X and Y centroid locations of each region, as well as Hu's first invariant calculated over the region.

*/

```
typedef struct [
int no_of_pixels;
float x_centroid;
float y_centroid;
float m_1;
] rg_struct;
int IMG_SIZE;
```

```
main()
{
int i,j,k,l;
int **yes_array;
int **image1,**image2,**image1tot,**image2tot;
int dummy;
FILE *in1,*in2,*out1,*out2,*cf;
float xhat,yhat,xNIH,yNIH,xNIHwprev,yNIHwprev;
char infile1[30],infile2[30],outfile1[30],outfile2[30],countfile[30];
int rg_ctr1;
int rg_ctr2;
float err;
rg_struct *rg_infol;
```

```

rg_struct *rg_info2;
float err2[5];
float min_m1,min_m2,min_m3,max_m1,max_m2,max_m3;
int ct_file,count_array[NUMBER_FILES + 1][2];
int t;

```

/* The following are a set of functions used by the program. They are described immediately above their respective sections of code. All these functions and the main program are present in the same file for simplicity. It is also possible to maintain the functions in a separate file which can be linked to the main one after compilation to create the executable code.

```

void grow_region(int **image,int i,int j,int **yes_array,int *count);
void mask_region(int **image,int **yes_array,int count,int mskrg);
void centroid(int **yes_array,int count,float *i,float *j);
void reorder_count(rg_struct *rg_info,int rg_ctr);
float cmpq(int **a,int c,int p,int q,float x,float y);
float ncmpq(int **a,int c,int p,int q,float x,float y);
float moment_inv_1(int **a,int c,float x,float y);

```

/* Inputs are read */

```

printf("\nGive name of input file 1: ");
scanf("%s",infile1);
printf("\nGive name of input file 2: ");
scanf("%s",infile2);
printf("\nGive name of output file 1: ");
scanf("%s",outfile1);
printf("\nGive name of output file 2: ");
scanf("%s",outfile2);
printf("\nGive name of output count file: ");
scanf("%s",countfile);
printf("\nGive cutoff size for regions: ");
scanf("%d",&t);
printf("\nGive size of image:(must be square and even): ");
scanf("%d",&IMG_SIZE);

```

/* The input and output files are opened and memory is allocated to the input images as well as to temporary matrices which store the reference and target density slices as an index moves through each possible value of the density. Memory is allocated to the list which will contain the co-ordinates of all pixels in a region. Memory is also allocated to the lists of struct type variables which will contain the 4 parameters representing each region in a slice.

*/

```

in1 = fopen(infile1,"r");
if (!in1)
{
printf("\nCouldnt open input file 1\n");
exit(1);
}
in2 = fopen(infile2,"r");
if (!in2)
{
printf("\nCouldnt open input file 2\n");
exit(1);
}
out1 = fopen(outfile1,"w");
if (!out1)
{
printf("\nCouldnt open output file 1\n");
exit(1);
}
out2 = fopen(outfile2,"w");
if (!out2)
{
printf("\nCouldnt open output file 2\n");
exit(1);
}
cf = fopen(countfile,"w");
if (!cf)
{

```

```
printf("\nCouldnt open output count file\n");
exit(1);
}

imageltot = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!imageltot)
{
    printf("\nCouldnt allocate block of memory for image 1 Total");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    imageltot[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!imageltot[i])
    {
        printf("\nCouldnt allocate %d block of memory for image 1 Total", i);
        exit(1);
    }
}
image2tot = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2tot)
{
    printf("\nCouldnt allocate block of memory for image2 Total");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image2tot[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image2tot[i])
    {
        printf("\nCouldnt allocate %d block of memory for image2 Total", i);
        exit(1);
    }
}
image1 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image1)
{
    printf("\nCouldnt allocate block of memory for image 1");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image1[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image1[i])
    {
        printf("\nCouldnt allocate %d block of memory for image 1", i);
        exit(1);
    }
}
image2 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2)
{
    printf("\nCouldnt allocate block of memory for image2");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image2[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image2[i])
    {
        printf("\nCouldnt allocate %d block of memory for image2", i);
        exit(1);
    }
}
yes_array = (int **)calloc((size_t)REGION_SIZE, (size_t)sizeof(int *));
if (!yes_array)
{
    printf("\nCouldnt allocate block of memory for yes_array");
    exit(1);
}
for(i=0; i< REGION_SIZE; i++)
{
    yes_array[i] = (int *)calloc((size_t)2, (size_t)sizeof(int));
    if (!yes_array[i])
```

```

        {
            printf("\nCouldnt allocate %d block of memory for yes_array",i);
            exit(1);
        }
    }
    rg_info1 = (rg_struct *)calloc((size_t)NO_OF_REGIONS,(size_t)sizeof(rg_struct));
    if (!rg_info1)
    {
        printf("\nCouldnt allocate block of memory for rg_info");
        exit(1);
    }
    rg_info2 = (rg_struct *)calloc((size_t)NO_OF_REGIONS,(size_t)sizeof(rg_struct));
    if (!rg_info2)
    {
        printf("\nCouldnt allocate block of memory for rg_info");
        exit(1);
    }
    /* The two input files are read */
    for(i=0;i<IMG_SIZE;i++)
        for(j=0;j<IMG_SIZE;j++)
            fscanf(in1,"%d",&image1tot[i][j]);
    for(i=0;i<IMG_SIZE;i++)
        for(j=0;j<IMG_SIZE;j++)
            fscanf(in2,"%d",&image2tot[i][j]);

    /* For each possible value of density (indexed using the variable ct_file
    going from 0 to NUMBER_FILES), we density slice the reference and
    target at the particular density level. These density slices (at a
    particular density level) for the target and reference are stored in
    temporary matrices. For each density slice in the target image, pixels
    are examined sequentially. As soon as a non-zero pixel is encountered,
    it becomes the seed pixel for a region aggregation operation. After
    region aggregation, if the number of pixels in the region is greater
    than the specified threshold "t", the parameters of the region (X and Y
    locations of the centroid, Hu's first invariant and the number of
    pixels in the region) are stored in the list rg_info1 (indexed by the
    integer variable rg_ctrl). The region is then "erased" from the density
    slice using the mask_region command. The sequential tracking operation
    continues with non-zero pixels becoming seed points for region
    aggregation. In this way a list of regions (represented by their 4
    parameters) is built in the variable rg_info1. The regions are then
    re-ordered in decreasing order of their number of pixels. The process
    is repeated for the density slice of the reference image. The variable
    count_array (a 2-column vector) stores the number of regions in the
    target and reference image for each slice pair. The lists of 4
    parameters for each slice for the reference and the target images are
    written into the specified output files. The entire procedure is then
    repeated for each density slice pair. Finally the number of regions in
    each slice of the target and reference (i.e. count_array) is written
    into the specified output count file.

    */

    for (ct_file = 1;ct_file <= NUMBER_FILES;ct_file++)
    {
        for(i=0;i<IMG_SIZE;i++)
            for(j=0;j<IMG_SIZE;j++)
                if (image1tot[i][j] == ct_file)
                    image1[i][j] = ct_file;
        for(i=0;i<IMG_SIZE;i++)
            for(j=0;j<IMG_SIZE;j++)
                if (image2tot[i][j] == ct_file)
                    image2[i][j] = ct_file;

        count_array[ct_file][0] = 0;
        rg_ctrl = 0;
        for(i=0;i < IMG_SIZE;i++)
        {
            for(j=0;j < IMG_SIZE;j++)
            {
                if(image1[i][j] != 0)
                {
                    grow_region(image1,i,j,yes_array,&rg_info1[rg_ctrl].no_of_pixels);

```

```

        mask_region(image1,yes_array,rg_infol[rg_ctrl].no_of_pixels,1);
        if (rg_infol[rg_ctrl].no_of_pixels >= t)
        {
            centroid(yes_array,rg_infol[rg_ctrl].no_of_pixels,
                &(rg_infol[rg_ctrl].x_centroid),&(rg_infol[rg_ctrl].y_centroid));
            rg_infol[rg_ctrl].m_1 = moment_inv_1(yes_array,rg_infol[rg_ctrl].
no_of_pixels,
            rg_infol[rg_ctrl].x_centroid,rg_infol[rg_ctrl].y_centroid);
            rg_ctrl++;
            if(rg_ctrl >= NO_OF_REGIONS)
            {
                printf("\nNO_OF_REGIONS requires larger value for first i
mage\n");
                exit(1);
            }
        }
    }
}

reorder_count(rg_infol,rg_ctrl);
count_array[ct_file][0] += rg_ctrl;

count_array[ct_file][1] = 0;
rg_ctr2 = 0;
for(i=0;i < IMG_SIZE;i++)
{
    for(j=0;j < IMG_SIZE;j++)
    {
        if(image2[i][j] != 0)
        {
            grow_region(image2,i,j,yes_array,&rg_info2[rg_ctr2].no_of_pixels);
            mask_region(image2,yes_array,rg_info2[rg_ctr2].no_of_pixels,1);
            if (rg_info2[rg_ctr2].no_of_pixels >= t)
            {
                centroid(yes_array,rg_info2[rg_ctr2].no_of_pixels,&(rg_info2[rg_c
tr2].x_centroid),
                &(rg_info2[rg_ctr2].y_centroid));
                rg_info2[rg_ctr2].m_1 = moment_inv_1(yes_array,rg_info2[rg_ctr2].
no_of_pixels,
                rg_info2[rg_ctr2].x_centroid,rg_info2[rg_ctr2].y_centroid);
                rg_ctr2++;
                if(rg_ctr2 >= NO_OF_REGIONS)
                {
                    printf("\nNO_OF_REGIONS requires larger value for second
image\n");
                    exit(1);
                }
            }
        }
    }
}

reorder_count(rg_info2,rg_ctr2);
count_array[ct_file][1] += rg_ctr2;

for(i = 0;i < rg_ctrl;i++)
{
    fprintf(out1,"%9.3f %9.3f %9.3f %3d\n",
        rg_infol[i].x_centroid,rg_infol[i].y_centroid,
        rg_infol[i].m_1,rg_infol[i].no_of_pixels);
}

for(i = 0;i < rg_ctr2;i++)
    fprintf(out2,"%9.3f %9.3f %9.3f %3d\n",
        rg_info2[i].x_centroid,rg_info2[i].y_centroid,
        rg_info2[i].m_1,rg_info2[i].no_of_pixels);
}

for(ct_file = 1;ct_file <= NUMBER_FILES;ct_file++)
    fprintf(cf,"%d %d\n",count_array[ct_file][0],
        count_array[ct_file][1]);
}

```

/* For each non-zero value in a density slice, region aggregation is performed using the following function. This program is essentially the same as the grow_region function in the file manualregion.c.

The inputs to this function are:

image: Contains density slice on which region aggregation has to be performed.

i and j: X and Y coordinates of seed point.

yes_array: Array to store pixels in the region, as the region is growing.

current_count: Pointer to the variable which will contain the count of the number of pixels in the region.

```

*/
void grow_region(image,i,j,yes_array,current_count)
int i,j,**image;
int **yes_array,*current_count;
{
    int count,k,l,**done_array;
    int count1,count2;

    done_array = (int **)calloc((size_t)IMG_SIZE,(size_t)sizeof(int *));
    if (!done_array)
    {
        printf("\nCouldnt allocate block of memory for done_array");
        exit(1);
    }
    for(count1=0;count1< IMG_SIZE;count1++)
    {
        done_array[count1] = (int *)calloc((size_t)IMG_SIZE,(size_t)sizeof(int));
        if (!done_array[count1])
        {
            printf("\nCouldnt allocate %d block of memory for done_array",count1);
            exit(1);
        }
    }

    /* Since each density slice is binary, the region aggregation operation is
    simple and simply accumulates all pixels in yes_array, which are
    non_zero. At the same time, pixels which have already been accumulated
    are marked using the matrix done_array, so that they are not
    re-processed. The function uses an 8-neighbor criterion for region
    growing. If the number of pixels accumulated exceeds the specified
    size of yes_array (constant REGION_SIZE), then the function prints out
    an error message and the program quits.
    */

    count = 0;
    *current_count = count;
    yes_array[count][0] = i;
    yes_array[count][1] = j;
    done_array[i][j] = 1;
    do
    {
        for(k=i-1;k<=i+1;k++)
        {
            for(l=j-1;l<=j+1;l++)
            {
                if(((k==i) && (l==j)) || (k >= IMG_SIZE) || ((l >= IMG_SIZE)) || (k < 0)
                || (l < 0))
                {
                    ;
                }
                else
                {
                    if (done_array[k][l] == 1)
                    {
                        ;
                    }
                    else
                    {
                        done_array[k][l] = 1;
                        if (image[k][l] == image[i][j])
                        {
                            count++;
                            if (count >= REGION_SIZE)
                            {
                                printf("\nYes_array size not enough\n");
                                exit(1);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

                                yes_array[count][0] = k;
                                yes_array[count][1] = l;
                                }
                                }
                                }
                                (*current_count)++;
                                i = yes_array[*current_count][0];
                                j = yes_array[*current_count][1];
                                }
                                while(*current_count <= count);

                                for(count1=0;count1< IMG_SIZE;count1++)
                                    free(done_array[count1]);
                                free(done_array);
                                }

/* This function is used to "wipe out" a region from a density slice,
after it has been aggregated and its 4 parameters have been calculated.
Since sequential tracking is performed, this operation is necessary so
that the same region does not get aggregated twice.

The inputs to this function are:
image: given density slice
yes_array: contains list of X and Y co-ordinates of pixels which have to
           "wiped out" in the density slice.
count: Number of pixels in the region.
mskrq: Variable which has a 0 or 1 value. If it has a 1 value, pixels in the
density slice at locations specified in yes_array are set to 0,
i.e. a true region masking operation is performed. Otherwise,
the pixels are set to 1, i.e., an inverse region masking
operation is performed.

*/

void mask_region(image,yes_array,count,mskrq)
int **image,**yes_array,count,mskrq;
{
    int i,fill_value;
    if (mskrq == 1)
        fill_value = 0;
    else
        fill_value = 1;
    for(i=0;i<count;i++)
        image[yes_array[i][0]][yes_array[i][1]] = fill_value;
}

/* This function calculates the centroid of a given region. The region is
represented by a list of the X and Y co-ordinates of the pixels
constituting the region. The region then calculates the centroid of the
region using standard formulae. This function is the same as the
function centroid in the file manualregion.c.

Inputs to this function are:
yes_array: List containing X and Y co-ordinates of pixels in the region
count: Number of pixels in the region
i and j: pointers to X and Y co-ordinates of centroid of the region
         (returned by the function).

*/

void centroid(yes_array,count,i,j)
int **yes_array,count;
float *i,*j;
{
    int k;

    *i = 0.0;
    *j = 0.0;
    for(k=0;k<count;k++)
    {
        *i += (float)yes_array[k][0];
        *j += (float)yes_array[k][1];
    }
    *i = *i/(float)count;

```

```

*j = *j/(float)count;
}
/* Below is the code for reordering the array containing the regions in
each slice (represented by a 4-parameter list), such that the number of
pixels in the regions are in decreasing order.

```

Inputs to this function are:

rg_info: array containing 4-parameter list of regions in a density slice
rg_ctr: Number of regions in the slice.

```

*/

```

```

void reorder_count(rg_info,rg_ctr)
rg_struct *rg_info;
int rg_ctr;
{
    int item,i;
    rg_struct temp;
    for(item=0;item < rg_ctr - 1;++item)
        for(i=item+1;i < rg_ctr;++i)
            if (rg_info[i].no_of_pixels > rg_info[item].no_of_pixels)
            {
                temp = rg_info[item];
                rg_info[item] = rg_info[i];
                rg_info[i] = temp;
            }
}

```

/* The function cmpq, ncmpq, and moment_inv_1 below which calculate the central moment of a given order, the normalized central moment of a given order and Hu's first invariant for a given region use formulae given in standard image processing textbooks.

```

*/

```

/* The function below calculates the central moment of order (p,q) of a given region represented as a list of X and Y values in variable "a", with (x,y) being the centroid of the region, and "c" containing the number of pixels in the region.

Inputs to this function are:

a: list of pixels constituting the given region
c: number of pixels in the region
p and q: Order of central moment
x and y: X and Y locations of the centroid of the region

```

*/

```

```

float cmpq(int **a,int c,int p,int q,float x,float y)
{
    float mu;
    int i;
    mu = 0.0;
    for(i=0;i < c;i++)
        mu += (float)(pow((double)((float)a[i][0] - x),(double)p)*pow((double)((float)a[i][1] - y),(double)q));
    return mu;
}

```

/* The function below calculates the normalized central moment of order (p,q) of a given region represented as a list of X and Y values in variable "a", with (x,y) being the centroid of the region, and "c" containing the number of pixels in the region.

Inputs to this function are:

a: list of pixels constituting the given region
c: number of pixels in the region
p and q: Order of normalized central moment
x and y: X and Y locations of the centroid of the region

```

*/

```



```
float ncmpq(int **a,int c,int p,int q,float x,float y)
{
    float eta;
    float gamma;
    float num,den;
    num = cmpq(a,c,p,q,x,y);
    den = cmpq(a,c,0,0,x,y);
    gamma = (((float)(p+q))/2.0) + 1.0;
    eta = num/((float)pow((double)den,(double)gamma));
    return eta;
}
/* This function calculates Hu's first invariant of a given region
represented as a list of X and Y values in variable "a", with (x,y)
being the centroid of the region, and "c" containing the number of
pixels in the region.

Inputs to this function are:
a: list of pixels constituting the given region
c: number of pixels in the region
x and y: X and Y locations of the centroid of the region
*/

float moment_inv_1(int **a,int c,float x,float y)
{
    float phi;
    phi = ncmpq(a,c,0,2,x,y) + ncmpq(a,c,2,0,x,y);
    return phi;
}
```

```
/* WRITTEN BY: RADHIKA SIVARAMAKRISHNA
FILE NAME: CONVERTVALUES.C
```

Given an input binary image where every pixel is either 0 or 255, this program tests each pixel and if it is 0, the corresponding output pixel too is 0. Else, the corresponding output pixel is set to a desired value. This is used in the starbyte registration algorithm to convert every density slice back to its original value after the morphological operation.

```
INPUTS TO THIS PROGRAM
```

```
-----
THIS PROGRAM HAS 4 INPUTS.
```

1. INPUT FILE (HAS TO BE SQUARE AND IN RAW TEXT FORMAT)
2. OUTPUT FILE (WILL BE WRITTEN OUT IN RAW TEXT FORMAT)
3. SIZE OF THE IMAGE
4. GREY LEVEL TO SET ALL "255" VALUES TO.

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
main() {
    int i,j,k,l;
    int **image1,**image2;
    FILE *in,*out;
    char infile[20],outfile[20];
    int IMG_SIZE;
    int gray_level;
    int ans;
```

```
/* Inputs are read */
```

```
printf("\nGive name of input file: ");
scanf("%s",infile);
printf("\nGive name of output file: ");
scanf("%s",outfile);
printf("\nGive size of input and output image:(must be square and even): ");
scanf("%d",&IMG_SIZE);
printf("\nAll '255' will be converted to what? ");
scanf("%d",&gray_level);
```

```
/* Input and output files are opened and memory is allocated to the input
and output images */
```

```
in = fopen(infile,"r");
if (!in)
{
    printf("\nCouldnt open input file\n");
    exit(1);
}
out = fopen(outfile,"w");
if (!out)
{
    printf("\nCouldnt open output file\n");
    exit(1);
}
```

```
image1 = (int **)calloc((size_t)IMG_SIZE,(size_t)sizeof(int *));
```

```
if (!image1)
```

```
{
    printf("\nCouldnt allocate block of memory for image 1");
    exit(1);
}
```

```
for(i=0;i< IMG_SIZE;i++)
```

```
{
    image1[i] = (int *)calloc((size_t)IMG_SIZE,(size_t)sizeof(int));
    if (!image1[i])
    {
        printf("\nCouldnt allocate %d block of memory for image 1",i);
        exit(1);
    }
}
```

```
    }
}
image2 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2)
{
    printf("\nCouldnt allocate block of memory for image2");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image2[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image2[i])
    {
        printf("\nCouldnt allocate %d block of memory for image2", i);
        exit(1);
    }
}

/* The input image is read */
for(i=0; i<IMG_SIZE; i++)
    for(j=0; j<IMG_SIZE; j++)
        fscanf(in, "%d", &image1[i][j]);

/* Each pixel in the input image is tested to see if it is either 0 or 255. If it is 255, it is set to the specified value (4th input), else it remains 0. */

for(i=0; i < IMG_SIZE; i++)
{
    for(j=0; j<IMG_SIZE; j++)
    {
        if(image1[i][j] == 255)

            image2[i][j] = gray_level;
        else
            image2[i][j] = 0;
    }
}

/* The output image is written into the specified file. */
for(i=0; i<IMG_SIZE; i++)
{
    for(j=0; j<IMG_SIZE; j++)
        fprintf(out, "%d ", image2[i][j]);
    fprintf(out, "\n");
}
}
```

```
/*
WRITTEN BY: RADHIKA SIVARAMAKRISHNA
FILE NAME: DENSITYSLICE.C
```

This program returns the specified density slice of a given image. Giving an input image which contains values between a minimum and a maximum value, this program returns an output image, which has non-zero values only in a particular range of the input image and 0 elsewhere. For example, if we specify that we want to only highlight those pixels in the input image which have values between 10 and 15, then every pixel in the input image is tested to see if it lies at or between these two values. If it doesn't, the corresponding pixel in the output image is set to 0, else the output value is either set to the same value as in the input image, or to 255. In the context of the starbyte transformation, if we specify the bounds for the density slicing operation to be the same, we are effectively creating an output image which highlights only one grey value in the input image. Thus we can create a binary image which is a map of a particular grey level in the input image.

INPUTS TO THIS PROGRAM

THIS PROGRAM HAS 6 INPUTS

1. INPUT FILE (HAS TO BE IN RAW TEXT FORMAT)
2. OUTPUT FILE (WILL BE WRITTEN OUT AS A RAW TEXT FILE)
3. SIZE OF INPUT AND OUTPUT FILES. ALL INPUT AND OUTPUT FILES ARE ASSUMED TO BE SQUARE. HENCE ONLY ONE NUMBER HAS TO BE INPUTTED AT THIS PROMPT
- 4 & 5. MINIMUM AND MAXIMUM BOUNDS FOR DENSITYSLICING. INPUT THESE NUMBERS ON THE SAME LINE WITH A SPACE SEPARATING THEM
6. THIS INPUT IS EITHER TO BE 0 OR 1. IF IT IS 1 THEN ORIGINAL GREY VALUES BETWEEN THE BOUNDS ARE RETURNED AT THE APPROPRIATE PIXEL LOCATIONS, ELSE ALL HIGHLIGHTED PIXELS ARE SET TO 255.

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
main() {
int i,j,k,l;
int **image1,**image2;
FILE *in,*out;
char infile[20],outfile[20];
int IMG_SIZE;
int density_slice_min,density_slice_max;
int ans;
```

```
/* Inputs are read */
```

```
printf("\nGive name of input file: ");
scanf("%s",infile);
printf("\nGive name of output file: ");
scanf("%s",outfile);
printf("\nGive size of input and output image:(must be square and even): ");
scanf("%d",&IMG_SIZE);
printf("\nGive gray levels to density slice at (min and max): ");
scanf("%d %d",&density_slice_min,&density_slice_max);
printf("\nInput image values in density slice:y/n (if no, then all 255) ");
scanf("%d",&ans);
```

```
/* Input and output files are opened and memory is allocated to the input
and output images */
```

```
in = fopen(infile,"r");
if (!in)
{
printf("\nCouldnt open input file\n");
exit(1);
}
out = fopen(outfile,"w");
if (!out)
{
```

```

printf("\nCouldnt open output file\n");
exit(1);
}

image1 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image1)
{
    printf("\nCouldnt allocate block of memory for image 1");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image1[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image1[i])
    {
        printf("\nCouldnt allocate %d block of memory for image 1", i);
        exit(1);
    }
}
image2 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2)
{
    printf("\nCouldnt allocate block of memory for image2");
    exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
    image2[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
    if (!image2[i])
    {
        printf("\nCouldnt allocate %d block of memory for image2", i);
        exit(1);
    }
}

/* The input image is read. */
for(i=0; i<IMG_SIZE; i++)
    for(j=0; j<IMG_SIZE; j++)
        fscanf(in, "%d", &image1[i][j]);

/* Each pixel in the input image is compared to the bound values. If it falls
within the bound limits, its value in the output image is set either to
the original value or to 255 depending upon the specification of the 6th
input to this program */
for(i=0; i < IMG_SIZE; i++)
{
    for(j=0; j<IMG_SIZE; j++)
    {
        if((image1[i][j] >= density_slice_min) && (image1[i][j] <= density_slice_max))
        {
            if(ans == 1)
                image2[i][j] = image1[i][j];
            else
                image2[i][j] = 255;
        }
        else
            image2[i][j] = 0;
    }
}

/* The output image is written into the specified file. */
for(i=0; i<IMG_SIZE; i++)
{
    for(j=0; j<IMG_SIZE; j++)
        fprintf(out, "%d ", image2[i][j]);
    fprintf(out, "\n");
}

```

```
/*
FILE NAME: MANUALREGION.C
```

```
WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

This program should be used after seed points have been extracted using the program uicp.m. This program performs region growing using the seed points specified through UICP, checks for duplicates and returns a 4-column list of control points. The first two columns represent the X and Y location of the centroid of the regions in the target images, while the last two columns represent the same for the reference image. The output of this program can then be used with the Matlab program splinewpimg.m and the reference image to obtain the final unwarped image.

THIS PROGRAM CAN BE COMPILED ON ANY KIND OF SYSTEM, THOUGH HOWEVER INCLUDE FILES MAY HAVE TO BE CHANGED FOR SYSTEMS OTHER THAN UNIX.

```
INPUTS TO THIS PROGRAM
-----
```

THIS PROGRAM HAS 6 INPUTS

1. INPUT FILE CONTAINING THE TARGET IMAGE (HAS TO BE IN RAW TEXT FORMAT)
2. INPUT FILE CONTAINING THE REFERENCE IMAGE (HAS TO BE IN RAW TEXT FORMAT)
3. COORDINATES DATA FILE (OUTPUT FROM UICP)
4. COUNT FILE (OUTPUT FROM UICP)

ALL THE ABOVE FILES ARE OUTPUTS OF THE UICP PROGRAM.

5. OUTPUT FILE CONTAINING 4-COLUMN LIST OF CONTROL POINTS
6. IMAGE SIZE (ASSUMED TO BE SQUARE).

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define REGION_SIZE 3000 /* Maximum region size (pixels) */
#define NUMBER_FILES 15 /* Number of density levels */
#define MAX_NO_SP_RGS 10 /* Maximum number of fragments in a given region */
```

```
/* Below are macros to define the square of a given quantity as well as the Euclidean distance between two points */
```

```
#define square(x) (x)*(x)
#define euc_dist_2(x1,y1,x2,y2) sqrt((square((x1) - (x2))) + (square((y1) - (y2))))
```

```
/* rg_struct below is used to store the X and Y centroid locations of each region. It also has a variable "dup_index", which is used to eliminate duplicates.
```

```
*/
```

```
typedef struct {
int no_of_pixels;
double x_centroid;
double y_centroid;
int dup_index;
} rg_struct;
int IMG_SIZE;
int NO_OF_REGIONS;
```

```
main()
{
int i,j,k,l;
int **yes_array;
int **image1,**image2,**image1tot,**image2tot;
int dummy;
FILE *in1,*in2,*in3,*out1,*out2,*cf;
double xhat,yhat,xNIH,yNIH,xNIHwprev,yNIHwprev;
/* Below cn1,cn2,cn3 and varstring are used to create variable names of
```

```

files */
char infile1[20],infile2[20],infile3[20];
char outfile1[20],countfile[20];
int rg_ctr;
double err;
rg_struct *rg_infol;
rg_struct *rg_info2;
double err2[5];
int no_of_split_regions_img1,no_of_split_regions_img2;
int sp_rg_1[MAX_NO_SP_RGS][3],sp_rg_2[MAX_NO_SP_RGS][3];
double sub_ct_1[MAX_NO_SP_RGS][3],sub_ct_2[MAX_NO_SP_RGS][3];
int count;
int count_array[NUMBER_FILES + 1];
double min_m1,min_m2,min_m3,max_m1,max_m2,max_m3;
int ct;
int n_of_r_in_e_file;
double dist1,dist2;

/* The following are a set of functions used by the program. They are
described immediately above their respective sections of code. All
these functions and the main program are present in the same file for
simplicity. It is also possible to maintain the functions in a separate
file which can be linked to the main one after compilation to create
the executable code.

*/

void non_zero_8_neighbor(int ** image,int i,int j,int *in,int *jn);
void grow_region(int **image,int i,int j,int **yes_array,int *count);
void centroid(int **yes_array,int count,double *i,double *j);

/* The inputs are read */

printf("\nGive name of input file 1: ");
scanf("%s",infile1);
printf("\nGive name of input file 2: ");
scanf("%s",infile2);
printf("\nGive name of coordinates data file: ");
scanf("%s",infile3);
printf("\nGive name of count file: ");
scanf("%s",countfile);
printf("\nGive name of output file 1: ");
scanf("%s",outfile1);
printf("\nGive size of image:(must be square and even): ");
scanf("%d",&IMG_SIZE);

/* The input files and the output file are opened and memory is allocated
to the input images as well as to temporary matrices which store the
reference and target density slices as an index moves through each
possible value of the density. Memory is also allocated to the list
which will contain the co-ordinates of all pixels in a region.

*/

in1 = fopen(infile1,"r");
if (!in1)
{
printf("\nCouldnt open input file 1\n");
exit(1);
}
in2 = fopen(infile2,"r");
if (!in2)
{
printf("\nCouldnt open input file 2\n");
exit(1);
}
in3 = fopen(infile3,"r");
if (!in3)
{
printf("\nCouldnt open coordinates data file\n");
exit(1);
}
cf = fopen(countfile,"r");
if (!cf)
{

```

```
printf("\nCouldnt open count file\n");
exit(1);
}
out1 = fopen(outfile1, "w");
if (!out1)
{
printf("\nCouldnt open output file 1\n");
exit(1);
}

image1 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image1)
{
printf("\nCouldnt allocate block of memory for image 1");
exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
image1[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
if (!image1[i])
{
printf("\nCouldnt allocate %d block of memory for image 1", i);
exit(1);
}
}
image2 = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2)
{
printf("\nCouldnt allocate block of memory for image2");
exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
image2[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
if (!image2[i])
{
printf("\nCouldnt allocate %d block of memory for image2", i);
exit(1);
}
}
imagetot = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!imagetot)
{
printf("\nCouldnt allocate block of memory for image 1 Total");
exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
imagetot[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
if (!imagetot[i])
{
printf("\nCouldnt allocate %d block of memory for image 1 Total", i);
exit(1);
}
}
image2tot = (int **)calloc((size_t)IMG_SIZE, (size_t)sizeof(int *));
if (!image2tot)
{
printf("\nCouldnt allocate block of memory for image2 Total");
exit(1);
}
for(i=0; i< IMG_SIZE; i++)
{
image2tot[i] = (int *)calloc((size_t)IMG_SIZE, (size_t)sizeof(int));
if (!image2tot[i])
{
printf("\nCouldnt allocate %d block of memory for image2 Total", i);
exit(1);
}
}
yes_array = (int **)calloc((size_t)REGION_SIZE, (size_t)sizeof(int *));
if (!yes_array)
{
printf("\nCouldnt allocate block of memory for yes_array");
exit(1);
}
```



```

}
for(i=0;i< REGION_SIZE;i++)
{
    yes_array[i] = (int *)calloc((size_t)2,(size_t)sizeof(int));
    if (!yes_array[i])
    {
        printf("\nCouldnt allocate %d block of memory for yes_array",i);
        exit(1);
    }
}
/* The input images and the count file are read. */
for(i=0;i<IMG_SIZE;i++)
    for(j=0;j<IMG_SIZE;j++)
        fscanf(in1,"%d",&image1tot[i][j]);
for(i=0;i<IMG_SIZE;i++)
    for(j=0;j<IMG_SIZE;j++)
        fscanf(in2,"%d",&image2tot[i][j]);

for(ct = 1;ct <=NUMBER_FILES;ct++)
    fscanf(cf,"%d",&count_array[ct]);

/* For each possible value of density (i.e. from 0 to NUMBER_FILES), we
density slice the reference and target at the particular density level.
The X and Y coordinates of the clicked points from UICP for that
particular slice are read in from the co-ordinates file. We then
perform region growing and calculate the centroid of the region in the
target and reference image. Since the co-ordinates file also indicates
the number of fragments making up a region, region aggregation and
centroid calculation are performed accordingly so that fragments of a
region can be grouped together as one unit.
*/

NO_OF_REGIONS = 0;
for(ct=1;ct <= NUMBER_FILES;ct++)
    NO_OF_REGIONS += count_array[ct];
rg_info1 = (rg_struct *)calloc((size_t)NO_OF_REGIONS,(size_t)sizeof(rg_struct));
if (!rg_info1)
{
    printf("\nCouldnt allocate block of memory for rg_info");
    exit(1);
}
rg_info2 = (rg_struct *)calloc((size_t)NO_OF_REGIONS,(size_t)sizeof(rg_struct));
if (!rg_info2)
{
    printf("\nCouldnt allocate block of memory for rg_info");
    exit(1);
}

rg_ctr = 0;
for (ct=1;ct <= NUMBER_FILES;ct++)
{
    for(i=0;i<IMG_SIZE;i++)
    {
        for(j=0;j<IMG_SIZE;j++)
        {
            if (image1tot[i][j] == ct)
                image1[i][j] = ct;
            else
                image1[i][j] = 0;
        }
    }
    for(i=0;i<IMG_SIZE;i++)
    {
        for(j=0;j<IMG_SIZE;j++)
        {
            if (image2tot[i][j] == ct)
                image2[i][j] = ct;
            else
                image2[i][j] = 0;
        }
    }
    n_of_r_in_e_file = 0;
}

```

```

while (n_of_r_in_e_file < count_array[ct])
{

    fscanf(in3,"%d",&no_of_split_regions_img1);
    for(i=0;i < no_of_split_regions_img1;i++)
    {

        fscanf(in3,"%d %d",&sp_rg_1[i][0],&sp_rg_1[i][1]);

    }
    count = 0;
    for(i=0;i < no_of_split_regions_img1;i++)
    {
        if(image1[sp_rg_1[i][0]][sp_rg_1[i][1]] == 0)
            non_zero_8_neighbor(image1,sp_rg_1[i][0],
                                sp_rg_1[i][1],&sp_rg_1[i][0],&sp_rg_1[i][1]);
        if(image1[sp_rg_1[i][0]][sp_rg_1[i][1]] == 0)
        {
            printf("\nZero region coordinates %d %d in Image 1, slice %d\n",
                sp_rg_1[i][0],sp_rg_1[i][1],ct);
            exit(1);
        }
        else
        {

            grow_region(image1,sp_rg_1[i][0],sp_rg_1[i][1],
                yes_array + count,&count);
            sp_rg_1[i][2] = count;
            if (i==0)
            {
                centroid(yes_array,sp_rg_1[i][2],
                    &sub_ct_1[i][0],&sub_ct_1[i][1]);
                sub_ct_1[i][2] = sp_rg_1[i][2];
            }
            else
            {
                centroid(yes_array + sp_rg_1[i-1][2],(sp_rg_1[i][2] -
                    sp_rg_1[i-1][2]),&sub_ct_1[i][0],&sub_ct_1[i][1]);

                sub_ct_1[i][2] = sp_rg_1[i][2] - sp_rg_1[i-1][2];
            }

        }

    }

}
for(i=0;i < no_of_split_regions_img1;i++)
{
    for(j = i+1;j < no_of_split_regions_img1;j++)
    {
        if ((sub_ct_1[i][0] == sub_ct_1[j][0]) &&
            (sub_ct_1[i][1] == sub_ct_1[j][1]))
        {
            sub_ct_1[j][0] = 0.0;
            sub_ct_1[j][1] = 0.0;
            sub_ct_1[j][2] = 0.0;
            count = count - (sp_rg_1[j][2] - sp_rg_1[j-1][2]);
        }

    }

}
rg_infol[rg_ctr].x_centroid = 0.0;
rg_infol[rg_ctr].y_centroid = 0.0;
for(i=0;i < no_of_split_regions_img1;i++)
{
    rg_infol[rg_ctr].x_centroid += sub_ct_1[i][0]*sub_ct_1[i][2];
    rg_infol[rg_ctr].y_centroid += sub_ct_1[i][1]*sub_ct_1[i][2];
}
rg_infol[rg_ctr].x_centroid /= (double)count;
rg_infol[rg_ctr].y_centroid /= (double)count;
rg_infol[rg_ctr].dup_index = 0;

fscanf(in3,"%d",&no_of_split_regions_img2);
for(i=0;i < no_of_split_regions_img2;i++)
{

    fscanf(in3,"%d %d",&sp_rg_2[i][0],&sp_rg_2[i][1]);

```

```

    }
    count = 0;
    for(i=0; i < no_of_split_regions_img2; i++)
    {
        if(image2[sp_rg_2[i][0]][sp_rg_2[i][1]] == 0)
            non_zero_8_neighbor(image2, sp_rg_2[i][0],
                                sp_rg_2[i][1], &sp_rg_2[i][0], &sp_rg_2[i][1]);
        if(image2[sp_rg_2[i][0]][sp_rg_2[i][1]] == 0)
        {
            printf("\nZero region coordinates %d %d in Image 2, slice %d\n",
                    sp_rg_2[i][0], sp_rg_2[i][1], ct);
            exit(1);
        }
        else
        {
            grow_region(image2, sp_rg_2[i][0], sp_rg_2[i][1],
                        yes_array + count, &count);
            sp_rg_2[i][2] = count;
            if (i==0)
            {
                centroid(yes_array, sp_rg_2[i][2],
                        &sub_ct_2[i][0], &sub_ct_2[i][1]);
                sub_ct_2[i][2] = sp_rg_2[i][2];
            }
            else
            {
                centroid(yes_array + sp_rg_2[i-1][2], (sp_rg_2[i][2] -
                    sp_rg_2[i-1][2]), &sub_ct_2[i][0], &sub_ct_2[i][1]);
                sub_ct_2[i][2] = sp_rg_2[i][2] - sp_rg_2[i-1][2];
            }
        }
    }
    for(i=0; i < no_of_split_regions_img2; i++)
    {
        for(j = i+1; j < no_of_split_regions_img2; j++)
        {
            if ((sub_ct_2[i][0] == sub_ct_2[j][0]) &&
                (sub_ct_2[i][1] == sub_ct_2[j][1]))
            {
                sub_ct_2[j][0] = 0.0;
                sub_ct_2[j][1] = 0.0;
                sub_ct_2[j][2] = 0.0;
                count = count - (sp_rg_2[j][2] - sp_rg_2[j-1][2]);
            }
        }
    }

    rg_info2[rg_ctr].x_centroid = 0.0;
    rg_info2[rg_ctr].y_centroid = 0.0;
    for(i=0; i < no_of_split_regions_img2; i++)
    {
        rg_info2[rg_ctr].x_centroid += sub_ct_2[i][0]*sub_ct_2[i][2];
        rg_info2[rg_ctr].y_centroid += sub_ct_2[i][1]*sub_ct_2[i][2];
    }
    rg_info2[rg_ctr].x_centroid /= (double)count;
    rg_info2[rg_ctr].y_centroid /= (double)count;
    rg_info2[rg_ctr].dup_index = 0;

    n_of_r_in_e_file++;
    rg_ctr++;
}

```

/* The loop below checks for duplicate regions. It may sometimes happen that due to incorrect clicking of points, a user may select the same region twice, or match the same region in one image to two different regions in the target image. The loop below checks for such duplicates and eliminates them.

```

*/
i = 0;
while ((i < NO_OF_REGIONS) && (!rg_infol[i].dup_index))
{
    j = i+1;
    while((j < NO_OF_REGIONS) && (!rg_infol[j].dup_index))

```

```

    {
        if ((rg_infol[i].x_centroid == rg_infol[j].x_centroid) &&
            (rg_infol[i].y_centroid == rg_infol[j].y_centroid))
        {
            dist1 = euc_dist_2(rg_infol[i].x_centroid, rg_infol[i].y_centroid,
                               rg_info2[i].x_centroid, rg_info2[i].y_centroid);
            dist2 = euc_dist_2(rg_infol[j].x_centroid, rg_infol[j].y_centroid,
                               rg_info2[j].x_centroid, rg_info2[j].y_centroid);
            if (dist1 > dist2)
            {
                rg_infol[i].dup_index = 1;
                rg_info2[i].dup_index = 1;
                j = NO_OF_REGIONS;
            }
            else
            {
                rg_infol[j].dup_index = 1;
                rg_info2[j].dup_index = 1;
            }
        }
        j++;
    }
    i++;
}

i = 0;
while ((i < NO_OF_REGIONS) && (!rg_info2[i].dup_index))
{
    j = i+1;
    while((j < NO_OF_REGIONS) && (!rg_info2[j].dup_index))
    {
        if ((rg_info2[i].x_centroid == rg_info2[j].x_centroid) &&
            (rg_info2[i].y_centroid == rg_info2[j].y_centroid))
        {
            dist1 = euc_dist_2(rg_info2[i].x_centroid, rg_info2[i].y_centroid,
                               rg_infol[i].x_centroid, rg_infol[i].y_centroid);
            dist2 = euc_dist_2(rg_info2[j].x_centroid, rg_info2[j].y_centroid,
                               rg_infol[j].x_centroid, rg_infol[j].y_centroid);
            if (dist1 > dist2)
            {
                rg_infol[i].dup_index = 1;
                rg_info2[i].dup_index = 1;
                j = NO_OF_REGIONS;
            }
            else
            {
                rg_infol[j].dup_index = 1;
                rg_info2[j].dup_index = 1;
            }
        }
        j++;
    }
    i++;
}
for(i=0; i < NO_OF_REGIONS; i++)
{
    if (!rg_infol[i].dup_index)
    {
        fprintf(out1, "%6.5f %6.5f %6.5f %6.5f\n",
            rg_infol[i].x_centroid, rg_infol[i].y_centroid,
            rg_info2[i].x_centroid, rg_info2[i].y_centroid);
    }
}
}

```

/* The function non_zero_8_neighbor is used to check for checking for a non-zero 8-neighbor of a clicked point. This function is used when a clicked point on a density slice is 0 (due to a mistake in cursor positioning in clicking). In that case, its 8 neighbors are checked for non-zero values, and the first non-zero neighbor encountered is used as a seed point for region aggregation.

The inputs to this program are the following:

image: Input density slice. The density slice is simply a binary image (created from the main input images).

i and j: Clicked x and y coordinates from UICP
 in and jn: Pointers to non-zero 8-neighbor of point (i,j) on the image.

```

*/
void non_zero_8_neighbor(image,i,j,in,jn)
int i,j,**image,*in,*jn;
{
    for(*in = i-1;*in <= i+1;*in++)
        for(*jn = j-1;*jn <= j+1;*jn++)
            if(image[*in][*jn] != 0)
                return;
    return;
}

```

/* Given a seed point (from UICP), this function performs region aggregation.

The inputs to this function are:

image: Contains density slice on which region aggregation has to be performed

i and j: X and Y coordinates of seed point

yes_array: Array to store pixels in the region, as the region is growing.

current_count: Pointer to the variable which will contain the count of
 the number of pixels in the region.

Since fragments can also be specified through UICP, the old value of
 current_count is stored so that pixels in fragments of a region can be
 stored contiguously in yes_array.

```

*/
void grow_region(image,i,j,yes_array,current_count)
int i,j,**image;
int **yes_array,*current_count;
{
    int count,k,l,**done_array;
    int count1,count2;
    int old_count;
    old_count = *current_count;

    done_array = (int **)calloc((size_t)IMG_SIZE,(size_t)sizeof(int *));
    if (!done_array)
    {
        printf("\nCouldnt allocate block of memory for done_array");
        exit(1);
    }
    for(count1=0;count1< IMG_SIZE;count1++)
    {
        done_array[count1] = (int *)calloc((size_t)IMG_SIZE,(size_t)sizeof(int));
        if (!done_array[count1])
        {
            printf("\nCouldnt allocate %d block of memory for done_array",count1);
            exit(1);
        }
    }
}

```

/* Since each density slice is binary, the region aggregation operation is simple and simply accumulates all pixels in yes_array, which are non_zero. At the same time, pixels which have already been accumulated are marked using the matrix done_array, so that they are not re-processed. The function uses an 8-neighbor criterion for region growing. If the number of pixels accumulated exceeds the specified size of yes_array (constant REGION_SIZE), then the function prints out an error message and the program quits.

```

*/
count = 0;
*current_count = count;
yes_array[count][0] = i;
yes_array[count][1] = j;
done_array[i][j] = 1;
do
{
    for(k=i-1;k<=i+1;k++)
    {
        for(l=j-1;l<=j+1;l++)

```

```

    {
        if(((k==i) && (l==j)) || (k >= IMG_SIZE) || ((l >= IMG_SIZE)) || (k < 0)
|| (l < 0))
            ;
        else
        {
            if (done_array[k][l] == 1)
                ;
            else
            {
                done_array[k][l] = 1;
                if (image[k][l] != 0)
                {
                    count++;
                    if (count >= REGION_SIZE)
                    {
                        printf("\nYes_array size not enough\n");
                        exit(1);
                    }
                    yes_array[count][0] = k;
                    yes_array[count][1] = l;
                }
            }
        }
    }
    (*current_count)++;
    i = yes_array[*current_count][0];
    j = yes_array[*current_count][1];
}
while(*current_count <= count);

for(count1=0;count1< IMG_SIZE;count1++)
    free(done_array[count1]);
free(done_array);
*current_count = *current_count + old_count;
}
/* This function calculates the centroid of a given region. The region is
represented by a list of the X and Y co-ordinates of the pixels
constituting the region. The region then calculates the centroid of the
region using standard formulae.

In this version of the function, it is assumed that all pixels in a
region are of the same intensity value, so that when calculating the
centroid, it is not required to explicitly multiply the value at the
pixel to the X and Y co-ordinate in the calculation. In case, this
function is to be used for centroid calculation for regions containing
varying pixel values, the actual value of the pixel has to also be
taken into account in the centroid calculation.

Inputs to the function are:

yes_array: List containing X and Y co-ordinates of pixels in the region
count: Number of pixels in the region
i and j: pointers to X and Y co-ordinates of centroid of the region (returned
by the function).

*/

void centroid(yes_array,count,i,j)
int **yes_array,count;
double *i,*j;
{
    int k;

    *i = 0.0;
    *j = 0.0;
    for(k=0;k<count;k++)
    {
        *i += (double)yes_array[k][0];
        *j += (double)yes_array[k][1];
    }
    *i = *i/(double)count;
    *j = *j/(double)count;
}

```



```

/*****

```

```

FILENAME: STARBYTE.C

```

```

WRITTEN BY: RADHIKA SIVARAMAKRISHNA

```

```

This program calculates the starbyte transformation of a given image.

```

```

THE INPUTS TO THE PROGRAM ARE:

```

1. NAME OF THE FILE CONTAINING THE GIVEN IMAGE (FOR WHICH WE NEED TO CALCULATE THE STARBYTE TRANSFORMATION). THE FILE FORMAT SHOULD BE RAW TEXT (NO HEADERS).
2. THE NAME OF THE FILE INTO WHICH THE STARBYTE-TRANSFORMED IMAGE WILL BE WRITTEN SHOULD BE SPECIFIED. THIS WILL BE WRITTEN AS A RAW TEXT FILE.
3. THE SIZE OF THE WINDOW OVER WHICH THE STARBYTE-TRANSFORM IS CALCULATED FOR EACH PIXEL. A LARGE WINDOW SIZE CREATES A MORE SMOOTH STARBYTE-IMAGE, WHILE TOO SMALL A SIZE RETURNS A NOISY OR GRAINY IMAGE. GOOD WINDOW SIZES ARE 9 OR 15. WINDOW SIZES SHOULD BE AN ODD MULTIPLE OF 3.
4. THE TYPE OF PATTERN CONFIGURATION. THERE ARE 8 CHOICES HERE. THESE ARE DESCRIBED IN THE THESIS UNDER THE CHAPTER "THE STARBYTE TRANSFORMATION". AGAIN GOOD CHOICES ARE 0 OR 2. THESE RETURN IMAGES WHICH HAVE 16 DISTINCT GRAY LEVELS, 0 TO 15. ESSENTIALLY CHOICES 0 TO 3 ARE 4-BIT PATTERNS (16 DISTINCT LEVELS), 4 AND 5 ARE 6-BIT (64 DISTINCT LEVELS) STARBYTE PATTERNS, WHILE 7 AND 8 ARE 8-BIT PATTERNS (256 LEVELS).
5. SIZE OF THE IMAGE: THE IMAGE IS ASSUMED TO BE SQUARE. TYPICAL SIZES ARE 128 X 128, OR 256 X 256. SINCE THE IMAGE IS ASSUMED TO BE SQUARE, IT IS ENOUGH IF THE NUMBER OF ROWS IS SPECIFIED. FOR EXAMPLE FOR A 128 X 128 IMAGE, IT IS ENOUGH TO SPECIFY "128" FOR THIS INPUT.

```

THIS PROGRAM CAN BE COMPILED ON ANY KIND OF SYSTEM, THOUGH HOWEVER INCLUDE FILES MAY HAVE TO BE CHANGED FOR SYSTEMS OTHER THAN UNIX.

```

```

*****/

```

```

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

```

```

main()

```

```

[
FILE *in,*out; /*input and output file pointers */
int **image; /* pointer for the input image */
int **pat_image; /*pointer to the final starbyte image */
char imfile[25],outfile[25]; /*variables to store input and output filenames*/
unsigned char fin_pattern,bit_pattern; /*variables to store intermediate

```

```

starbyte values */

```

```

/
float p[8],ptot; /* stores averages over different sectors */
int i,j,k,l,m,n,o;
int ws,wo,no_of_sectors;
float area_array[9]; /* stores averages over different
k x k blocks of a 3k x 3k neighborhood */

```

```

int pattern;
float mult_factor;
int DIM; /* size of the square image */

```

```

/* user prompts to read the 5 inputs to the program */

```

```

printf("\nGive name of file to read: ");
scanf("%s",imfile);
printf("\nGive name of file to write: ");
scanf("%s",outfile);
printf("\nGive size of window: ");
scanf("%d",&ws);
printf("\nGive type of pattern configuration: ");
scanf("%d",&pattern);

```



```

printf("\nGive size of image: (must be even and square) ");
scanf("%d",&DIM);

/* input and output files are opened */
in = fopen(infile,"r");
if (!in)
{
printf("\nCouldnt open input file\n");
exit(1);
}
out = fopen(outfile,"w");
if (!out)
{
printf("\nCouldnt open output file\n");
exit(1);
}

/* memory is allocated for the input image and the final starbyte image */
image = (int **)calloc((size_t)DIM,(size_t)sizeof(int *));
if (!image)
{
printf("\nCouldnt allocate block of memory for image");
exit(1);
}
for(i=0;i< DIM;i++)
{
image[i] = (int *)calloc((size_t)DIM,(size_t)sizeof(int));
if (!image[i])
{
printf("\nCouldnt allocate %d block of memory for image",i);
exit(1);
}
}
pat_image = (int **)calloc((size_t)DIM,(size_t)sizeof(int *));
if (!pat_image)
{
printf("\nCouldnt allocate block of memory for pat_image");
exit(1);
}
for(i=0;i< DIM;i++)
{
pat_image[i] = (int *)calloc((size_t)DIM,(size_t)sizeof(int));
if (!pat_image[i])
{
printf("\nCouldnt allocate %d block of memory for pat_image",i);
exit(1);
}
}

/* ws is the window size.
wo is used for indexing and is given by (ws -1)/2 */

wo = (ws - 1)/2;

/* the input file is read */
for (i=0;i < DIM;i++)
for(j=0;j < DIM;j++)
fscanf(in,"%d",&image[i][j]);

/*
Below is the loop to calculate the starbyte value at each pixel */ A
border of size wo is left on all sides of the starbyte image. The
neighborhood size is always an odd multiple of 3. Thus a 3k x 3k
neighborhood is converted to a 3 x 3 neighborhood by dividing the 3k x
3k neighborhood into a 3 x 3 block matrix, each block being of size k x
k. Then each block is replaced by its average grey level, thus reducing
the 3k x 3k neighborhood to a 3 x 3 neighborhood. The variable
"area_array" stores the average grey values for each of these block
matrices. "ptot" contains either the average grey level over the whole
neighborhood or the grey value of the pixel, depending upon the
protocol used. The loops below calculate the average grey value over
each k x k block of the 3k x 3k neighborhood.
*/

for(i=wo;i < DIM - wo;i++) {

```

```

for(j=wo; j < DIM - wo; j++) {
    for(o=0; o < 9; o++)
        area_array[o] = 0.0;
    for(o=0; o < 8; o++)
        p[o] = 0.0;
    fin_pattern = 0x00;
    ptot = 0.0;
    for(k = i - wo; k <= i + wo; k++) {
        for(l = j - wo; l <= j + wo; l++) {
            m = k - i + wo;
            n = l - j + wo;
            if(m < (ws/3)) {
                if(n < (ws/3))
                    area_array[0] += (float)image[k][l];
                if((n < (2*ws/3)) && (n >= (ws/3)))
                    area_array[1] += (float)image[k][l];
                if(n >= (2*ws/3))
                    area_array[2] += (float)image[k][l];
            }
            if((m < (2*ws/3)) && (m >= (ws/3))) {
                if(n < (ws/3))
                    area_array[3] += (float)image[k][l];
                if((n < (2*ws/3)) && (n >= (ws/3)))
                    area_array[4] += (float)image[k][l];
                if(n >= (2*ws/3))
                    area_array[5] += (float)image[k][l];
            }
            if(m >= (2*ws/3)) {
                if(n < (ws/3))
                    area_array[6] += (float)image[k][l];
                if((n < (2*ws/3)) && (n >= (ws/3)))
                    area_array[7] += (float)image[k][l];
                if(n >= (2*ws/3))
                    area_array[8] += (float)image[k][l];
            }
        }
    }
}

```

/* After "area_array" has been filled, the program calculates the starbyte value based on one of 8 protocols. These 8 protocols has been described in the thesis in the chapter "The starbyte transformation". The user is required to specify a number between 1 and 8 to select the protocol to use. If the user selects a number outside this range, the program uses the 1st protocol by default. The calculations below consist of calculating the average grey levels over different sectors in the neighborhood. These averages are stored in the array "p", with "ptot" containing either the average over the whole neighborhood or the grey level at the pixel under consideration, depending upon the protocol used.

```

*/
switch(pattern) {
    case 1:
        no_of_sectors = 4;
        mult_factor = (9.0/(float)(4*ws*ws));
        p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[3]
+ area_array[4]);
        p[1] = mult_factor*(area_array[1] + area_array[2] + area_array[4]
+ area_array[5]);
        p[2] = mult_factor*(area_array[3] + area_array[4] + area_array[6]
+ area_array[7]);
        p[3] = mult_factor*(area_array[4] + area_array[5] + area_array[7]
+ area_array[8]);
        ptot = mult_factor*(4.0/9.0)*(area_array[0] + area_array[1] + are
a_array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
        break;
    case 2:
        no_of_sectors = 4;
        mult_factor = (9.0/(float)(2*ws*ws));

```

```

        p[0] = mult_factor*(area_array[1] + area_array[2]);
        p[1] = mult_factor*(area_array[5] + area_array[8]);
        p[2] = mult_factor*(area_array[6] + area_array[7]);
        p[3] = mult_factor*(area_array[0] + area_array[3]);
        ptot = mult_factor*2.0*area_array[4];
        break;

    case 3:
        no_of_sectors = 4;
        mult_factor = (9.0/(float)(4*ws*ws));
        p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[2]
+ area_array[4]);
        p[1] = mult_factor*(area_array[2] + area_array[4] + area_array[5]
+ area_array[8]);
        p[2] = mult_factor*(area_array[4] + area_array[6] + area_array[7]
+ area_array[8]);
        p[3] = mult_factor*(area_array[0] + area_array[3] + area_array[6]
+ area_array[4]);
        ptot = mult_factor*(4.0/9.0)*(area_array[0] + area_array[1] + are
a_array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
        break;

    case 4:
        no_of_sectors = 4;
        mult_factor = (9.0/(float)(3*ws*ws));
        p[0] = mult_factor*(area_array[0] + area_array[4] + area_array[8]
);
        p[1] = mult_factor*(area_array[2] + area_array[4] + area_array[6]
);
        p[2] = mult_factor*(area_array[1] + area_array[4] + area_array[7]
);
        p[3] = mult_factor*(area_array[3] + area_array[4] + area_array[5]
);
        ptot = mult_factor*3*(area_array[4]);
        break;

    case 5:
        no_of_sectors = 6;
        mult_factor = (9.0/(float)(3*ws*ws));
        p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[2]
);
        p[1] = mult_factor*(area_array[2] + area_array[5] + area_array[8]
);
        p[2] = mult_factor*(area_array[6] + area_array[7] + area_array[8]
);
        p[3] = mult_factor*(area_array[0] + area_array[3] + area_array[6]
);
        p[4] = mult_factor*(area_array[0] + area_array[4] + area_array[8]
);
        p[5] = mult_factor*(area_array[2] + area_array[4] + area_array[6]
);
        ptot = mult_factor*(3.0/9.0)*(area_array[0] + area_array[1] + are
a_array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
        break;

    case 6:
        no_of_sectors = 6;
        mult_factor = (9.0/(float)(3*ws*ws));
        p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[2]
);
        p[1] = mult_factor*(area_array[3] + area_array[4] + area_array[5]
);
        p[2] = mult_factor*(area_array[6] + area_array[7] + area_array[8]
);
        p[3] = mult_factor*(area_array[0] + area_array[3] + area_array[6]
);
        p[4] = mult_factor*(area_array[1] + area_array[4] + area_array[7]
);
        p[5] = mult_factor*(area_array[2] + area_array[5] + area_array[8]
);
        ptot = mult_factor*(3.0/9.0)*(area_array[0] + area_array[1] + are
a_array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);

```

```

        break;

        case 7:
            no_of_sectors = 8;
            mult_factor = (9.0/(float)(ws*ws));
            p[0] = mult_factor*(area_array[0]);
            p[1] = mult_factor*(area_array[1]);
            p[2] = mult_factor*(area_array[2]);
            p[3] = mult_factor*(area_array[3]);
            p[4] = mult_factor*(area_array[5]);
            p[5] = mult_factor*(area_array[6]);
            p[6] = mult_factor*(area_array[7]);
            p[7] = mult_factor*(area_array[8]);
            ptot = mult_factor*(area_array[4]);
            break;

        case 8:
            no_of_sectors = 8;
            mult_factor = (9.0/(float)(4*ws*ws));
            p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[3]
+ area_array[4]);
            p[1] = mult_factor*(area_array[1] + area_array[2] + area_array[4]
+ area_array[5]);
            p[2] = mult_factor*(area_array[3] + area_array[4] + area_array[6]
+ area_array[7]);
            p[3] = mult_factor*(area_array[4] + area_array[5] + area_array[7]
+ area_array[4]);
            p[4] = mult_factor*(area_array[0] + area_array[1] + area_array[2]
+ area_array[8]);
            p[5] = mult_factor*(area_array[2] + area_array[4] + area_array[5]
+ area_array[8]);
            p[6] = mult_factor*(area_array[4] + area_array[6] + area_array[7]
+ area_array[8]);
            p[7] = mult_factor*(area_array[0] + area_array[3] + area_array[6]
+ area_array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
            ptot = mult_factor*(4.0/9.0)*(area_array[0] + area_array[1] + are
array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
            break;

        default:
            no_of_sectors = 4;
            mult_factor = (9.0/(float)(4*ws*ws));
            p[0] = mult_factor*(area_array[0] + area_array[1] + area_array[3]
+ area_array[4]);
            p[1] = mult_factor*(area_array[1] + area_array[2] + area_array[4]
+ area_array[5]);
            p[2] = mult_factor*(area_array[3] + area_array[4] + area_array[6]
+ area_array[7]);
            p[3] = mult_factor*(area_array[4] + area_array[5] + area_array[7]
+ area_array[8]);
            ptot = mult_factor*(4.0/9.0)*(area_array[0] + area_array[1] + are
array[2] + area_array[3] + area_array[4] + area_array[5] + area_array[6] + area_array[7] + area
array[8]);
            break;
    }

    /* Below is the actual binary comparison between the values in the array
    "p" to "ptot". The variable "fin_pattern" contains the starbyte.
    */
    for(o=0;o < 8;o++) {
        if(p[o] > ptot)
            bit_pattern = 0x01 << o;
        else
            bit_pattern = 0x00;
        fin_pattern = fin_pattern | bit_pattern;
    }

    /* The starbyte value at the pixel is put into its correct
    location in the starbyte image */
    pat_image[i][j] = (unsigned int)fin_pattern;

```

```
    }  
}  
/* The starbyte image is written into the specified file. */  
for (i=0; i < DIM; i++) {  
    for(j=0; j < DIM; j++)  
        fprintf(out, "%d ", pat_image[i][j]);  
    fprintf(out, "\n");  
}
```

```
%FILENAME: ADDIMAGES.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
% This is a simple Matlab program to load density slices numbered from 0 to
% 15 (we assume a 4-bit protocol)
% and to add these slices up. These slices are then written as a text file
% (with no headers) "sbtot.asc", using a Matlab program "textwrite.m".
% sbttot.asc will then be moved into the
% original ST file using the Unix command "mv", called from the shell program
% "totset". For textwrite.m to be used successfully, the directory containing
% these files has to be included in the path of Matlab, using the path command
% as given below. In this case the file "textwrite.m" is present in the
% directory "/home/radhika/bin".
% The number 15 has to be changed appropriately for a different sized protocol.
```

```
path(path, '/home/radhika/bin')
for i=0:15,
    eval(['load dsp92', num2str(i), '.asc']);
end
sz = size(dsp920);
sz = sz(1);
sbtot = zeros(sz);
for i=0:15,
    out = eval(['dsp92', num2str(i)]);
    sbttot = sbttot + out;
end
textwrite(sbtot, 'sbtot.asc', sz)
```

%FILENAME: AUTO4.M

%WRITTEN BY: RADHIKA SIVARAMAKRISHNA

%This function takes in the 4-column lists corresponding to the regions
 %in every density slice of the target and reference STs and obtains the
 %list of corresponding points for each slice by calling the function
 %test4 for every density value. Here autol and auto2 contain the list of
 %4 parameters for every region in every density slice, while counttot is
 %a 2-column vector which contains the number of regions in every density
 %slice for the target and reference ST. c1 and c2 are the constants T1
 %and T2 described in the 5th chapter of the thesis while c3 and c4
 %define the boundaries outside which matches are not considered. The
 %function outputs ef, the lowest error, ed, the difference between the
 %lowest and second lowest error, final, the list of matches selected,
 %numarray, the size of the regions in the target and reference STs which
 %have been matched up, and ct, the number of matches in each density
 %slice. ef, ed, numarray and ct have been output only for diagnostics
 %and can be ignored for actual image registration.
 %Usage: [ef,ed,final,numarray,ct] = auto4(autol,auto2,counttot,c1,c2,c3,c4)

```
function [ef,ed,final,numarray,ct] = ...
    auto4(autolold,auto2old,counttotold,c1,c2,c3,c4)
counttot = counttotold;
autol = autolold;
auto2 = auto2old;
counttot = cumsum(counttot);
bestmatches = [];
mat1 = autol(1:counttot(1,1),:);
mat2 = auto2(1:counttot(1,2),:);
szmat1 = size(mat1);
szmat2 = size(mat2);
ct = [];
ef = [];
ed = [];
cttemp = [];
bestmatchestemp = [];
eftemp = [];
edtemp = [];
[cttemp,bestmatchestemp,eftemp,edtemp] = test4(mat1,mat2,c1,c2,c3,c4);
bestmatches = [bestmatches;bestmatchestemp];
if (~isempty(cttemp))
    ct = [ct;cttemp 1];
end
if (~isempty(eftemp))
    ef = [ef;eftemp];
end
if (~isempty(edtemp))
    ed = [ed;edtemp];
end

for i=2:15,
    mat1 = autol(counttot(i-1,1) + 1:counttot(i,1),:);
    mat2 = auto2(counttot(i-1,2) + 1:counttot(i,2),:);
    szmat1 = size(mat1);
    szmat2 = size(mat2);
    cttemp = [];
    bestmatchestemp = [];
    eftemp = [];
    edtemp = [];
    [cttemp,bestmatchestemp,eftemp,edtemp] = test4(mat1,mat2,c1,c2,c3,c4);
    bestmatches = [bestmatches;bestmatchestemp];
    if (~isempty(cttemp))
        ct = [ct;cttemp 1];
    end
    if (~isempty(eftemp))
        ef = [ef;eftemp];
    end
    if (~isempty(edtemp))
        ed = [ed;edtemp];
    end
end
if ~isempty(bestmatches),
    numarray = bestmatches(:,3:6);
    bestmatches = bestmatches(:,[1 2 4 5]);
    final = [bestmatches(:,1:2) bestmatches(:,3:4)];
end
```

end


```
%FILENAME: BESTSECOND4.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%This function is called by the function test4. It takes in a 4-column
%matrix a, corresponding to all regions in a density slice of the target
%ST. For every region in the reference slice it calculates an error
%against every region in the corresponding density slice of the target
%ST using the function "minmax4". The list b is a 4-column matrix
%representing every region in the corresponding density slice of the
%reference ST. weights is a 4-element vector which can be used to weight
%each of the 4 elements to decide their degree of importance in the
%error calculation. r, the output of the function, is a 5-column matrix,
%with the same number of rows as the number of regions in the density
%slice of the reference ST. The first 2 columns of r give the indices
%in the target slice of the best match and the second best match for
%every region in the reference slice, while the 3rd column gives the
%index in the reference slice. Hence the 3rd column merely runs from 1
%to the number of regions in the reference density slice. The last two
%columns give the errors corresponding to the best and second best match.
%Usage: r = bestsecond4(a,b,weights)
```

```
function r = bestsecond4(a,b,weights)
szw = size(weights);
ni = szw(2);
sza = size(a);
szb = size(b);
na = sza(1);
nb = szb(1);
if na == 1,
    r = zeros(nb,3);
else
    r = zeros(nb,5);
end
for i = 1:nb,
    bmat = ones(na,ni)*diag(b(i,:));
    e = minmax4(a',bmat',weights);
    if na == 1,
        r(i,:) = [1 i e];
    else
        [en,j] = sort(e);
        r(i,:) = [j(1:2) i en(1:2)];
    end
end
end
```

```
%FILENAME: BILINEAR.M  
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
% This function can be called from within Matlab.  
% Given the non-integer X and Y location in an image, whose  
% grey values are available only at integral X and Y locations, this function  
% performs bilinear interpolation to estimate the grey value at the  
% non-integral location. The inputs to this function are x and y, the  
% non-integer locations, as well as the given image, mat.  
% Usage: r = bilinear(x,y,mat)
```

```
function r = bilinear(x,y,mat)  
DIM = max(size(mat));  
if ((x < 1) | (y < 1) | (x > (DIM - 2)) | (y > (DIM - 2)))  
    r = 0;  
else  
    tsi = floor(x);  
    eta = floor(y);  
    a = x - tsi;  
    b = y - eta;  
    r = (1 - a)*(1-b)*mat(tsi+1,eta+1) + a*(1 - b)*mat(tsi+2,eta+1) + b*(1-a)*mat(tsi+1,eta+2)  
    + a*b*mat(tsi+2,eta+2);  
end
```

```
%FILENAME: HELP1STR.M
```

```
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%This file when executed from within the Matlab environment, will open up  
%a help dialog box which will provide instructions to use UICP.
```

```
b = blanks(10);
str00 = str2mat(['UICP'],...
['A USER-INTERACTIVE PROGRAM'],...
['USED IN CONJUNCTION WITH'],...
['THE STARBYTE TRANSFORMATION'],...
['TO EXTRACT CONTROL POINTS FOR'],...
['IMAGE REGISTRATION'],...
[''],...
['WRITTEN BY RADHIKA SIVARAMAKRISHNA'],...
['DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING'],...
['UNIVERSITY OF MANITOBA, WINNIPEG, MANITOBA'],...
['CANADA R3T 5V6'],...
['SEPTEMBER 1997'],...
['']);
str11 = strcat(['This is a user-interactive program which can be used to extract control points f  
or matching'],...
[' a pair of given images. We assume that the starbyte transforms (STs) of the given pair of'],...
[' original images have already been created and "cleaned up" with a binary open-and-close'],...
[' operation. This program allows the user to loads the STs and specify the number of density'],...
[' levels. The program then density slices each ST at every possible density level. The user'],...
[' is then allowed to separate concatenated regions in each slice. If a particular slice is'],...
[' changed because of this, the original version of the slice is saved into a file. The user'],...
[' is then allowed to click on matching regions in the corresponding slices. The clicked'],...
[' points are saved into a file. Region aggregation and control point extraction (not part of'],...
[' this program), will be done using these clicked points as seed points. ']);
helpmat1 = str2mat(str00,str11);
clear b str00 str11;
f1 = helpdlg(helpmat1,'Startup Message');
set(f1,'Position',[50,100,650,500],'Name','Startup Message','Color',[1 1 1]);
```

%FILENAME: HELP2STR.M

%WRITTEN BY: RADHIKA SIVARAMAKRISHNA

%This file when executed from within the Matlab environment, will open up
%a help dialog box which displays a Startup Message for UICP.

```
str1 = strcat('When the program starts, a dialog box will open up and',...  
' you will be asked to type',...  
' in the name of the first ST to load. After this is loaded',...  
' you will be asked to',...  
' enter the name of the second ST to load. You will',...  
' also be asked to enter the',...  
' name of the file into which clicked points will be saved.',...  
' You will also have to',...  
' enter the name of a "count file" which contains the number',...  
' of clicked regions in',...  
' each slice. The program will then ask you to enter the number',...  
' of density levels',...  
' through a dialog box. Press "Done" after you enter the number.',...  
' The density',...  
' levels considered will be from 1 to the maximum',...  
' number specified, because the',...  
' density level "0" will not be considered. The program',...  
' will density slice the');
```

```
str2 = strcat('images and display corresponding slices side by side.',...  
' After this you can edit the images to separate',...  
' concatenated regions or slice out',...  
' unwanted portions of a region. You will be allowed to edit',...  
' the left image first',...  
' and then the right image. The original "arrow" symbol on',...  
' the figure window will',...  
' change to a "crosshair" symbol whenever you are allowed to',...  
' click on the figure',...  
' window, either for separating concatenated',...  
' regions or for selecting matching',...  
' regions. For separating concatenated regions, you have',...  
' to click on a starting',...  
' point and an ending point to define a line of separation',...  
' along which the regions',...  
' will be separated. This can be done as many times as',...  
' necessary to separate all');
```

```
str3 = strcat('concatenated regions. When you have',...  
' finished separating lines, the original',...  
' version of the slice will be saved into a file',...  
' with the slice number as part of',...  
' its name. It will have a suffix "old.asc". If it',...  
' is a slice on the left side, it',...  
' will have the prefix "slw". For slices on the',...  
' right side the prefix "sl" will be',...  
' used. The new version of the same slice replaces',...  
' the old version in the ST. At',...  
' the end, after regions have been selected from',...  
' all slices, the STs will be saved',...  
' back in their original files from where',...  
' they were loaded.',...  
' After concatenated regions have been',...  
' separated, you can start selecting');
```

```
str4 = strcat(...  
' corresponding regions in the two STs, one',...  
' at a time (first the left side, then',...  
' the right side). This is done by clicking on',...  
' any one point using either the',...  
' first or second button of the mouse in a region',...  
' in the first image and then',...  
' pressing return. Then click on the corresponding',...  
' region in the other image and',...  
' press return once again. Where you click in each',...  
' region is not important, as',...  
' long as the crosshair is inside the region.',...  
' If a region is fragmented, click on',...  
'
```

```
' a point in each of the fragments and then',...
' press return (after all fragments have',...
' been clicked on).If you wish to continue',...
' selecting regions, repeat the',...
' procedure. Otherwise press the 3rd (right)',...
' button of the mouse when the program',...
' allows you to click on the lefthand side image.',...
' This will signal the program to');

str5 = strcat(...
' display the next pair of density slices.',...
' This procedure will be repeated for all',...
' density slices. If there are no matching',...
' regions in a pair of slices, the 3rd or',...
' right button of the mouse can be clicked',...
' right away so that the program can',...
' display the next pair. After the process',...
' is repeated for all slices, the program',...
' will display the total number of selected',...
' regions and quit, after saving the',...
' modified STs, the "count file" and the clicked points. ');
helpmat2 = strcat(str1,str2,str3,str4,str5);
clear str1 str2 str3 str4 str5
f2 = helpdlg(helpmat2,'Instructions');
set(f2,'Position',[80,60,550,600],'Name','Instructions','Color',[1 1 1]);
```

```
%FILENAME: MINMAX4.M
```

```
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%This function calculates the sum of the absolute error between the  
%4-parameter vector corresponding to a region in the reference image to  
%the 4-parameter vector corresponding to a region in the target image. "w"  
%is a 4-element vector which can be used to weight the elements of the  
%error before the final summation. a and b are the input  
%4-parameter vectors, while w is the weight vector.  
%Usage: e = minmax4(a,b,w);
```

```
function e = minmax4(a,b,w);  
dim = size(a);  
r = dim(1);  
c = dim(2);  
wmat = [w(1)*ones(1,c);w(2)*ones(1,c);w(3)*ones(1,c);w(4)*ones(1,c)];  
newmat = abs(a - b);  
e = sum(newmat.*wmat);
```

```
%FILENAME: RSQREVAL.M
```

```
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
% Usage: out = rsqreval(x,y,xvect,yvect)
```

```
% This function has merely been written to simplify the calculation of  
% each (rlogr)^2 calculation in the TPS equation. x and y are the X and Y  
% co-ordinates of the control points in the target image, while xvect and  
% yvect contain the list of X and Y co-ordinates of the corresponding  
% points in the reference image. The function calculates the sum of the  
% (rlogr)^2 terms in the TPS equation.
```

```
function out = rsqreval(x,y,xvect,yvect);
```

```
out = (x - xvect).^2 + (y - yvect).^2;
```

```
out = out.*(log(-out + out));
```

```
%FILENAME: SPLCOEFF.M
```

```
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%Given two images A and B, where B has been created from A using a thin  
%plate spline whose coefficients are given by "coeff", this function takes  
%in "xy", 2-column list of co-ordinates in B which were used to obtain  
%"coeff", as well as the present 2-column list of co-ordinates in B  
%"ij", which have to be mapped back to co-ordinates in A to give the  
%output "xhatyhat". It uses the function rsqreval to compute the  
%(rlogr)^2 terms in the TPS equation.  
%Usage: xhatyhat = splcoeff(ij,xy,coeff)
```

```
function xhatyhat = splcoeff(ij,xy,coeff);  
szxy = size(xy);  
n = szxy(1);  
szij = size(ij);  
m = szij(1);  
for ct = 1:m,  
    i = ij(ct,1);  
    j = ij(ct,2);  
    xhat(ct) = i + coeff(1,1) + coeff(2,1)*i + coeff(3,1)*j +...  
        0.5*sum(coeff(4:n+3,1).*rsqreval(i,j,xy(:,1),xy(:,2)));  
    yhat(ct) = j + coeff(1,2) + coeff(2,2)*i + coeff(3,2)*j +...  
        0.5*sum(coeff(4:n+3,2).*rsqreval(i,j,xy(:,1),xy(:,2)));  
end  
xhatyhat = [xhat' yhat'];
```



```
%FILENAME: SPLINEWPIMG.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
% This function calculates the unwarped image using a thin plate spline
% (TPS) and the reference image. This function has to be called from
% within Matlab after loading in the reference image, and the
% list of control points output by the C program manualregion.c.
% The unwarped image is returned in the
% variable "f" while the coefficients of the TPS are returned in the
% variable "coeff". The image "f" can be saved in a suitable file using
% the function textwrite.m from within Matlab.
```

```
% The inputs to this function are:
```

```
% (1) img, the reference image and (2) bestmatches, the 4-column list of
% control points, with the co-ordinates of the target image specified in
% the first two columns followed by the co-ordinates of the reference
% image. This function calls the function tpscoeff.m which calculates
% the co-efficients of the thin plate spline. It then calls the function
% rsqreval, which has been written to evaluate the (rlogr)^2 terms in the
% TPS equation. Finally, the grey value at each pixel in the unwarped
% image is obtained from the grey values in the reference image using
% bilinear interpolation.
```

```
%Usage: [f,coeff] = splinewpimg(img,bestmatches)
```

```
function [f,coeff] = splinewpimg(img,bestmatches);
coeff = tpscoeff(bestmatches);
xy = bestmatches(:,1:2);
isz = max(size(img));
f = zeros(isz);
n = max(size(xy));
for i=0:isz-1,
    for j=0:isz-1,
        xhat = i + coeff(1,1) + coeff(2,1)*i + coeff(3,1)*j + ...
0.5*sum(coeff(4:n+3,1).*rsqreval(i,j,xy(:,1),xy(:,2)));
        yhat = j + coeff(1,2) + coeff(2,2)*i + coeff(3,2)*j + ...
0.5*sum(coeff(4:n+3,2).*rsqreval(i,j,xy(:,1),xy(:,2)));
        f(i+1,j+1) = bilinear(xhat,yhat,img);
    end
end
f = round(f);
```

```
%FILENAME: TEST4.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%This function performs the actual matching of regions in a density
%slice of the target ST to regions in a density slice of the reference
%ST. mat1 and mat2 contain the 4-column list of parameters representing
%the regions in the target and reference density slice. c11 and c22 are
%the same as parameters T1 and T2 described in chapter 5 of the thesis,
%while c33 and c44, represent boundaries, outside which no matches are
%considered.
```

```
%Initially the target list is normalized column-wise so that every
%column varies from 0 to 1. Then the reference list is normalized
%column-wise using the same normalization constants as in the target
%slice. After this a region corresponding to the reference list is
%compared with every region in the target list. This is done through the
%function bestsecond4, which in turn calls the function minmax4. The
%function bestsecond4 returns the variable requal, which contains the
%index number on the target slice corresponding to the lowest and second
%lowest error. The values of the error are also returned. If the lowest
%error is greater than c11, the match is rejected, or if the ratio of
%the second lowest error to the lowest is greater than c22, the match is
%rejected. This procedure is repeated for all the regions in the
%reference image. Finally all matches outside the boundary defined by
%c33 and c44 are rejected. The outputs of this program are (1) bm, the
%list of matches with the co-ordinates in the target slice given in the
%first two columns, and the co-ordinates in the reference slice given in
%the last two columns (2) ct, the number of matches (3) ef, the list of
%errors corresponding to the best matches in each cases (4) ed, the list
%of differences in value between the second lowest and the lowest
%error. ct, ef, and ed are just output for diagnostics and are not really
%necessary for image registration.
```

```
%Usage: [ct,bm,ef,ed] = test4(mat1,mat2,c11,c22,c33,c44)
```

```
function [ct,bm,ef,ed] = test4(mat1,mat2,c11,c22,c33,c44)
n = 4;
szmat1 = size(mat1);
szmat2 = size(mat2);
c1 = szmat1(1);
c2 = szmat2(1);
if ((c1 == 0) | (c2 == 0)),
    return
end
c = max(c1,c2);
minmat1 = ones(c,n)*diag(min(mat1));
maxmat1 = ones(c,n)*diag(max(mat1));
mat1new = (mat1 - minmat1(1:c1,:))./(maxmat1(1:c1,:) - minmat1(1:c1,:));
mat2new = (mat2 - minmat1(1:c2,:))./(maxmat1(1:c2,:) - minmat1(1:c2,:));
weightequal = [0.25 0.25 0.25 0.25];
requal = bestsecond4(mat1new,mat2new,weightequal);
if isempty(requal),
    return
end
szrequal = size(requal);
szrequal = szrequal(2);
if szrequal == 3,
    errmat = requal(:,3);
    e = errmat < c11;
    bestmatches = requal(:,[1 2]);
else
    errmat = requal(:,4);
    errdiffmat = requal(:,5) - requal(:,4);
    errnewmat = requal(:,5)./requal(:,4);
    e = (errmat < c11).*(errnewmat >= c22);
    bestmatches = requal(:,[1 3]);
end
i = find(e == 1);
efirst = errmat(i,:);
if szrequal == 3,
    ediff = zeros(size(efirst));
else
    ediff = errdiffmat(i,:);
```

```

end
bestmatches = bestmatches(1,1:2);

if isempty(bestmatches),
    return
end

%The code immediately below checks for duplicates in bestmatches and
%removes them.

szbstm = size(bestmatches);
b = szbstm(1);
bestmatches = [bestmatches zeros(b,1)];
for i = 1:b,
    for j = i+1:b,
        if ((bestmatches(i,1) == bestmatches(j,1)) | (bestmatches(i,2) == ...
            bestmatches(j,2))),
            bestmatches(i,3) = 1;
            bestmatches(j,3) = 1;
        end
    end
end

if isempty(bestmatches),
    return
end
i = find(bestmatches(:,3) == 0);
if isempty(i),
    return
end

bestmatches = bestmatches(1,1:2);
bmtemp = [mat1(bestmatches(:,1),[1 2 4]),mat2(bestmatches(:,2),[1 2 4])];
efirst = efirst(i,:);
ediff = ediff(i,:);
i = find(sum((bmtemp(:,[1 2 4 5]) >= c33) &...
    (bmtemp(:,[1 2 4 5]) <= c44))'') == 4);
if ~isempty(i),
    bm = bmtemp(i,:);
    ef = efirst(i,:);
    ed = ediff(i,:);
    ct = size(bm);
    ct = ct(1);
end

```

```
%FILENAME: TEXTWRITE.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA

% This matlab function allows the user to write out a matrix (containing
% integer values) in text format so that it can be read by a C program.
% x is the matrix to be written, s is the file to be written into (specified
% within quotes), n is the number of rows of the matrix to be written.
% Usage: textwrite(x,s,n)

function textwrite(x,s,n)
fid = fopen(s,'w');
for i=1:n,
    count = fprintf(fid,'%d ',x(i,:));
    count = fprintf(fid,'\n');
end
```

```
%FILENAME: TPSCOEFF.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
% The function below calculates the co-efficients of a thin plate spline
% used as a mapping function for image registration. The input "vect"
% is a 4-column list of control points with the first 2 columns representing
% the X and Y co-ordinates of the control points in the target image, with
% the last two columns representing the X and Y co-ordinates of
% the corresponding points in the reference image.
% The calculations below are based on the paper "Warping digital
% images using thin plate splines" by I. Barrodale et al,
% Pattern Recognition, Vol. 26, No. 2, pp. 375-376, 1993.
% We have adopted below the translation and scaling strategy prescribed in
% the paper.
% Usage: out = tpscoeff(vect)
```

```
function out = tpscoeff(vect)
zi2 = vect(:,4) - vect(:,2);
zilmat = [zeros(3,1);zi1];
zi2mat = [zeros(3,1);zi2];
xy = vect(:, [1 2]);
c = max((max(xy(:,1)) - min(xy(:,1))), (max(xy(:,2)) - min(xy(:,2))));
mnxi = min(xy(:,1));
mnyi = min(xy(:,2));
xy(:,1) = (xy(:,1) - mnxi)/c;
xy(:,2) = (xy(:,2) - mnyi)/c;
[x1,x2] = meshgrid(xy(:,1),xy(:,1));
%x2 = flipud(x2);
[y1,y2] = meshgrid(xy(:,2),xy(:,2));
%y2 = flipud(y2);
zmat = zeros(3);
sz = size(vect);
n = sz(1);
amat = [ones(n,1) xy];
rmat = rsqrfun(x1,x2,y1,y2);
finmat = [zmat amat';amat rmat];
finmat = pinv(finmat);
out1 = finmat*zilmat;
out2 = finmat*zi2mat;
out1(4:n+3) = out1(4:n+3)*2;
out2(4:n+3) = out2(4:n+3)*2;
out1(1) = out1(1) - out1(2)*(mnxi/c) - out1(3)*(mnyi/c) - (log(c))*...
sum(out1(4:n+3).*(xy(:,1).^2 + xy(:,2).^2));
out1(2) = out1(2)/c;
out1(3) = out1(3)/c;
out1(4:n+3) = out1(4:n+3)/(c.^2);
out2(1) = out2(1) - out2(2)*(mnxi/c) - out2(3)*(mnyi/c) - (log(c))*...
sum(out2(4:n+3).*(xy(:,1).^2 + xy(:,2).^2));
out2(2) = out2(2)/c;
out2(3) = out2(3)/c;
out2(4:n+3) = out2(4:n+3)/(c.^2);
out = [out1 out2];
```

```
%FILENAME: UICP.M
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
% This is the manual program used to extract control points from the
% starbyte-transformed images. It has been written using Matlab
% Version 4.2c (The Mathworks Inc., Natick, Massachusetts) on a Unix platform.
% Extensive instructions to use this program are given in a Startup Message
% and Instruction help dialog box, which have been captured and printed along
% with the code contained in this file.
```

```
help1str
w = waitforbuttonpress;
pause(2)
help2str
w = waitforbuttonpress;
pause(2)
```

```
% n represents the maximum number of slices considered, ie. 0 to maximum.
% Currently it has been set to 15, because all the protocols used in the
% thesis are 4-bit protocols.
n = 15;
```

```
% The reference and target "cleaned-up" STs are read and put into variables
% "outtot1" and "outtot2".
```

```
[infile1,pathnmin1] = uigetfile('*.asc','Input file 1');
eval(['load ',[pathnmin1,infile1]])
s1 = strrep(infile1,'.asc','');
[infile2,pathnmin2] = uigetfile('*.asc','Input file 2');
eval(['load ',[pathnmin2,infile2]])
s2 = strrep(infile2,'.asc','');
```

```
outtot1 = eval(s1);
outtot2 = eval(s2);
whitebg('w')
```

```
% Below is the loop to allow the user to define lines of separation
% between concatenated regions in the images.
% To do this the user clicks on the start and end
% of the line along which concatenated regions will be separated.
```

```
fg1 = figure(1);
set(fg1,'Position',[10 350 500 500]);
set(fg1,'NumberTitle','off')
fg2 = figure(2);
set(fg2,'Position',[520 350 500 500]);
set(fg2,'NumberTitle','off')
```

```
% Below is where the user can specify the file to write the
% corresponding clicked points into. These clicked points will form
% the seed points for region-aggregation.
```

```
[outfile,pathnmout] = uiputfile('','Save All Corresponding points into');
fid = fopen([pathnmout,outfile],'w');
[outotfile,pathnmct] = uiputfile('','Count Array saved to');
```

```
% Reference and target density slice pairs 1 to 15 are displayed side by
% with the slice corresponding to the target on the right and that of the
% reference on the left.
```

```
count = zeros(1,n);
for i=1:n,
    out1 = 255*(outtot1 == i);
    out2 = 255*(outtot2 == i);
    set(fg1,'Name','SEPARATING REGIONS')
    set(fg2,'Name','SEPARATING REGIONS')
    figure(1)
    colormap(gray(256))
    image(out1);
    whitebg('w');
    figure(2);
    colormap(gray(256))
    image(out2);
    whitebg('w');
```

```

helpdlg('SEPARATE REGIONS','Message');
w = waitforbuttonpress;
pause(2)
figure(1);

```

```

% The user is allowed to click anywhere on the screen
% in the target image slice, through a call to
% the function "ginput", with one of the
% mouse buttons. If the user selects the 3rd mouse button and presses
% RETURN, then the program moves over to the reference image slice, and
% allows the user to select a point on the screen with "ginput".
% If the user presses the 3rd mouse button again with a RETURN, then
% the program moves on to the next stage.
% If the 3rd mouse button is not pressed, then the selected point forms
% one end of a line which will be used to separate two concatenated regions..
% The user then has to select a second point on the screen.
% The user can separate as many concatenated regions as necessary for the
% target and the reference image. When a slice has been modified to a region
% separation operation, the original slice is saved into a file which has
% the slice number and the word "old" appended to it. If it is a target slice,
% the letter "w" is also added to the file name.

```

```

k = 1;
sz11 = size(out1);
iter = 0;
while k ~= 3,
    eval(['yw',num2str(i),num2str(iter),'xw',num2str(i),...
          num2str(iter),'button'] = ginput(2);'])
    x = eval(['xw',num2str(i),num2str(iter)]);
    y = eval(['yw',num2str(i),num2str(iter)]);
    if button == 3,
        if iter == 0,
            eval(['outw1old',num2str(i),' = (out1/255)*i;']);
        end
        out1 = del3line(out1,x,y);
        image(out1);
        iter = iter + 1;
    end
    k = button;
end

if (iter > 0)
    eval(['textwrite(outw1old,num2str(i),'','slw',num2str(i),...
    'old.asc',sz11(1))']);
    outtot1 = outtot1 - (outtot1 == i)*i + (out1/255)*i;
end
eval(['clear xw',num2str(i),num2str(iter)])
eval(['clear yw',num2str(i),num2str(iter)])

figure(2);

k = 1;
sz12 = size(out2);
iter = 0;
while k ~= 3,
    eval(['y',num2str(i),num2str(iter),'x',num2str(i),...
          num2str(iter),'button'] = ginput(2);'])
    x = eval(['x',num2str(i),num2str(iter)]);
    y = eval(['y',num2str(i),num2str(iter)]);
    if button == 3,
        if iter == 0,
            eval(['outlold',num2str(i),' = (out2/255)*i;']);
        end
        out2 = del3line(out2,x,y);
        image(out2);
        iter = iter + 1;
    end
    k = button;
end

if (iter > 0)
    eval(['textwrite(outlold,num2str(i),'','sl',num2str(i),...
    'old.asc',sz11(1))']);
    outtot2 = outtot2 - (outtot2 == i)*i + (out2/255)*i;
end

```

```

end
eval(['clear x',num2str(i),num2str(iter)])
eval(['clear y',num2str(i),num2str(iter)])

set(fg1,'Name','CLICKING CORRESPONDING POINTS')
set(fg2,'Name','CLICKING CORRESPONDING POINTS')
helpdlg('CLICK ON CORRESPONDING POINTS','Message');
w = waitforbuttonpress;
pause(2)

% This is the loop to allow the user to alternately click on corresponding
% regions in both the images, one region at a time. Again using "ginput", the
% user can click alternately on a matching region in the target, select the
% corresponding region in the reference and click on another region in the
% target and so on.
% To specify fragmented regions, the user clicks anywhere on each of
% the fragments.
% The clicked points are written into a file, with the number of
% fragments preceding the actual coordinates of the clicked points.
% Points corresponding to the target are written first, followed by
% the corresponding points on the reference. This is to facilitate
% accurate region growing, which will be performed by a C program.

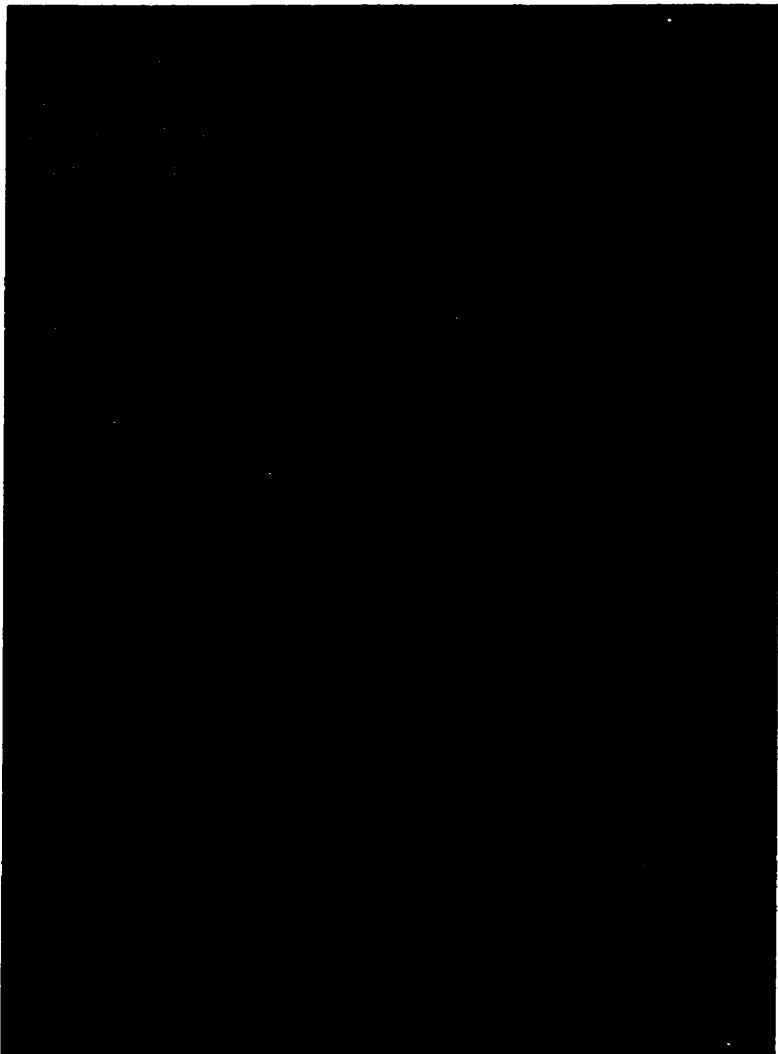
k = 1;
while k ~= 3,
    figure(1);
    [y1,x1,button] = ginput;
    if button == 3,
        count(1) = count(1) + 1;
        f1 = [x1 y1];
        sz1 = size(f1);
        figure(2)
        [y2,x2] = ginput;
        f2 = [x2 y2];
        sz2 = size(f2);
        pt = [sz1(1) reshape(f1',prod(sz1),1)' sz2(1)...
              reshape(f2',prod(sz2),1)'] - 0.5;
        pt = round(pt);
        fprintf(fid,'%d ',pt);
        fprintf(fid,'\n');
    end
    k = button;
end
end
helpdlg(['You have selected ',num2str(sum(count)),' regions'],'Message');

% The program outputs four files: (1) A "coordinates data"
% file containing the coordinates of all
% clicked points. Since individual fragments of a region can be specified
% using this program, the number of fragments forming a region are specified
% in this file before the coordinates of the seed points in each of these
% fragments. (2) A "count" file which contains the number of regions selected
% for each density slice. (3) The target ST image after modification.
% (4) The reference ST image after modification.
% Since region separation operations can change
% the reference and target ST images, these are written
% after modification into their original files.

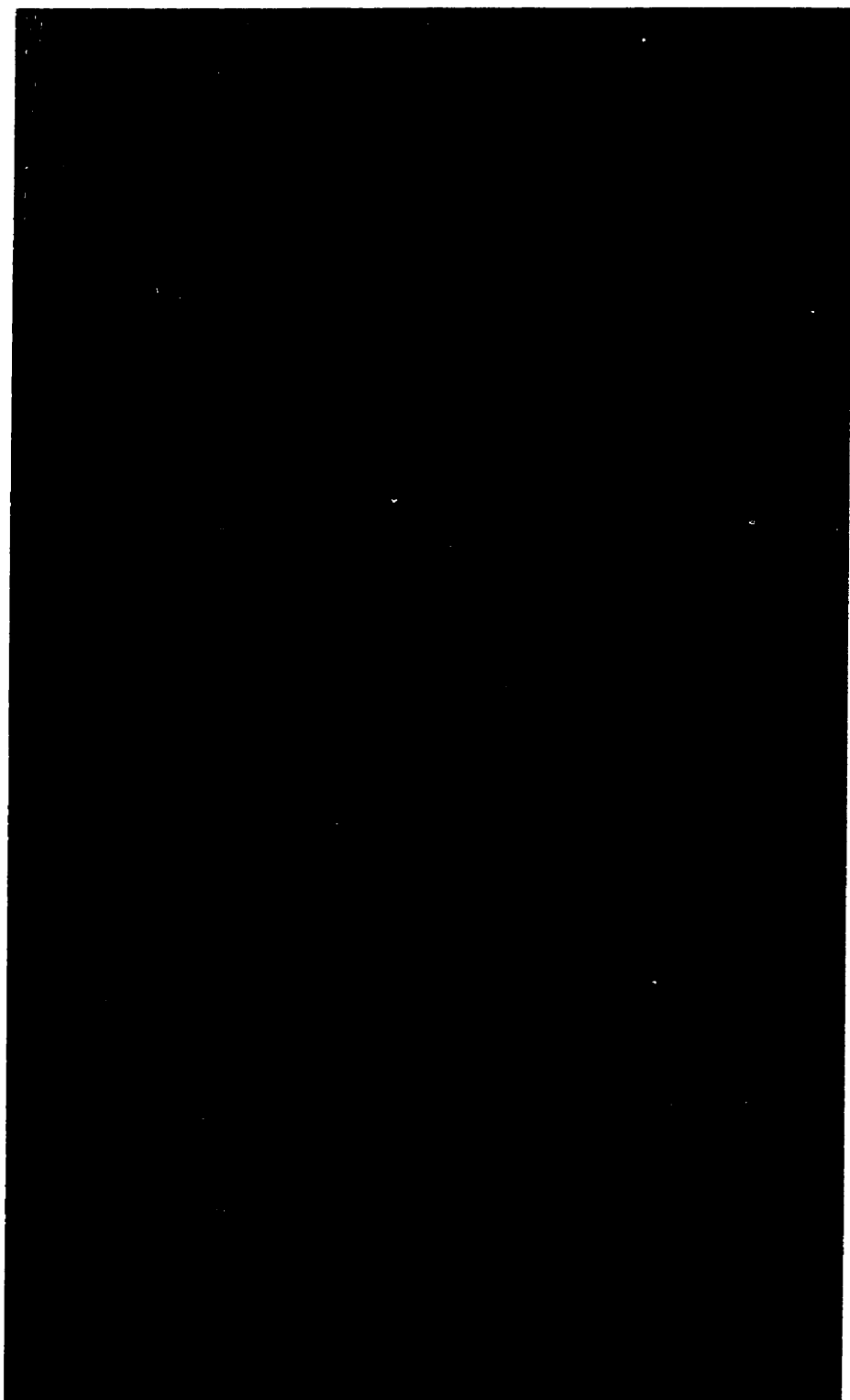
textwrite(count,[patthnmct,outctfile],1);
textwrite(outtot1,[pathnmin1,infile1],sz1(1));
textwrite(outtot2,[pathnmin2,infile2],sz2(1));

```


Startup Message



Instructions



```
%FILENAME: WARPBACKTOREF.M  
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
%This is a simple Matlab function to map back the co-ordinates in the  
%intermediate image to the corresponding co-ordinates in the reference  
%image. It takes in as input the 4-column matrix "final" (which is the set of  
%corresponding points between the intermediate and the target image),  
%"final1", the list of corresponding points from the previous iteration  
%(which were used to create the intermediate image), and "coeff1", the TPS  
%coefficients relating the reference image to the intermediate image. It  
%then invokes the function "splcoeff", which performs an  
%inverse mapping to obtain the co-ordinates in the reference image,  
%corresponding to the given co-ordinates in the intermediate image. It  
%then returns "f", a 4-column matrix whose first 2 columns contain  
%co-ordinates of points in the target image, and the last two columns  
%contain the co-ordinates of their corresponding points in the reference  
%image.  
%Usage: f = warpbacktoref(final,final1,coeff1)  
  
function f = warpbacktoref(final,final1,coeff1)  
xyorig1 = splcoeff(final(:,3:4),final1(:,1:2),coeff1);  
f = [final(:,1:2) xyorig1];
```

```
# This is a Unix shell program to smoothen each density slice of a
# starbyte-transformed image (ST). Here it is assumed that a 4-bit protocol
# has been used. Hence the number 15 below has been used, because
# the density slices have been numbered from 0 to 15. If a 6-bit protocol
# had been used, the slices would be numbered from 0 to 63, and for an
# 8-bit protocol, they would be numbered from 0 to 255 and so on.
# The batch program dsmorphbatch (which in turn contains calls to
# batch programs for the executable C programs densityslice, morph,
# and convertvalues) is called 16 times. After all density slices have
# been cleaned, they are assembled back together using the Matlab program
# addimages.m. This program is called through a Matlab batch program called
# matcomb.
```

```
# There are 2 inputs to this shell program (1) The name of the uncleaned ST.
# The name is assumed to have a .asc extension. Hence, before being inputted
# to this program, all ST files to be cleaned are to have a .asc extension.
# Hence the filename has to be specified without any extension.
# (2) Size of the file (assumed to be square).
```

```
# The program then creates each density slice in files with names
# dsp920.asc to dsp9215.asc. After the cleaning operation,
# these are loaded into the program addimages.m
# and then added together. The new cleaned ST is written into a
# file "sbtot.asc" by addimages.m and then moved into the original ST file.
```

```
#!/bin/csh
setenv i 0
while ($i <= 15)
    dsmorphbatch $1 $i dsp92$i $2
    setenv i `expr $i + 1`
end
matcomb addimages.m output.out
mv sbtot.asc $1.asc
```

#This Unix shell program can be used to obtain the unwarp image, after
#matching regions have been extracted using the Matlab program "uicp".

#The inputs to this program are:

#\$1 and \$2: Target and reference ST (after they have been updated by uicp.m)

#\$3: Co-ordinate data file returned by uicp.m

#\$4: Count file returned by uicp.m

#\$5: Name of output file which will contain the list of control points

returned by the C program "manualregion"

#\$6: Size of the images

#\$7: The reference image (without the .asc extension)

#\$8: The file into which the unwarp image has to be written (without
the .asc extension). The .asc is added automatically.

```
manualbatch $1 $2 $3 $4 $5.dat $6
echo "load $7.asc" > uicpunwarp.m
echo "load $5.dat" >> uicpunwarp.m
echo "[temp,coeff] = splinewpimg($7,$5);" >> uicpunwarp.m
echo "textwrite(temp,'$8.asc',$6)" >> uicpunwarp.m
echo "clear temp $7" >> uicpunwarp.m
echo "save $5.dat" >> uicpunwarp.m
matcomb uicpunwarp.m output1.out
```

#This Unix shell program can be used to run AUTOCP in non-iterative mode.
#Inputs to this program are:

#\$1: Target ST (specified with the .asc extension)
#\$2: Reference ST (specified with the .asc extension)
#\$3: Threshold size of regions to be considered for matching. Regions
#whose size falls below this threshold are discarded.
#\$4: Size of the images.
#\$5: File to save control points into. The control points are saved into.
#a variable "finall" in the specified file, which is given a ".mat" extension.
#Subsequent iterations can be run by using the Unix shell program "iautocp"

```
autobatch $1.asc $2.asc autol.dat auto2.dat counttot.dat $3 $4
echo "load autol.dat" > autocp.m
echo "load auto2.dat" >> autocp.m
echo "load counttot.dat" >> autocp.m
echo "[ef,ed,finall,numarray,ct] = auto4(autol,auto2,counttot,.07,3,16,112);" >> autocp.m
echo "clear autol auto2 counttot ed ef ct numarray" >> autocp.m
echo "save $5" >> autocp.m
matcomb autocp.m output1.out
```

```
%FILENAME: IAUTOCP
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
#This Unix shell program can be used to run AUTOCP in iterative mode.
#After the 1st iteration is completed using the shell program autocp and
#the list of corresponding points have been saved to a file ($2), this
#program can be called to calculate the intermediate image using TPS,
#obtain their starbyte transforms, clean these STs, and run the C
#program autoregion.c on these STs. After this, the Matlab program auto4
#is called to obtain the final list of corresponding points between the
#intermediate and the target image. The co-ordinates of points on the
#intermediate image are mapped back to co-ordinates in the reference
#image using the Matlab program warpbacktoref and the results of the
#second iteration are saved in the same file as the results of the first
#iteration. This program can be run as many times as necessary in order
#to perform subsequent iterations.
```

```
#The 11 inputs for this program are:
```

```
#$1: Target ST (specified with the .asc extension)
#$2: File containing control points (without the .mat extension) returned
#by the shell program
#autocp. The variable containing the control points is always named
#"final1", so that the results of subsequent iterations can be
#final2, final3 etc.
#$3: prefix of file names to save intermediate images. The iteration
#number is appended to this input. (specify the name without an extension, the #extension .asc is
added automatically)
#$4: prefix of file names to save the STs of intermediate images (again, the #.asc extension is a
dded automatically)
#$5: The reference image file (without the .asc extension)
#$6: current iteration number
#$7: $6 + 1
#Since iautocp is always run after the shell program "autocp", the same #neighborhood size and pr
otocol have to be used for the starbyte
#transformation performed here, as was used earlier.
#$8: Neighborhood size used for starbyte transformation
#$9: Protocol used for starbyte transformation
${10}: Size of the images
${11}: Threshold size to be used for the C program "autoregion"
```

```
#!/bin/csh
echo "load $5.asc" > si.m
echo "load $2" >> si.m
echo "path(path, '/home/radhika/bin')*" >> si.m
echo "[temp,coeff$6] = splnewpimg($5,final$6);" >> si.m
echo "textwrite(temp, '$3$6.asc', ${10})" >> si.m
echo "clear temp $5" >> si.m
echo "save $2" >> si.m
matcomb si.m output2.out
starbytebatch $3$6.asc $4$6.asc $8 $9 ${10}
clean $4$6
autobatch $1 $4$6.asc autol.dat auto2.dat counttot.dat ${11} ${10}
echo "load $2" > autorun.m
echo "load autol.dat" >> autorun.m
echo "load auto2.dat" >> autorun.m
echo "load counttot.dat" >> autorun.m
echo "[ef,ed,final,numarray,ct] = auto4(autol,auto2,counttot,.07,3,16,112);" >> autorun.m
echo "final$7 = warpbacktoref(final,final$6,coeff$6);" >> autorun.m
echo "clear final autol auto2 counttot ed ef ct numarray" >> autorun.m
echo "save $2" >> autorun.m
matcomb autorun.m output.out
```

%FILENAME: AUTOBATCH

%WRITTEN BY: RADHIKA SIVARAMAKRISHNA

This is a batch program to call the executable program "autoregion",
whose source code is in the file "autoregion.c".

```
echo $1 > tempfilt
echo $2 >> tempfilt
echo $3 >> tempfilt
echo $4 >> tempfilt
echo $5 >> tempfilt
echo $6 >> tempfilt
echo $7 >> tempfilt
autoregion < tempfilt > tempfilt1
rm tempfilt
rm tempfilt1
```



```
%FILENAME: CVBATCH  
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
# This is a batch program to call the executable program "convertvalues",  
# whose source code is in the file "convertvalues.c".
```

```
echo $1 > tempfilt  
echo $2 >> tempfilt  
echo $3 >> tempfilt  
echo $4 >> tempfilt  
convertvalues < tempfilt > tempfilt1  
rm tempfilt  
rm tempfilt1
```

```
%FILENAME: DSBATCH  
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
# This is a batch program to call the executable program "densityslice",  
# whose source code is in the file "densityslice.c".
```

```
echo $1 > tempfilt  
echo $2 >> tempfilt  
echo $3 >> tempfilt  
echo $4 >> tempfilt  
echo $5 >> tempfilt  
echo $6 >> tempfilt  
densityslice < tempfilt > tempfilt1  
rm tempfilt  
rm tempfilt1
```

```
%FILENAME: DSMORPHBATCH  
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
# This is a batch program to call the executable programs  
# densityslice, morph and convertvalues in sequence,  
# to smoothen each slice of the starbyte-transformed images.
```

```
dsbatch $1.asc temp.out $4 $2 $2 0  
ocbatch temp.out $4 plus temp.out  
cvbatch temp.out $3.asc $4 $2  
rm temp.out
```

%FILENAME: MANUALBATCH

%WRITTEN BY: RADHIKA SIVARAMAKRISHNA

This is a batch program to call the executable program "manualregion",
whose source code is in the file "manualregion.c".

```
echo $1 > tempfilt  
echo $2 >> tempfilt  
echo $3 >> tempfilt  
echo $4 >> tempfilt  
echo $5 >> tempfilt  
echo $6 >> tempfilt  
manualregion < tempfilt > tempfilt1  
rm tempfilt  
rm tempfilt1
```

```
%FILENAME: OCBATCH
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
# This is the batch program to perform the morphological
# operation on each slice of the starbyte-transformed images.
# $1 is the input file here.
# $2 is the size of the images (assumed to be square).
# $3 is the type of structuring element
# which can be 3x3, 5x5, plus etc.
# $4 is the output file.
# The morph program accepts only files in the Sun
# raster format. The package comes with a program called "asc2ras"
# which converts the input image to the raster format before being input to
# the morph program. After the morph program the output is converted back
# to a text format using the program "ras2asc" (also provided in the morph
# package). There are two calls to morph, one for the open operation, followed
# by one for the close operation.
```

```
asc2ras < $1 > temp1.out -X $2 -Y $2
morph < temp1.out > temp2.out -m o -i b -s b -k $3
morph < temp2.out > temp3.out -m c -i b -s b -k $3
ras2asc < temp3.out > $4
rm temp1.out
rm temp2.out
rm temp3.out
```

```
%FILENAME: STARBYTEBATCH
```

```
%WRITTEN BY: RADHIKA SIVARAMAKRISHNA
```

```
# This is a batch program to call the executable program "starbyte",  
# whose source code is in the file "starbyte.c".
```

```
echo $1 > tempfilt  
echo $2 >> tempfilt  
echo $3 >> tempfilt  
echo $4 >> tempfilt  
echo $5 >> tempfilt  
starbyte < tempfilt > tempfilt1  
rm tempfilt  
rm tempfilt1
```