List of Figures

1.1	Random Reference Quantization	5
1.2	Test Targets	7
1.3	Image Processing for Position Estimation	10
1.4	The CCD Camera	11
1.5	A Gaussian Form	15
1.6	An Example Locale Pattern	17
2.1	Example Target Shapes	30
2.2	The Conjugate Planes of a Lens	31
2.3	The Point Spread Function	31
2.4	Lens Distortion	32
2.5	A Photo-sensor Response Profile	33
2.6	The Gaussian Dot	40
2.7	A Binary Pixel Patterns for the Gaussian Dot	40
2.8	Locales for the Gaussian Dot	42
2.9	Periodic Extension of Locales	42
2.10	A Locale Map for Cross-A	44
2.11	A Locale Map for Cross-B	44
2.12	Locale Map Generation	46
2.13	Orientation and Offset of an Edge	48
2.14	The (b, θ) Locale Map for an Edge	50
2.15	Disconnected Locales	52
2.16	Another Disconnected Locale	52
2.17	Adjacent Locales	54
2.18	Locale Boundary Multiplicity	55
2.19	Image-Adjacency VS Position-Adjacency	56
2.20	A Locale Network in Image Space	57

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**HIGH PRECISION
POSITION ESTIMATION
IN DIGITAL IMAGE METROLOGY**

by

DAVID IAN HAVELOCK, B.Sc., M.Sc.

**A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of the requirements
for the degree of
Doctor of Philosophy**

**Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
5 July, 1989**

**© Copyright
1989, David I. Havelock**

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

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

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

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

ISBN 0-315-54389-2

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

**"HIGH PRECISION
POSITION ESTIMATION
IN DIGITAL IMAGE METROLOGY"**

submitted by
DAVID IAN HAVELOCK, B.Sc., M.Sc.

in partial fulfilment of the requirements
for the degree of Doctor of Philosophy.

David Havelock 7/31/89

Chair, Department of Systems and Computer Engineering

David Havelock

Thesis Co-supervisor

E. Pelletier

Thesis Co-supervisor

F.C.A.

External Examiner

Carleton University

5 July 1989

Abstract

Geometric precision in digital images refers to the ability to determine parameters such as position and orientation for an object in an image. This precision is fundamentally limited by the spatial sampling and the intensity quantization of the image digitization process. By using knowledge about the targets and about the image digitization process, errors in the estimation of geometric parameters can be minimized. A unifying theory of geometric precision is developed, based on the concept of position equivalence classes called *locales*. The concept of locales leads naturally to the definition of an optimally precise position estimate and a new algorithm is presented which implements this optimal estimate. The algorithm is applicable to all target shapes and a broad range of digitizing schemes. It can be generalized to geometric parameters other than (x, y) position. Simulations are used to demonstrate that the algorithm has superior position estimation precision and is robust in the presence of noise. Based on the fact that the algorithm is optimal and its performance can be predicted, it may serve as a performance benchmark and provide an objective measure of the noise level in an image. A new technique is reported here for determining the available geometric precision in digital images. The technique provides an easily computed bound on position estimation precision for arbitrary target shape and digitizing scheme. It is also shown how the distribution of geometric precision can be displayed by a simple graphical method which generates a locale map.

Acknowledgements

I am grateful for the support and encouragement offered to me by the National Research Council of Canada (NRC). I thank Dr. Marek J. Laubitz, Director of the Division of Physics at NRC, for his unqualified support throughout times of scarce resources and difficult organizational changes. I thank the past members of the Photogrammetric Research Section of NRC for their support, and in particular, I thank Zarko Jaksic, then Section Head, for leading me to this interesting and challenging research area. I thank Paul Cripwell for his detailed comments on early drafts of the manuscript. I thank Dr. James R. Pekelsky for many engaging and valuable discussions throughout this research. I wish to thank Dr. David Coll of Carleton University and Dr. Emil Petriu of the University of Ottawa for supervising this thesis project and for providing a source of patient, experienced advice. Finally, and eternally, I am grateful to my wife, Barbara, who has unhesitatingly undertaken so many extra chores on my behalf.

To my Parents

Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
Table of Contents	vi
List of Figures	xi
List of Symbols	xiv
1 INTRODUCTION AND BACKGROUND	1
1.1 Terms of Reference	2
1.1.1 Metrology	2
1.1.2 Image metrology	6
1.1.3 Digital image metrology	8
1.1.4 Intensity quantization and spatial sampling	14
1.2 Statement of the Problem	17
1.3 State of the Art	19
1.3.1 Digital image representation and mensuration	20
1.3.2 Videometry: characterizing solid state cameras.	22
1.4 Scope and Assumptions	23
1.5 Summary of Contributions	25
1.6 Document Organization	26

CONTENTS

2	THE LOCALES FORMALISM	27
2.1	The Target Model	27
2.1.1	Target position space: D	28
2.1.2	The brightness function: f	29
2.1.3	The optical image function: O	29
2.1.4	The entity function: E	32
2.1.5	The image function: I	34
2.1.6	The digital image: I	35
2.2	Definition of Locales	35
2.2.1	The sampling window: W	35
2.2.2	Image space: Q^K	36
2.2.3	Locales: $\mathcal{L}$	37
2.2.4	Periodicity of locales	39
2.2.5	Example locales	39
2.3	Generating Locale Maps	41
2.3.1	Superpositioned contours	43
2.3.2	Periodicity in the locale map	45
2.3.3	Unconventional locale maps	45
2.3.4	Locale maps for straight edges	47
2.4	The Topology of Locales	49
2.4.1	Connectedness	51
2.4.2	Convexity of locales	52
2.4.3	Adjacency	53
2.4.4	Scattered locales	56
3	BOUNDS ON PRECISION	59
3.1	Optimal Position Estimation	60
3.2	Lower Bounds on Estimation Errors	63
3.2.1	Dependence on locale map parameters	65
3.2.2	Lineal density and the directional MSE	67
3.3	Bounds on Locale Density	70
3.3.1	The area density of locales	71
3.3.2	Total variation	72
3.3.3	Determining the total variation	74
3.3.4	The lineal bound	76

CONTENTS

3.4	Examples	78
3.4.1	Binary disk	79
3.4.2	Gaussian dot	80
3.4.3	Binary edge segment	81
4	POSITION DECODING	88
4.1	Purpose of the Algorithm	89
4.2	Position Decoding Generalities	90
4.2.1	Position decoding	90
4.2.2	Analogy with signal decoding	91
4.3	Basic Position Decoding	93
4.3.1	Intersection of feasible regions	94
4.3.2	Locale centroid and size	95
4.3.3	The truth map	96
4.3.4	Dilation of feasible regions	98
4.3.5	Iterated truth map resolution	100
4.3.6	Partial evaluation	104
4.4	Position Decoding with Noise	104
4.4.1	Noise and the likelihood function	105
4.4.2	The cost function and feasible regions	106
4.4.3	The truth map revisited	108
4.4.4	Scattered locales	108
4.4.5	Locale weighting	109
4.4.6	The tolerance parameter: backtracking	111
4.5	Implementing Position Decoding	113
4.5.1	Target model parameter selection	114
4.5.2	Target model format	114
4.5.3	Selecting active pixels	116
4.5.4	Cost function selection	119
4.5.5	Pixel defects	120
4.6	Evaluating Position Decoding	121
4.6.1	Unique features of position decoding	122
4.6.2	Flexibility and stability	123
4.6.3	Optimality	124
4.6.4	Computational complexity	124

CONTENTS

5	SIMULATIONS	128
5.1	General Framework	130
5.1.1	Computational environment	130
5.1.2	Target models	130
5.1.3	Definition of parameters	135
5.1.4	Error estimation	136
5.1.5	Integrity checks	137
5.1.6	Computational efficiency	139
5.1.7	Repeatability	140
5.2	Performance Comparisons	140
5.2.1	Targets and parameters	141
5.2.2	Description of the algorithms	142
5.2.3	Results of the comparison test	146
5.3	Modelling Errors	147
5.4	Effects of Target Size	148
5.4.1	Target and parameters	149
5.4.2	Results and discussion	149
5.5	Additive Noise	153
5.5.1	Targets and parameters	154
5.5.2	Performance curves	156
5.5.3	Linearity of performance	157
5.5.4	Anisotropic performance	158
5.6	Summary	159
6	SUMMARY AND CONCLUSIONS	181
6.1	Summary	181
6.2	Future Research	184
6.3	Conclusions	188
A	DERIVATIONS AND PROOFS	190
A.1	COM Estimation Errors	190
A.2	Periodicity of Locales	191
A.3	Area Density of Locales	192
A.3.1	Locale density in image space	192
A.3.2	Locale neighbours in position space	194
A.3.3	Area-density of locales in position space	195

CONTENTS

A.4	The Shape Parameter	196
A.5	The Optimal MSE and the Parameters S , D and N_s	197
A.5.1	Lagrange minimization	198
A.5.2	Taylor expansion	199
A.6	Noise-Equivalent Target Parameters	200
B	DISCUSSIONS	206
B.1	Example Target Model Transformations	206
B.1.1	Optical blurring and distortion	207
B.1.2	Photosensitivity and area averaging	208
B.1.3	Filtering, resampling and quantization	208
B.1.4	Digital storage	210
B.2	Feasible Region Dilation	210
B.3	The Modified Lineal Bound	211
B.4	Empty Resolution	213
B.5	Noise Tolerance and Locale Topology	214
C	SOFTWARE LISTINGS	217
C.1	The Position Decoding Modules	217
C.2	The Truth Map Display Modules	232
C.3	Test Software for the Gaussian Dot Target	234
C.4	Test Software for the Binary Edge Target	238
C.5	The Center-of-Mass Position Estimator	250
C.6	The Fourier-Phase Position Estimator	251
C.7	The Exponential-Interpolation Position Estimator	253
	BIBLIOGRAPHY	256

List of Figures

1.1	Random Reference Quantization	5
1.2	Test Targets	7
1.3	Image Processing for Position Estimation	10
1.4	The CCD Camera	11
1.5	A Gaussian Form	15
1.6	An Example Locale Pattern	17
2.1	Example Target Shapes	30
2.2	The Conjugate Planes of a Lens	31
2.3	The Point Spread Function	31
2.4	Lens Distortion	32
2.5	A Photo-sensor Response Profile	33
2.6	The Gaussian Dot	40
2.7	A Binary Pixel Patterns for the Gaussian Dot	40
2.8	Locales for the Gaussian Dot	42
2.9	Periodic Extension of Locales	42
2.10	A Locale Map for Cross- <i>A</i>	44
2.11	A Locale Map for Cross- <i>B</i>	44
2.12	Locale Map Generation	46
2.13	Orientation and Offset of an Edge	48
2.14	The (b, θ) Locale Map for an Edge	50
2.15	Disconnected Locales	52
2.16	Another Disconnected Locale	52
2.17	Adjacent Locales	54
2.18	Locale Boundary Multiplicity	55
2.19	Image-Adjacency VS Position-Adjacency	56
2.20	A Locale Network in Image Space	57

FIGURES

2.21	Scattering of Locale Clusters	58
3.1	Skewed Position Estimates	63
3.2	Bounds on Precision	64
3.3	The Lineal Density of Locales	67
3.4	Notation for a line segment	69
3.5	Bounds on Area Density of Locales in Image Space	72
3.6	Determining the Lineal Variation.	75
3.7	Unimodal and Multimodal Functions	76
3.8	The Total Variation of the Binary Disk	80
3.9	The Total Variation of the Gaussian Dot	81
3.10	Deviation of The Total Variation from Linearity	82
3.11	A Locale Map for a Binary Edge	83
3.12	The Total Variation for the Binary Edge	85
4.1	Decoding a Position	93
4.2	The Feasible Region	95
4.3	Intersecting Feasible Regions	96
4.4	Missing Small Locales	98
4.5	Region Dilation	99
4.6	Missing Thin Regions	100
4.7	Iterated Truth Map Resolution	101
4.8	Locale Centroid Estimation Errors	103
4.9	The Likelihood Function	106
4.10	Nearest Locales	110
4.11	Scattered Locales	111
4.12	Backtracking	112
4.13	Active Pixels for a Binary Edge	116
4.14	Selecting Active Pixels	117
4.15	The Active Pixel Locale Map	119
4.16	The Computational Cost of the Truth Map Size	126
4.17	The Computational Cost of Backtracking	126
5.1	The Pixel List for a Binary Edge	133
5.2	The Pixel List for a Gaussian Dot	133
5.3	The Locale Map for a Binary Edge	134
5.4	Thresholding a Binary Edge	155

FIGURES

5.5	Additive Noise in a Binary Edge	155
5.6	Performance Comparisons Without Noise	161
5.7	Detailed Comparisons Without Noise	162
5.8	Performance Comparisons With Noise	163
5.9	Detailed Comparisons With Noise	164
5.10	Tolerance to Modelling Errors	165
5.11	Edge Offset Estimation Errors	166
5.12	Edge Orientation Estimation Errors	167
5.13	Edge Offset and Orientation Maximum Errors	168
5.14	Anisotropic Orientation Estimation Errors	169
5.15	Anisotropic Offset Estimation Errors	170
5.16	Optimality Demonstrated: Anisotropic Errors	171
5.17	Linear Performance for a Noisy Gaussian Dot	172
5.18	Non-Linear Performance for COM Estimate	173
5.19	Orientation Estimation Precision for Noisy Edges	174
5.20	Offset Estimation Precision for Noisy Edges	175
5.21	Performance Linearity with A'	176
5.22	Linearity of Edge Orientation Errors with K'_θ	177
5.23	Linearity of Edge Offset Errors with K'_b	178
5.24	Isotropic Edge Orientation Errors with Noise	179
5.25	Isotropic Edge Offset Errors with Noise	180
A.1	Slope and Offset for RMSE in Orientation	204
A.2	Slope and Offset for RMSE in Offset	204
A.3	Estimating K'_θ	205

List of Symbols

A	Area of D . Amplitude of a Gaussian dot.
A_ℓ	Area of locale $\ell \in D$.
b, θ	Offset b and orientation θ of an edge.
$b_{ij}(\theta_{ij})$	Sinusoidal contours of a locale map for an edge target.
b_ℓ	Skew of a position estimate.
$b_{\ell,x}$	Skew in the x -direction of a position estimate.
B_1, B_2, B_3	Bounds on the number of locales in image space.
c_k	Range of value for $\mathcal{I}_{i_k j_k}$ over D .
c_r	Truth map resizing parameter.
$C_k; C$	Cost function for pixel value $\tilde{p}_k$; Overall cost function
C_t	Tolerance parameter
D	Size parameter for a locale pattern.
D	Target position space.
D'	Alternate target position space for binary edges.
ε_ℓ	Deviation of $ X_\ell $ from $S/(N_\alpha s_\ell)$.
$e_{\ell,x}$	Average distance between $x_\ell(y)$ and the locale center.
E, E_{ij}	Entity function for sensor indices (i, j) .
$f, f_{\tilde{p}}, f_{\tilde{x}}$	Target brightness function.
h, h_t, h_0	Edge intensity profile function with binary quantization threshold h_t and maximum value h_0 .
$\mathcal{I}, \mathcal{I}_{ij}$	Image function for pixel indices (i, j) .
$\mathbf{I}, \mathbf{I}_{\tilde{p}}, \mathbf{I}_{\tilde{x}}$	Digital image representing the target.
k, n, m	Integer index variables
K	Number of samples in the sampling window W . The dimensionality of image space. The number of pixels in the digital image. The length of a binary edge.
$\Lambda(y); \Lambda_\ell(y)$	Line segment parallel to the x -axis at a distance y from the axis; Restriction of this line segment to the interior of ℓ .

SYMBOLS

ℓ, ℓ_1, ℓ_2	Locale in position space or in image space.
$m_\ell^{(2)}$	Second central moment of locale ℓ .
$M_j(y), m_j(y)$	Local maxima and minima, respectively, along $\Lambda(y - j)$.
MSE	Mean square error.
MSE _x	Directional mean square error in the x -direction.
MSE ₀	Optimal mean square error (non-directional).
MSE _{ℓ}	Optimal mean square error within a locale ℓ .
N	Total number of locales. Width of the truth map.
N_a	Area-density of locales.
N_l	Average lineal density of locales.
$N_l(y)$	Lineal density of locales along $\Lambda(y)$.
$O, O_{\bar{p}}, O_{\bar{x}}$	Optical image function.
Π	Mapping of points in target position space to points in image space resulting in the definition of position equivalence classes and locales.
$\Phi_1, \Phi_2, \Phi_3, \Xi$	Transformations leading to the digital image I.
$p_{ij}; p_k$	Pixel value for pixel index (i, j) ; The k^{th} pixel value in the sampling window.
$\bar{p}, \bar{p}_k, \{\bar{p}_k\}$	Observed pixel value; Pixel value for index k ; Set of pixel values.
$P(a; b)$	Probability of observing a given b .
q_i	Component of $\vec{q}$ in image space.
$\vec{q}, \vec{q}_1, \vec{q}_2$	Point in image space.
$Q, \mathbf{Q}, Q$	Intensity quantization scheme having a set $\mathbf{Q}$ of Q quantization values.
$\mathbf{Q}^K$	K -dimensional image space.
$r; r_k$	Radius of a binary disk; Radii of contours of a Gaussian dot.
$R_{\bar{p}}; R_{\{\bar{p}_k\}}$	Feasible region for $\bar{p}_k$; Feasible region for the set $\{\bar{p}_k\}$.
RMSE	Root mean square error: $\sqrt{\text{MSE}}$.
RRMSE	Reciprocal root mean square error: $1/\text{RMSE}$.
s_ℓ	Shape coefficient of an individual locale.
S_x, S_y, A_x, A_y	Scale and offset coefficients for the truth map.
S	Shape parameter for a locale pattern.
$T; T'; T^k$	Main truth map or position estimator; Auxiliary truth map; The k^{th} iteration of the main truth map.
T_0	Optimal position estimator, $T_0 : \ell \mapsto (\bar{x}_\ell, \bar{y}_\ell)$.
U	One-dimensional unit step function.
$V; V(y); v(y)$	Total variation; Lineal variation; Variation.

SYMBOLS

V_b	Total variation in edge offset b .
$V_b(\theta)$	Lineal variation in b at orientation θ .
V_θ	Total variation in edge orientation θ .
$V_\theta(\theta)$	Lineal variation in θ at orientation θ .
W	Sampling window.
χ_ℓ	Characteristic function of locale ℓ . Equals 1 within ℓ and 0 elsewhere.
χ_{ij}	Set of target positions for which p_{ij} or $\mathcal{I}_{ij}$ is non-zero.
(i, j)	Indices of a pixel or a sensor element
(x, y)	Position in target parameter space D .
(x_0, y_0)	Current target position.
(x_{ij}, y_{ij})	Coordinates of sensor (i, j) in the image plane.
$(\hat{x}, \hat{y}); (\hat{x}_\ell, \hat{y}_\ell)$	Estimated target position; Position estimate for locale ℓ .
$(\bar{x}_\ell, \bar{y}_\ell)$	Centroid of locale ℓ .
$(x_\ell(y), y)$	Mid-point of the line segment $\Lambda_\ell(y)$.
$[\bar{x}]_n$	Position equivalence class in position space.
$ \cdot $	Magnitude operator for numbers (such as pixel values), the area operator for regions (such as locales in position space), and the enumeration operator for discrete sets such as image space.
$[\cdot]$	Quantization by integer truncation.
$\Pi : D \mapsto \mathcal{L}$	Functional mapping notation. The domain D of the mapping Π is mapped to range $\mathcal{L}$.
$\{ (x, y) : y=y_0 \}$	Set notation. The designated elements are members if they satisfy the conditions following the colon. When no colon and conditions are present, set elements are explicitly and exhaustively indicated (as in: $\{-1, 0, 1\}$) or collectively indicated (as in: $\{p_{ij}\}$).

Chapter 1

INTRODUCTION AND BACKGROUND

"I often say that when you can measure what you are speaking about, and express it in numbers you know something about it; but when you cannot measure it, when you cannot express it in numbers your knowledge is of a meagre and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science whatever the matter may be. "

Lord Kelvin

This thesis investigates the limitations imposed on position estimation precision by the discrete nature of digital images. This is a subject of fundamental importance to the young and developing field of digital image metrology and several new contributions are presented here.

The introductory chapter begins with a discussion of metrology and the theory of measurement. An example of the effects of intensity quantization and two-dimensional spatial sampling on position estimation in digital images is then presented, followed by a statement of the specific problems which this thesis addresses and a description of existing work in the area. The general approach to the research problem is described and the salient contributions of the thesis are then summarized. The chapter closes with an outline of the rest of the document.

1.1 Terms of Reference

In this section, the problem of high precision position estimation in digital image metrology is presented in the context of the broader field of dimensional metrology. Within measurement science, as metrology is often called, are the nested disciplines of dimensional metrology, image metrology, and digital image metrology. To appreciate the problem at hand it is necessary to have a basic understanding of metrology and how spatial sampling and intensity quantization introduce the interesting limitations on position estimation precision which are being investigated here. To illustrate these limitations on precision, some simple examples are discussed.

1.1.1 Metrology

Metrology, or measurement science, is as diverse in its scope as it is in its application. It has a history going as far back as the ancient Greeks [HBMS, s.1.2]. Modern measurement science is a discipline which is practiced in institutions and manufacturing centres throughout the world. It is coordinated through a pyramidal system of *traceability*, whereby each level in the pyramid maintains traceability to the level above by means of calibration procedures.

At the top of the pyramid are the *international standards*, which are the embodiment of physical units of measure such as the meter, second, and kilogram. These

standards are passed down through layers of the pyramid by careful calibration techniques to a network of primary, secondary and tertiary standards, all of which remain traceable to the international standards [Vanie88].

There is a strong interdependence between all levels of the measurement science pyramid. At the base of the pyramid, measurement science may be so diversified and integral to production procedures that it can go unnoticed. Nevertheless, it has been strongly argued that [Quinn88]

“... advanced technology is critically dependent upon precision measurement and that progress in one is always linked to progress in the other. ”

Measurements are made by *instruments*, which sense some physical property and ascribe to it a descriptive numerical value. The numerical output of an instrument must be determined (calibrated) with respect to some physical unit or scale so that it can systematically describe some aspect of the world around us.

An informal definition of measurement is provided by Finkelstein, wherein measurement is described as a *quantization* process by which analogical quantities (the *measurand*) are assigned specific numerical values [Finke82]:

“Measurement is the process of empirical, objective, assignment of numbers to properties of objects or event of the real world in such a way as to describe them.”

The quantization process always leads to a *quantization error* component that affects the measurement. A more formal, mathematical theory of measurement, referred to as a *representational theory of measurement* can be found in [Finke82], [Pfan68] and [Krant71].

There are two principal qualities required for a good measurement. They are distinct and complementary, yet they are often confused or interchanged. The first is *precision*, which is the combined qualities of *discrimination* and *repeatability*. Discrimination describes the ability to obtain different results when a quantity changes

while repeatability describes the ability to obtain the same result when a quantity does not change. The second quality of a good measurement is *accuracy*, which describes the correctness of the representation.

Borrowing in part from the Handbook of Measurement Science [HBMS, s.3.2.2], the following definitions are offered;

Precision The quality which characterizes the ability of a measuring instrument to react with small changes in the quantity measured and to give the same value with no changes in the quantity measured.

Accuracy The quality which characterizes the ability of a measuring instrument to give indications approximating to the true value of the quantity measured.

An illustration of these terms is available from dimensional metrology, which is the discipline of measuring quantities such as length, area, angle, flatness and position. Simplistically, a ruler (or *scale*) is laid along an interval to be measured and the numerical difference between the numbers on the scale at the start and end of the interval provide the measure of the distance. The fineness of the graduations on the ruler and the stability of the ruler's structure determine the precision with which the measurement can be made. The alignment of the ruler with the measured interval and the calibration of the ruler's intervals in terms of an agreed standard determine the accuracy of the measurement.

A term *resolution* is sometimes confused with precision. The resolution of a ruler, for example, is the fineness of its graduations. Factors other than resolution, including the way the instrument is used, can affect the measurement precision.

The measurement quality is always limited by the level of technology available to implement the measurement; however, the quality of the measurement can generally be improved by using a priori knowledge, when it is available, about the quantization process and the measurand. The use of *random reference quantization* for analog to digital conversion is a technique which demonstrates this principle of utilizing a priori

FIGURE 1.1 Random reference quantization (RRQ) is performed by adding an auxiliary signal a to the primary input signal V prior to quantization to obtain V^* . The *dithered* output values V_q^* can be averaged to reduce the quantization error below that obtained without RRQ

knowledge to reduce quantization errors.

Based on the assumption that the input signal V is uniformly distributed over the quantization interval, a *dithering* technique can be used in conjunction with sample averaging to reduce the *quantization errors* [Casta71]. Dithering is done by randomly changing the quantization thresholds (and possibly the assignment values) by small amounts between successive quantization samples. Knowledge of the dithering probability distribution allows thus for a better estimation of the input signal from the statistics of the dithered, quantized output values.

In Figure 1.1 a special case of dithering, referred to as *random reference quantization* (RRQ) is depicted. An *auxiliary signal* a is added to the primary input signal V to obtain the dithered, quantized value V_q^* . Adding the auxiliary signal in this way is equivalent to shifting all of the quantization thresholds by the amount a .

If the auxiliary signal a is uniformly distributed on the interval $[-\frac{1}{2}\Delta, +\frac{1}{2}\Delta]$, where Δ is the width of the quantization interval, then the probability $P^*(q_k)$ of obtaining

an output value $V_q^* = q_k$, as a triangular function of the primary signal value V

$$p^*(q_k) = \max \left\{ 0, 1 - \frac{2|q_k - V|}{\Delta} \right\}. \quad (1.1-1)$$

The expected quantization error in the dithered case is the same as that of the non-dithered case; $E[(V_q^* - V)^2] = \Delta^2/12$ but, with dithering, N quantization values can be averaged to reduce the expected error by a factor of $1/N$. Furthermore, the mean of the averaged values approaches the *exact* value of V as N gets large.

The RRQ technique uses a priori knowledge about the auxiliary signal $\mathbf{a}$ and about how it interacts with the primary signal, to improve measurement precision.

1.1.2 Image metrology

As the name suggests, image metrology derives measurements from images which represent physical qualities of interest. Image metrology is most suitable for dimensional measurement applications for which direct contact with the object being measured is undesirable or difficult. Large or distant objects such as buildings or the landscape can be imaged with special cameras and the images subsequently used to make measurements. Image metrology is also useful for making measurements of dynamic processes, such as vehicle trajectories for crash test analysis [Kratk71].

The process of obtaining a measurement for image metrology involves a number of distinct stages. The first stage, image acquisition, is perhaps the most critical since it determines the information captured in the image. Information is also available from sources external to the image, such as camera calibration parameters which relate *image coordinates* to *object coordinates*, and targets with known geometry. Targets are often placed in the scene, for example, as ground targets in aerial photography. They may also be within the camera system itself, as in the case of fiducial marks and reseau grids [James77]. The choice of target shape, size and reflectance is an important consideration. It has been the subject of research for aerial photogrammetry [OConn67], [Ziema82] (see Figure 1.2) and occupies many paragraphs of the Design Handbook for Imagery Interpretation Equipment [HBIIE, s.3.1].

FIGURE 1.2 The effectiveness of targets for providing precise position information depends on their shape, size and contrast. A diverse collection of ground targets, being tested for their effectiveness in aerial surveying applications, is shown here. For applications using digital images, the selection of suitable targets is equally important.

Reproduced from [Ziema82]

The second stage is the determination of image coordinate data. A skilled operator, using sophisticated opto-mechanical equipment, determines the position of specific points in an image relative to the coordinate frame of the image. This *pointing* task is essential to dimensional image metrology. It provides the raw numerical data from which the final measurement is deduced. Determination of image coordinate data does not itself constitute the desired measurement since the data at this stage does not describe the physical quality of interest. The term *mensuration* has been used by various authors ([Forst87], [Gonsa76] and [Unruh82] for example) in reference to image coordinate determination.

The third stage is to combine image coordinate data with other information to obtain a *model* of the physical dimension of interest. In the case of stereo photogrammetry for example, image coordinate data from several images, information about camera calibration and orientation, and information about reference target geometry are all combined to obtain a three-dimensional model coordinate system. The difficulty of this process increases as more images and reference targets are involved. Computationally complex photogrammetric procedures such as *block adjustment* and *bundle adjustment* [Slama80] are used to provide a unified geometry for a collection of (overlapping) images.

Finally, the model data is related to the physical quality of interest, whether that quality be object position, area or volume. This process utilizes knowledge about how the reference targets are geometrically related to the physical quality. For example, a measurement of the diameter of a circle might be derived from the model coordinates of many targets known to lie equally spaced about its circumference. A least squares estimation procedure might be used to express this geometric knowledge algebraically.

The photographic camera is a familiar image acquisition mechanism for image metrology but it is not the only one. Radar imaging by synthetic aperture radar for remote sensing, acoustic imaging for ultra-sound medicine, and imaging by nuclear magnetic resonance for medicine are among the many other examples. Images need not conform to our conventional notion of pictures; laser interferometry and moiré fringe techniques generate geometric patterns of lines representing the object. Many other diverse examples exist but this thesis is mainly concerned with conventional (picture) images acquired with solid state cameras, as discussed in the next section.

1.1.3 Digital image metrology

A *digital image* is a two-dimensional array of integers, with each integer representing the image intensity in a small *picture element*, or *pixel*. The main advantage of digital imagery is that it can be stored in and processed by digital computers.

Digital images can be displayed for human viewing but manual mensuration of image coordinate data is time consuming and error prone. For applications in which the targets are well defined and the mensuration procedure is routine, it is possible to use digital image processing techniques to completely eliminate the human visual system from the process ([ElHak86], for example).

Automated Image Coordinate Mensuration. As an overview of this process, the steps involved in obtaining image coordinate data from a digital image are depicted in Figure 1.3. The digital image must first be transferred to the memory of the computer. Image enhancement for automated processing attempts to emphasize important features such as edges and suppress undesired detail. Image segmentation is then performed using the edges in the image to partition the image area into unqualified *blobs*. Each of these blobs is then examined by a target recognition procedure and all identified targets are catalogued.

Target detection and recognition provides an approximate position for each target. These initial position estimates are used by a subsequent subpixel position estimation procedure. As indicated in the figure, subpixel position estimation uses the non-enhanced digital image. This is because the enhancement and segmentation operations tend to corrupt the geometric information in the image. Enhancement and segmentation each address the *textural* and *contextual* information in the image, whereas subpixel position estimation addresses the *geometric* information in the image. Geometric information ultimately determines the quality of the target position estimate but it is a prerequisite to have successful target identification in the first place.

Image Acquisition: The CCD camera. A common source of digital images is the CCD camera. The abbreviation CCD means *charge coupled device*, which describes one of several methods of charge transfer in the integrated circuit technology

FIGURE 1.3 Sub-pixel position estimation for a target in a digital image is preceded by the image processing operations of enhancement, segmentation, candidate target selection and target recognition. These steps typically provide a localization of the target to within a pixel. Precise position estimation uses the original image data since processing, such as image enhancement, tends to corrupt the geometric information in the image.

Based on [ElHak86]

used for imaging sensors. Other methods of charge transfer exist, such as charge injection devices (CID), but the term *CCD camera* will be used here to refer liberally to any camera using a solid state imaging array. Solid state imaging arrays are of particular interest because of their excellent geometric regularity and intensity resolution. Their geometric regularity is due to the rigid monolithic structure of the integrated circuit and the precise photolithographic processes used in their production. Some basic characteristics and operating principles of these cameras are discussed below.

Figure 1.4 illustrates the components of a CCD camera, which are: a lens identical to that of a conventional camera, an integrated circuit containing the imaging array, and external controlling electronics to read, amplify and transmit the image data. A CCD camera can operate at the same light levels as conventional film cameras. The imaging array in the CCD camera is continually exposed to the scene illumination, generating an image 30 times a second. The solid state imaging array contains photo-

FIGURE 1.4 The principal components of a solid state camera are depicted. The lens is identical to that of a conventional camera with a typical focal length of 25mm (wide angle) and full aperture of about $f:1.4$. Aperture and focus adjustment are usually provided but no shuttering mechanism is necessary. The photographic emulsion is replaced by a small solid state imaging array. A *charge coupled device* (CCD) is one of several possible types of imaging arrays. The photosensitive area is about one square centimeter, compared to about 8 for commercial 35mm film. Accumulated charge in the imaging array is transferred serially to external electronics which control the imaging array integrated circuit and condition the data for transmission to a computer. In the computer, the image is represented as an array of integers.

sensors arranged in a regular two-dimensional grid. Sensor spacing is in the range 10 to 30 μm . A typical imaging array has between 400 and 1000 elements on a side and has a photosensitive area of about one square centimeter. Arrays having as many as 2048 elements on a side have been manufactured (see [Blouk85], for an example).

Individual photo-sensors accumulate electric charge proportional to the incident radiant energy by the process of photoelectric emission [Richt69, s.6.7]. Each sensor can hold about 250,000 electrons [Koson88] [Janes87] before it overflows into adjacent ones, causing the familiar phenomenon of *blooming*. The charge in a full sensor will contain noise equivalent to about 1000 electrons, allowing about 250 distinguishable levels of incident radiant energy.

The accumulated charge in each sensors is read from the array 30 times a second under the control of the external (off-chip) electronics in the camera body. The accumulated charge in each sensor is transferred from the photosensitive area to the peripheral read-out circuitry for amplification and transmission to a computer, where it is recorded as pixel values. The transmission of data from the imaging array to the computer is done serially; that is, one sensor value at a time. The sequence of transmitted image data is interspersed with timing signals. At the computer, the timing signals are stripped off and used for synchronization. The sensor data is digitized and stored as pixel values in an integer array. With careful synchronization, the indices of the array correspond to physical locations of the sensors on the CCD array; thus, a pixel value $p_{i,j}$ in the digital image corresponds to the output of the sensor at row (i) and column (j) on the CCD imager.

Most digital imaging systems currently adhere to an image transmission standard such as *RS170*[†] when transferring data from the camera to the computer. This standard has serious limitations for image metrology applications because it uses a continuous (analogue) signal to represent the pixel values and does not incorporate a pixel clock [Lenz88b].

Data Volume. The usual format aerial photograph, which is about 20 cm square, can represent much more data than a typical CCD camera image. The spatial and intensity resolution characteristics of the two mediums are quite different, however, and their relative data capacities may not reflect their relative potential for geometric precision.

Information is stored in a computer as *bytes* of data, with each 8-bit byte representing a number between 0 and 255 inclusive. One CCD camera image, with 8 bits per pixel and 1000 pixels on a side, has 10^6 bytes of data. For comparison, the data capacity of an aerial photograph, based on film emulsion grain geometry, is estimated

[†]Note the dichotomous use of the word *standard* here. Standards such as *RS170* are sometimes called *technical standards* or *specifications* to distinguish them from physical standards.

at about 3.6×10^5 bytes per square centimeter for Pan-X film [HBSPSE, s.5.7.5]. Accordingly, an aerial photograph which is 20 cm square has about 1.44×10^8 bytes of information – which is 144 times more data than a CCD image, or enough data to fill a bookshelf of 400 floppy disks.

In terms of spatial resolution, a digital image can resolve half its sampling frequency, according to the Nyquist criterion. The resolution of an aerial photograph is determined by the lens and emulsion and is in the order of 50 line pairs per millimeter [Ziema87]. In terms of pixel density, photographic resolution corresponds to about 100 pixels per millimeter. This implies 20,000 pixels across the width of an aerial photograph, as compared to 1000 across a CCD image.

The intensity resolution of a CCD image is 256 levels. An aerial photograph, with 1.44×10^8 bytes of information and $(20,000)^2$ pixels, has an intensity resolution per pixel of 2.9 bits, or about 8 intensity levels.

From this comparison, the CCD image is seen to have 20 times less spatial resolution and 32 times more intensity resolution than an aerial photograph, on a per pixel basis. The intensity resolution of a photograph can be improved by averaging over a larger area. The improvement is linear with the width of the area, so to obtain 256 intensity levels rather than 8, a patch of 32×32 pixels in the photograph must be averaged. This reduces the spatial resolution of the photograph to $20,000/32=624$ pixels, which is about the same as that of a CCD image.

It will be seen that the position estimation precision for a small dot in a digital image depends on the product of the spatial and intensity resolutions (Section 3.4.2) and so the theoretical precision of a CCD image approaches that of an aerial photograph if full advantage is taken of the available information. Due to the large intensity resolution and low spatial resolution of the CCD image format, its metrological potential can be realized only with techniques which achieve optimal *subpixel* precision.

Vernier Acuity. Research in human vision reveals that spatial resolution, or *acuity* can be specified either in the context of the ability to detect high spatial frequencies

(Nyquist theory), or in the context of the ability to align objects (*vernier acuity*). In human vision, *vernier acuity*, or *hyperacuity* as it is often called, is about 5 to 10 times better than acuity based on spatial frequency [Smith66, s.5.3]. Vernier acuity is analogous to subpixel position estimation in digital image metrology. It is appropriate to refer to subpixel position estimation as *vernier resolution*. In digital images, vernier resolution may be two orders of magnitude better than the spatial (Nyquist) resolution.

Digital image metrology can make use of the knowledge and techniques in the broader field of image metrology but, in regard to the limitation on measurement precision imposed by the discrete nature of the imagery, digital image metrology has need of specialized theory and techniques of its own. Quantization and sampling restrict measurement precision in a manner which are quite different from the effects of random noise. Conventional analytical methods, using Nyquist theory and analysis of variance, for example, do not adequately explain the relationship between position estimation errors and the discrete nature of digital imagery.

1.1.4 Intensity quantization and spatial sampling

To illustrate some of the effects which intensity quantization and spatial sampling have on position estimation precision, consider a small dot-shaped target defined by the Gaussian form (see Figure 1.5)

$$E(x + x_0, y + y_0) = A e^{-(x^2 + y^2)/2\sigma^2}, \quad (1.1-2)$$

where (x_0, y_0) is the position of the dot. The positive parameter A determines the amplitude, or intensity, of the image of the target. The width of the target is determined by σ . The centre of mass (COM) position estimate, defined as

$$\hat{x} = \frac{\iint x E(x, y) dx dy}{\iint E(x, y) dx dy} \quad \text{and} \quad \hat{y} = \frac{\iint y E(x, y) dx dy}{\iint E(x, y) dx dy}, \quad (1.1-3)$$

gives the true target position in the absence of noise; that is, $(\hat{x}, \hat{y}) = (x_0, y_0)$.

FIGURE 1.5 A circular dot is smoothed by the optical transfer function of the imaging system. Its imaged form is represented by a Gaussian function.

Assume that the image is spatially sampled so that the Gaussian function is defined at the integer coordinates $(x, y) = (i, j)$. Further assume that the target position (x_0, y_0) is within a unit square centred on the origin and only the 9 samples obtained with $i \in \{-1, 0, 1\}$ and $j \in \{-1, 0, 1\}$ are available for estimating the target position. The COM estimate now has the form

$$\hat{x} = \sum_{i=-1}^1 \sum_{j=-1}^1 \frac{iE(i, j)}{E(i, j)} \quad \text{and} \quad \hat{y} = \sum_{i=-1}^1 \sum_{j=-1}^1 \frac{jE(i, j)}{E(i, j)}. \quad (1.1-4)$$

Explicit calculation gives the following closed form expressions (see Appendix A.1 for details);

$$\hat{x} = \frac{\sinh(x_0/\sigma^2)}{\cosh(x_0/\sigma^2) + \frac{1}{2}e^{1/2\sigma^2}} \quad \text{and} \quad \hat{y} = \frac{\sinh(y_0/\sigma^2)}{\cosh(y_0/\sigma^2) + \frac{1}{2}e^{1/2\sigma^2}}. \quad (1.1-5)$$

This COM estimate gives a unique and distinct value for each target position, so it is perfectly *precise*, but since $(\hat{x}, \hat{y}) \neq (x, y)$, it is not *accurate* (as per the definitions on page 4).

Now assume that there is intensity quantization with no spatial sampling. In this case, the COM estimate has the form

$$\hat{x} = \frac{\iint x [E(x, y)] dx dy}{\iint [E(x, y)] dx dy}, \quad \text{and} \quad \hat{y} = \frac{\iint y [E(x, y)] dx dy}{\iint [E(x, y)] dx dy}, \quad (1.1-6)$$

where $[\cdot]$ indicates quantization, by integer truncation for example. This quantized Gaussian function has the same symmetry as the non-quantized Gaussian, so the COM estimate will give the true position of the target centre.

A digital image has *both* spatial sampling and intensity quantization. In this case, there is unresolvable uncertainty introduced in the position estimate. The COM estimate for a digital image is

$$\hat{x} = \sum_{i=-1}^1 \sum_{j=-1}^1 \frac{i[E(i,j)]}{[E(i,j)]}, \quad \text{and} \quad \hat{y} = \sum_{i=-1}^1 \sum_{j=-1}^1 \frac{j[E(i,j)]}{[E(i,j)]}. \quad (1.1-7)$$

A *binary* digital image has only two intensity levels, customarily assigned the values zero and one. In the case of the Gaussian form, assume that if $E(i,j)$ is greater than $\frac{1}{2}A$ then the value one is assigned and that the value zero is assigned otherwise. This leads to a very limited set of possible values for the position estimate $(\hat{x}, \hat{y})$, namely; the 13 element set

$$\left\{ (0,0), (0, \pm \frac{1}{2}), (\pm \frac{1}{2}, 0), (\pm \frac{1}{3}, \pm \frac{1}{3}), (\pm \frac{1}{2}, \pm \frac{1}{2}) \right\}. \quad (1.1-8)$$

Each of these values for the position estimates corresponds to a region, rather than a single point, of possible object positions. These regions will be referred to as *locales* and are the central issue of this thesis. Locales are regions of indistinguishable target position. The size of the locales determines the position estimation uncertainty caused by sampling and quantization, and so it determines a bound on the precision of the resultant measurement.

Locales are smaller than the size of a pixel. An individual pixel may contain many locales, as in Figure 1.6, where the locales for the binary Gaussian target are shown. For this example, the average locale area is 0.077.

Each locale contains a dot corresponding to the position estimate obtained by Equation 1.1-7. The best position estimate for a locale is provided by the centroid of the locale, as shown in Section 3.1. The ability to easily generate the locale pattern for a target, as in Figure 1.6, or to derive parameters for it such as the average locale size, leads to new methods of analyzing position estimation uncertainty.

FIGURE 1.6 The object with intensity function $E(x, y) = Ae^{-(x^2+y^2)}$ is spatially sampled and intensity quantized, resulting in regions within which the digital image of the object is constant. These regions of constant digital representation are called *locales* and are shown here within a unit square area of object positions. The square is the size of an image pixel. A dot within each locale indicates the position estimate obtained when the object is anywhere within the locale.

1.2 Statement of the Problem

To improve the quality of measurements obtained with digital images, it is helpful to have a full understanding of the effects of quantization and sampling on the geometric information in an image. To maximize the use of available information, the a priori knowledge provided by a target within the image, and the a priori knowledge about the quantization scheme must be combined. The interaction of the image content with the quantization scheme affects the available precision and must be taken into account to minimize the quantization errors to their essential minimum.

To design a digital image metrology system having the best possible precision,

it must be understood how its precision is affected by parameters such as intensity resolution, spatial sampling resolution and target shape. Prior to this work, there has been no comprehensive analysis of geometric precision in digital images and no indication of a general or unifying approach evolving. Without a foundational framework, research in this area has been disjointed and less effective.

For a given application there may be many contending position estimation algorithms and the best one may be selected on the basis of speed and precision. The speed of an algorithm can usually be estimated easily, but not its precision. A useful benchmark against which the performance of other algorithms may be compared can be provided by an algorithm which is theoretically optimal for extracting geometric information from digital images independent of the target shape or digitizing scheme. Such a benchmark has not been available, so that results for specific applications have no universal basis for comparison and cannot be used to infer performance in other applications.

To further the capabilities of digital image metrology it is necessary to gain a more complete understanding of (a) the effects of target shape and digitizing scheme on the representation of geometric information in digital images, and (b) the optimal extraction of geometric information from a digital image. These issues are both fundamental and complex. This research project undertakes to resolve specific components of these issues and to explore new techniques for the analysis of geometric precision in digital images. Specific objectives of this work are itemized below.

1. Determine the optimal position estimation precision for arbitrary target forms and arbitrary digitizing schemes.
2. Find a simple, practical method for computing a bound on the optimal precision obtainable from a digital image with arbitrary target form and digitizing scheme.
3. Design an algorithm which achieves the optimal precision.

4. Implement a variation of the theoretically optimal algorithm which is practical (computationally efficient) and suitable for use with noisy data.
5. Establish a unifying theoretical framework for the analysis of geometric precision in digital images based on the concept of locales.

For practical applications a simple method is needed for determining the optimal precision for a given combination of target shape and digitizing scheme. The true optimum is often difficult to determine, so the objective here is to find a simple approximation or bounding value. A theoretically optimal position estimation algorithm is an immutable benchmark against which the performance of other estimation algorithms can be rated. It also provides a computational method of determining the optimal precision. If the (noise-free) optimal algorithm can be made to be robust enough to be useful with noisy data then, because it is based on sound theoretical principles, it may provide better precision than other algorithms currently available.

The general approach of this research is to investigate the theory of locales; to derive new analytical techniques for studying the precision of position estimation; to apply these techniques to some commonly used target forms (dots and edges); to deduce the functional dependence of precision on spatial and intensity resolution; and to find a highly precise, but practical, position estimation algorithm for digital image metrology applications.

1.3 State of the Art

The body of knowledge about high precise position estimation in digital image is recent, sparse and somewhat disassociated. From the digital image metrology literature, three categories are identified, namely those using statistical, empirical, and analytical methods. They are each introduced and discussed in the following subsection. Finally, the emerging field of videometry is discussed at the end of the section. Videometry arises from the recent, rapid development of solid state camera technology

and its application to digital image metrology.

Before proceeding to discuss the state of the art, it is noted that discrete systems theory, which is used extensively in digital image analysis and processing, has not provided techniques specific to the analysis of geometric precision because it is primarily concerned with amplitude representation of images. Errors are expressed in terms of deviations in the amplitudes rather than in the geometric properties of images. Much of discrete systems theory, such as Fourier transform theory for example, considers only spatial sampling and treats intensity quantization as additive Gaussian noise. Thorough and insightful analyses of quantization induced errors are found in [Widro65] and [Bruck87]. They are exemplary of works in discrete systems theory, providing detailed amplitude error analysis but not addressing the problem of geometric error analysis.

1.3.1 Digital image representation and mensuration

The problems of discrete image representation and mensuration have been studied in a variety of ways. Three general approaches are discussed here: statistical approximation, empirical testing and analytical enumeration.

The first approach approximates the discrete nature of the digital image by statistical models. The second obtains performance data for specific systems and attempts to generalize the results. The third method attempts to enumerate the distinct digital representation to provide some estimate of the availability of, or limitation on, geometric precision.

Statistical Approximation. Quantization in digital imagery leads to analytic issues which are different from those of continuous imagery. Techniques which have proven useful for continuous systems are nevertheless applied to the discrete case. The processing of digital data, with its characteristic quantization noise and interpolation errors, is often analyzed in the same fashion as non-digital data by first

incorporating statistical models for the errors and then applying well established statistical techniques, as in [Bruck87], [Ehler82], [Forst82], [Forst87] and [Hill80]. The content of a digital image may be modeled as a stochastic process as well, so that statistical analysis or simulation of the mensuration algorithm can provide estimates or bounds for the error variances, as in [Dvorn83], [Gonsa76], [Ho83] and [Ryan80]. Optimal digitizing can be derived from such statistical models, as in [Bruck87], [Niels84] and [Widro65]. Parametric modeling of specific targets may be used, in which case, positional error is inferred from the statistical properties of the parameter estimates, as in [Akey83], [McGil77], [Tabat84] and [Thurg82b] for example. The various statistical methods allow well known analytic methodologies of continuous systems to be applied to the discrete case. The results further our knowledge of expected performance of mensuration techniques, but they often fail to illuminate the underlying role played by image digitizing.

Empirical Tests. Experimental investigations produce position measurement error estimates for specific digital mensurations systems [Anuta70], [Mikha83], [Mikha85], [Mitch84], [Thurg82a], [Unruh82], [Vosse88]. Some general statements, such as the minimum beneficial pixel size [Unruh82], can be drawn from these studies, but they are not supported by a sufficiently general framework and theoretical foundation to allow their diverse experiential knowledge to be related. Empirical tests are important as a means to obtain practical knowledge in a timely manner but they are not in line with the approach of this study, which is rather to seek out basic principles and new investigative techniques to further the general understanding of precision in digital image metrology.

Analytical Enumeration. The analytical techniques used in digital image representation theory are in considerable contrast to the statistical methods mentioned above. Each digital representation corresponds to a locale and the number of locales provides some measure of the average position estimation precision; with more locales, their average size is smaller and hence position estimation precision is greater. The

enumeration of these representations has been studied for simple binary shapes such as lines, as in [Beren87], [Beren88] and [Dorst84d], but few results exist for general shapes and non-binary images.

As an alternative to enumeration of digital representations, the size of the largest locale might be used as an upper (pessimistic) bound on the position estimation errors due to digitizing, as in [Dorst84d]. This article, which bears the appealing title of "Spirograph Theory", provides a thorough account of both the largest and the average size of locales for binary lines of arbitrary length and slope. Non-binary line segments have been considered in [Klaas75]. Most studies of line segments address binary lines encoded with *chain coding* [Freem69], [Koplo81], [Koplo87], [Neuho85], [Srira89], [Vosse82]. Results specific to binary line segments are not easily extended to other target forms or digitizing schemes because of the difficulty in the theoretical formulation of definitions for various target shapes. Attempts to clearly formulate the character of digital lines [Dorst84d] and circles [Doros84] [Nakam84] illustrate the complexity of the problem.

A formulation of the best linear unbiased estimate of position [Dorst86s] and length [Dorst87] for binary line segments is a specific case of the general principle that the best estimate of object position is given by the centroid of the locale [Havel89]. The concept of locales (although not the terminology) is used in [Beren88] to define *feasibility regions* as areas in the two-dimensional offset-slope parameter space of binary line segments. Locales are implicitly applied to a one-dimensional position parameter space in [Dorst84d].

1.3.2 Videometry: characterizing solid state cameras.

A discussion of the state of the art would not be complete without mentioning the newly emerging area of research which has been referred to as *videometry* [Lenz88b]. Videometry can be defined as the study and characterization of solid state cameras. Thorough and accurate calibration of these cameras is necessary to relate

the image coordinate data obtained by high precision digital image mensuration to real world coordinates. Not surprisingly, camera calibration efforts depend strongly on precise target mensuration techniques. Solid state camera calibration has been studied by [Beyer88], [Curry86], [Dijak85], [Lenz87], [Lenz88a], [Luhma87], [Rosen86] and [Vosse88]. The optical factors in photographic camera calibration are thoroughly treated by existing photogrammetric techniques, which use precise point mensuration within the digital images to determine lens geometry and distortion parameters. Only recently have extensive efforts been directed at solid state camera calibration, although the problem has been long recognized [Sobel74].

Much of the current research in videometry is concerned with the characteristics (idiosyncrasies) of the non-imaging electronic components of the camera and the computer redigitizer, as discussed in [Beyer88], [Dahle87] and [Lenz88b]. Less progress has been made in regard to the imaging array itself, although some investigators, such as [Chamb78] and [Purll83], have addressed this issue. As solid state imaging technology and associated computer image acquisition technology mature, this research area can be expected to progress quickly. At present, the design of commercial image acquisition equipment addresses the requirements for general viewing. More advanced systems, developed for industrial machine vision requirements, provide image processing capabilities which are more suited to image understanding and the analysis of scene context than they are to image metrology. The geometric fidelity of such systems is essentially unquantified.

1.4 Scope and Assumptions

For this study of geometric precision in digital image metrology, only images which contain well defined targets are considered. It is presumed that the characteristics of the imaging system are known so that a complete model of the target in the digital image is available. The image processing tasks of target detection and recognition are presumed successful and it is assumed that an initial target position estimate to

within one pixel is available.

The target models are assumed to be realistic representations of physical targets. Although the physical target may approximate a step-edge (for example), the model is assumed to account for the smoothing effects of the imaging systems, thereby excluding unrealistic components such as delta functions or non-binary step functions from the target model.

The image digitizing scheme is presumed to be completely determined. For simplicity it is often assumed that the sampling raster is a regular square grid and that intensity quantization is performed by integer truncation. Although extensions to more general digitizing schemes are possible, they are not pursued explicitly.

Targets with arbitrary shape and reflectance are considered. The target is presumed to be isolated from confusing background clutter. Partial occlusion of the target is tolerated, with the assumption that the non-occluded region is identified.

Arbitrary target position parameters are considered but the presentation concentrates on the conventional two-dimensional Euclidean position, which is the most common position estimation problem and the most intuitive one.

A lower bound on position estimation errors is sought, rather than an upper bound, because it identifies conditions under which a desired level of precision is not attainable. An upper bound provides no such definitive statement; it purports an attainable level of precision, but external noise may prohibit it. A lower bound of zero is truistic, and the best (tightest) bound is given by the optimal precision. Explicit analytic determination of the optimal precision is not within the scope of this research; rather, as tight a bound as possible is sought within the constraint of it being easily computed for general target forms and digitizing schemes. Other work [Dorst84d] has demonstrated that exact expressions for the optimal precision for a specific target form can involve complex and detailed analysis. Comparably arduous analysis is needed to obtain asymptotic results [Beren88]. On this basis, it seems unreasonable to expect more than an approximate bound to be readily computable

for arbitrary target forms.

Noise is formally excluded from the analysis but the effects of noise on precision is investigated for a few scenarios through simulations. A thorough analysis of the effects of noise is beyond the scope of this research and generalizations based on the reported observations are preliminary in nature.

The position decoding algorithm is used in simulation studies with pseudo-random noise to investigate the effects of noise on precision and to establish empirical estimates of attainable precision for specific target forms. Real images of physical targets were not used because, in order to obtain results which could be related to the theory in a detailed and meaningful way, a highly constrained laboratory environment (controlled illumination and target positioning, for example) would be needed. Simulations provide more convenient and equally effective procedures for investigating the properties of the algorithm.

1.5 Summary of Contributions

- A unifying theory of geometric precision is developed, using the concept of *locales*. Optimal position estimation is rigorously defined and the optimum precision is expressed in terms of parameters of the locale distribution.
- An easily implemented method for computing a lower bound (called the *lineal bound*) on position estimation errors for an arbitrary target shape and digitizing scheme is developed.
- A graphical method for displaying the locale pattern is developed.
- An optimal position estimation algorithm is developed and demonstrated.
- A noise-equivalent target amplitude and size are defined which linearize the empirical position estimation error curves.

1.6 Document Organization

Following this introductory chapter, the essential details of the locales formalism are presented in Chapter 2, including a definition of the target model, a formal definition of locales, a locale map generation technique, and a discussion of the topology of locales.

Chapter 3 introduces the problem of optimal position estimation and the bounds on performance. Precision is expressed in terms of locale geometry and a method of determining the locale density is explained, leading to the definition of the lineal bound on precision. Several examples are discussed to illustrate the principles.

The optimal position estimation algorithm and a near-optimal, robust, efficient variation of it are described in detail in Chapter 4.

Results of the simulation studies using the position decoding algorithm are reported in Chapter 5.

Chapter 6 summarizes the thesis and discusses areas of future research.

The appendices contain various mathematical derivations, technical discussions and the software listings.

Chapter 2

THE LOCALES FORMALISM

2.1 The Target Model

The digital representation of a target is provided by a set of pixel values. The purpose of a model for this representation is to predict the pixel values for a given target position. The *target model* developed in this section is a sequence of transformations and functions describing the imaging process which results in a digital image. The target model is described using the following functions and transformations.

$$f_{\vec{p}}(x, y) \xrightarrow{\Phi_1} \mathcal{O}_{\vec{p}}(x, y) \xrightarrow{\Phi_2} E_{ij}(\vec{p}) \xrightarrow{\Phi_3} \mathcal{I}_{ij}(\vec{p}) \xrightarrow{\Phi_4} I_{\vec{p}}(i, j) . \quad (2.1-1)$$

Each of the functions and transformations in this sequence is introduced below, followed by a more detailed discussion of each of the functions in subsequent sections. The five functions in Equation 2.1-1 above are named as follows:

- $f_{\vec{p}}$ The target brightness function
- $\mathcal{O}_{\vec{p}}$ The optical image function
- E_{ij} The entity function
- $\mathcal{I}_{ij}$ The image function
- $I_{\vec{p}}$ The digital image

When the context of the term *target model* requires a specific function, the image function $\mathcal{I}_i$ will be implied, it being the last one before the digital image. The transformations $\Phi_1, \dots, \Phi_4$ in the target model are each identified with a physical process involved in acquiring the digital image from a solid state camera, as discussed further in Appendix B.1. The transformations are described as follows.

- Φ_1 Optical blurring and distortion
- Φ_2 Photosensitivity and area averaging
- Φ_3 Filtering, resampling and intensity quantization
- Φ_4 Digital storage

Before proceeding to describe each of the function in the target model in more detail, the concept of target position space is introduced.

2.1.1 Target position space: $\mathbf{D}$

A digital image is a two-dimensional representation of the world and target position conventionally refers to the coordinates of the target in a two-dimensional *object plane*. The *target position space*, or *position parameter space*, refers to the region of target positions being considered and is denoted $\mathbf{D}$. The formal presentation here concentrates on the conventional two-dimensional *translational position parameters*, however, other parameters such as target size, orientation, brightness or colour can be used. The choice of parameters is constrained only by the fact that they must be continuous. The number of sides of a polygon target, for example, would not make a suitable position parameter since it is discrete.

Target position parameters other than translational position are examined in examples and extensions to the main development but, for clarity, the formal theory does not attempt to pursue them rigorously. A typical target position space $\mathbf{D}$ is the unit square in a two-dimensional translational position space. (The term *space*

is used loosely, sometimes referring to a bounded region such as the unit square and sometimes referring to the linear vector space generated by the position parameters.)

When generality is invited, or emphasized, the target position parameters are denoted by a vector $\vec{p}$. When two-dimensional translational position parameters are being emphasized, target position coordinates such as (x_0, y_0) are used.

2.1.2 The brightness function: f

Target modelling begins with a brightness function which describes the amount of light entering the imaging system from each point on the target. The brightness function, denoted $f_{\vec{p}}(x, y)$, is parametrized by the target position parameters $\vec{p}$.

Some typical target shapes are shown in Figure 2.1. Targets are often made to have high contrast, resulting in a brightness function which has large step discontinuities. It is assumed that the target is separated from the background clutter and that the target surround has a brightness of zero.

2.1.3 The optical image function: $\mathcal{O}$

The target is imaged onto an *image plane* within the camera by the lens system. As depicted in Figure 2.2, there is an *object plane* at the object which is conjugate to the image plane in the camera.

The optical image function $\mathcal{O}_{\vec{p}}(x, y)$ represents the incident brightness at a point (x, y) on the image plane when the target has position parameters $\vec{p}$. It incorporates optical characteristics such as blurring (Figure 2.3) and distortion (Figure 2.4).

An ideal point source brightness function, defined as a Dirac delta function, imaged through an ideal lens will produce an *Airy disk* [Smith66, s.6.9] at the image plane. An Airy disk is a diffraction pattern which has a bright, circular central region. A rule-of-thumb estimate for the diameter of the central region (in μm) is the f -number of the lens. A typical f -number for a CCD camera lens is between $f/1.4$ and $f/11$.

FIGURE 2.1 Six examples of targets for high precision position estimation. The Gaussian dot (a) and the binary edge targets (f) are examined in the simulation studies of Chapter 5 and are discussed as examples of the formalism. Binary targets (b) – (f) are particularly easily generated and analyzed. The Reseau cross (d) is designed to be scale invariant and the annular target (e) is rotation invariant. Edges are important, not only because many different targets shapes can be constructed from them, but also because they are common in natural scenes.

A real lens, even when it is perfectly focused, produces more blurring than the diffraction-limited Airy disk, so it is reasonable to assume that the smallest point or the sharpest edge in the image plane will extend over more than $10\ \mu\text{m}$. Since the sensor size in a CCD imaging array is about $10\ \mu\text{m}$ wide, the optical image function will be smooth at the scale of the imaging sensor, even if the brightness function has large step discontinuities.

The surface of the solid state imaging array lies in the image plane and the distance between the centres of its sensors defines the pixel size there. The pixel size is used as a unit of length in either the image plane, the object plane or the digital image. This interchangeability of the unit pixel interval avoids the confusing scale conversions needed when real-world coordinates are used.

Consider, for example, a 25 mm lens imaging a target 10 meters from the lens onto an imaging array with $10\ \mu\text{m}$ square sensor elements. In the image plane, each pixel

FIGURE 2.2 The lens of an image acquisition system has two *conjugate planes*. One is where the object lies (the *object plane* on the left), the other is where the image is formed (the *image plane* on the right). The planes are parallel to each other and perpendicular to the optical axis. The distances d_1 and d_2 from the optical centre to the object plane and image plane, respectively, are related by Snell's law: $\frac{1}{d_1} + \frac{1}{d_2} = \frac{1}{f}$, where f is the focal length of the lens. The width of a pixel in the image plane is $\frac{d_2}{d_1}$ times the width of a pixel in the object plane.

FIGURE 2.3 The point spread function of a lens is a two-dimensional function which describes how the image of an ideal point (delta function) in the object plane is blurred due to the finite aperture size, non-ideal lens geometry and optical imperfections. An ideal lens exhibits a point spread function which is a diffraction pattern called an *Airy disk*.

Based on [Hecht79, Fig. 11.9]

FIGURE 2.4 A square grid pattern (a) imaged through a typical lens will exhibit either *barrel distortion* (b) or *pincushioning* (c). These effects are a type of radial lens distortion

From [Jenki50, Fig 9T]

is the size of the sensor, namely $10\ \mu\text{m}$ wide, whereas in the object plane, each pixel is approximately 4 mm wide.

2.1.4 The entity function: E

The entity function $E_{ij}(x_0, y_0)$ combines the aperture sensitivity of individual sensors with the optical image function to describe the output of individual sensors as a function of the target position parameters. Sensor response characteristics such as, surface transmittance, lateral diffusion, aperture shape and quantum efficiency are incorporated in the entity function [Chamb78]. (See Figure 2.5.)

The entity function can be greatly simplified by assuming that it is *translation invariant*, meaning that translating the target is equivalent to translating the sensors. A formal definition of translation invariance follows.

Translation Invariance Let $h_{\vec{p}}(\vec{x})$ be a function which is parametrized by $\vec{p}$. The function h is said to be *translation invariant* if there exists some parameter transformation g which maps the parameters $\vec{p}$ into the domain of h and some *reference function* h_0 such that $h_{\vec{p}}(\vec{x}) = h_0(\vec{x} - g(\vec{p}))$.

Rigorous details such as closure of the domain of h and continuity issues are

FIGURE 2.5 The response of an individual photo-sensor varies across its photo-receptive area. Among the factors which affect the response profile are the transmittance of the sensor surface and the sensitivity of the substrate. Transmittance is determined by the geometry of the electrodes which are placed over the photo-sensitive substrate to generate the potential wells, or *charge buckets* which hold the photo-emitted charge.

omitted to avoid obscuring the basic idea. The principle of translation invariance, simply stated, treats the target position and the pixel indices as vectors which can be combined (as $\vec{x} - g(\vec{p})$) so that only one reference function is needed for the target representation. The reference function can be defined as $h_{\vec{p}}$ with $\vec{p} = \vec{0}$ (the zero vector or parameter origin).

A special class of target models is defined now.

Translational Target Model A *translational target model* is one which has the conventional two-dimensional translational position parameters (x_0, y_0) and a translation invariant entity function satisfying

$$E_{ij}(x_0, y_0) = E_0(x_0 - x_{ij}, y_0 - y_{ij}). \quad (2.1-2)$$

Furthermore, it is usually assumed that $(x_{ij}, y_{ij}) = (i, j)$, so that the sampling raster is a regular unit-spaced square grid.

Notice that the entity function of a translational model can be thought of as a function of target position (x_0, y_0) parametrized by the sensor indices (i, j) or vice versa.

The word *entity* is used to describe E_i , because of its connotation of the “thing”, “form” or “pattern” of interest. It is the starting point of most discussions in digital image processing. The entity function is formulated here as being derived from the brightness and optical image function, rather than being axiomatically defined, to underscore the importance of effects such as smoothing which must be incorporated into it prior to digitizing in order to obtain a realistic target representation.

2.1.5 The image function: $\mathcal{I}$

The image function $\mathcal{I}_{ij}(\vec{x}_0)$ is a discrete valued function of the target position $\vec{x}_0 \in \mathbf{D}$. Like the entity function, the image function is parametrized by the pixel indices. The image function incorporates the filtering, resampling and quantization operations of the electronics which convert the photo-sensor output to digital data. (Resampling may cause the pixel indices of the image function to differ from those of the entity function but, to simplify the notation, this detail is ignored.)

A quantization function $\mathcal{Q}$ is a piecewise constant monotonic function which maps continuous (intensity) values to (non-negative) integer values. Unless otherwise specified, quantization is presumed to be integer truncation. The notation $\lfloor x \rfloor$ is used to denote quantization of a value x ; that is, $\mathcal{Q} : x \mapsto \lfloor x \rfloor$. The quantization function has a finite set of quantized values $\mathbf{Q} = \{0, 1, \dots, Q-1\}$, where $Q = |\mathbf{Q}|$ is the number of elements in $\mathbf{Q}$.

In the case of a translational target model, with no filtering or resampling, the image function of a target positioned at $(x_0, y_0) \in \mathbf{D}$ is defined as

$$\begin{aligned} \mathcal{I}_{ij}(x_0, y_0) &= \mathcal{I}_0(x_0 - x_{ij}, y_0 - y_{ij}), \\ &= \lfloor E_0(x_0 - x_{ij}, y_0 - y_{ij}) \rfloor. \end{aligned} \quad (2.1-3)$$

2.1.6 The digital image: I

A *digital image*, or *digital representation*, $I_{(x_0, y_0)}$ of a target which is modelled by an image function $\mathcal{I}_{ij}(x_0, y_0)$ will mean a (small) set of pixel values obtained by sampling the image function at (x_0, y_0) for K distinct pixel indices (i_k, j_k) , with $1 \leq k \leq K$. There is no requirement that the selected pixels form a rectangular array, as in a conventional digital image. For our purposes a digital image may be defined, for example, by the pixels around the perimeter of a circular target or along the boundary of an edge.

The digital image I is formally defined as a set of pixel values, parametrized by the target position (x_0, y_0) ;

$$I_{(x_0, y_0)} = \{ p_{ij} \in \mathbf{Q} : p_{ij} = \mathcal{I}_{ij}(x_0, y_0), (i, j) \in \mathbf{W} \}, \quad (2.1-4)$$

where $\mathbf{W}$ is the *sampling window* defined further below.

It is convenient to assume that the pixel indices are ordered in some way, in which case the pixel values are written as p_k with

$$p_k = \mathcal{I}_{(i_k, j_k)}(x_0, y_0) \quad 1 \leq k \leq K. \quad (2.1-5)$$

The choice of the order for the pixel indices is not important but it is presumed to be fixed.

2.2 Definition of Locales

2.2.1 The sampling window: W

A sampling window for a digital image is a set of pixels indices $\{(i, j)\}$. The window is typically defined to provide a square or rectangular area of pixels which cover the image of the target; however, other window choices are possible and may be more useful. For example, a thin annular target may have a sampling window which covers only the perimeter of the target. A partially occluded target may have a

sampling window which covers only the non-occluded portion of the target. The position decoding algorithm introduced in Chapter 4 accepts an arbitrary sampling window.

A sampling window which is large enough to cover all of the target, no matter where the target is positioned within the position space $\mathbf{D}$, is called a *sufficient* sampling window. The number of elements in a sufficient window depends on the size of the target position space as well as the size of the target. A formal definition of a sufficient sampling window follows, stated in a way which will be useful later.

Sufficient Sampling Window Let χ_{ij} be the set of all points $\vec{x}$ in the target position space $\mathbf{D}$ which yield a non-zero value for the image function; that is,

$$\chi_{ij} = \{ \vec{x} \in \mathbf{D} : I_{ij}(\vec{x}) \neq 0 \}.$$

A sampling window $\mathbf{W}$ is *sufficient* if it contains the set $\mathbf{W}_0$, where

$$\mathbf{W}_0 = \{ (i, j) : \chi_{ij} \text{ is not empty} \}.$$

The set $\mathbf{W}_0$ is the *minimal sufficient* sampling window for the position space $\mathbf{D}$.

If $\mathbf{W}$ is a sufficient sampling window and (i, j) is not in $\mathbf{W}$ then $I_{ij}(\vec{x})=0$ for all $\vec{x} \in \mathbf{D}$. If $\mathbf{W}$ is the minimal sufficient sampling window and (i, j) is in $\mathbf{W}$ then $I_{ij}(\vec{x}) \neq 0$ for some $\vec{x} \in \mathbf{D}$.

2.2.2 Image space: $\mathbf{Q}^K$

If the pixel indices of the sampling window are ordered then the set of pixel values $\{p_k : 1 \leq k \leq K\}$ of the digital image corresponds to a vector (or point) $\vec{q}$ in the K -dimensional space $\mathbf{Q}^K$, where $\mathbf{Q}$ is the set of quantization values in the range of the quantization operator $\mathcal{Q}$. The vector $\vec{q}$ is defined as

$$\vec{q} = (p_1, p_2, \dots, p_K). \quad (2.2-1)$$

and the K -dimensional space $\mathbf{Q}^K$ is called the *image space*. Each coordinate of image space has $Q = |\mathbf{Q}|$ discrete points on it and so there are Q^K points in the entire image space.

The distance between two points, $\vec{q}$ and $\vec{q}'$, in image space can be defined by a norm such as the Hamming distance (or ℓ_1 -norm),

$$\|\vec{q} - \vec{q}'\|_1 = \sum_{i=0}^K |q_i - q'_i| ,$$

or the Euclidean distance (or ℓ_2 -norm),

$$\|\vec{q} - \vec{q}'\|_2 = \left(\sum_{i=1}^K (q_i - q'_i)^2 \right)^{\frac{1}{2}} .$$

Unless otherwise specified, the Hamming distance will be used as the distance measure between points in image space.

Note that many points in image space are not digital images of the target, if fact, relatively few points in $\mathbf{Q}^K$ will have coordinates which correspond to pixels of some digital representation of the target.

2.2.3 Locales: $\mathcal{L}$

Each target position $\vec{x} = (x_0, y_0)$ in the position space $\mathbf{D}$ results in a digital image $\mathbf{I}_{\vec{x}}$ which corresponds to some point $\vec{q}$ in image space. This correspondence can be expressed as a mapping Π from $\mathbf{D}$ into $\mathbf{Q}^K$,

$$\Pi : \mathbf{D} \rightarrow \mathbf{Q}^K ,$$

with

$$\Pi(\vec{x}) = \vec{q} = (p_1, \dots, p_K) . \quad (2.2.2)$$

where p_k are the pixel values defined by Equation 2.1-5 and $\vec{x} \in \mathbf{D}$.

This mapping is many-to-one and so it induces an equivalence relation ($\equiv$) in its domain $\mathbf{D}$. The following definition is central to the locales formalism.

Position Equivalence Classes Position equivalence classes are induced in the target position space by the mapping Π from (continuous) position space $\mathbf{D}$ to discrete image space $\mathbf{Q}^K$. Two points $\vec{x}_1$ and $\vec{x}_2$ in $\mathbf{D}$ are equivalent if and only if their digital images are identical; that is to say,

$$\begin{aligned}\vec{x}_1 \equiv \vec{x}_2 &\iff \|\Pi(\vec{x}_1) - \Pi(\vec{x}_2)\|_1 = 0, \\ &\iff \mathcal{I}_{ij}(\vec{x}_1) = \mathcal{I}_{ij}(\vec{x}_2) \quad \forall (i, j) \in \mathbf{Q}.\end{aligned}\tag{2.2-3}$$

This equivalence relation partitions the target position space into regions called *locales*. Target positions within a locale are completely indistinguishable with respect to the digital image of the target. This indistinguishability determines the limitations on position estimation precision imposed by the discrete nature of digital images.

The equivalence class of a point $\vec{x} \in \mathbf{D}$ is denoted $[\vec{x}]_\Pi$ ($\vec{x}$ modulo Π ; see [Bobro74]). The set of all locales is denoted $\mathcal{L} = \mathbf{D}/\Pi$. The symbol $\mathcal{L}$ is also used to denote the set of points in image space which are digital representations of the target, and so, a locale $\ell \in \mathcal{L}$ can refer to either a region $[\vec{x}]_\Pi$ in position space or a point $\vec{q} = (p_1, \dots, p_K)$ in image space. The dual notation, $\mathcal{L} \subset \mathbf{D}/\Pi$ and $\mathcal{L} \subset \mathbf{Q}^K$, emphasizes the close relationship between image space and position space.

Locales Each point in image space which corresponds to a digital representation of the target is called a *locale*. The set of all locales in image space is denoted $\mathcal{L}$. For each locale $\ell \in \mathcal{L}$ in image space there corresponds a unique equivalence class in position space. These equivalence classes are also called locales (and may be distinguished by specifying "in position space"). The symbol ℓ is reserved to represent a locale in either position space or image space.

Every point in position space belongs to some locale, hence the locales partition (or tessellate) position space. Not every point in image space is a locale, in fact, locales may be sparse in image space.

2.2.4 Periodicity of locales

In Appendix A.2 it is proven that for a translational target model with a sufficient sampling window, the locales partition the position space $\mathbf{D}$ in a periodic manner and that the period is the same as the pixel spacing. This regularity in the locale pattern makes it natural to use the unit pixel as the standard target position space.

Periodicity of the locale pattern makes the term *tessellation* particularly appropriate for the way the target position space is partitioned into locales. In the next section, a graphical method is presented for displaying the locale tessellation pattern.

2.2.5 Example locales

Consider the Gaussian dot model for a point target, which was introduced in Section 1.1.4. The image function is defined as the quantized Gaussian

$$\mathcal{I}_0(x, y) = \lfloor A e^{-(x^2+y^2)/(2\sigma^2)} \rfloor, \quad (2.2-4)$$

where $\lfloor \cdot \rfloor$ indicates quantization. For the target positioned at (x_0, y_0) , the pixel values p_{ij} are

$$p_{ij} = \mathcal{I}_0(x_0 - i, y_0 - j). \quad (2.2-5)$$

A binary image of the dot is obtained by defining a quantization threshold t_b , above which the pixel values are set to one (*on*) and below which the pixel values are set to zero (*off*). All pixels whose centres fall within a circle of radius r_b of the target position (x_0, y_0) will be *on*, and all others will be *off*, where r_b is given by

$$r_b^2 = 2\sigma^2 \ln \frac{A}{t_b}. \quad (2.2-6)$$

Figure 2.6 depicts the Gaussian dot model with a 3×3 sampling window and $r_b \approx 1.3$. The nine dots in the figure represent the centres of the pixels. If the origin is at the centre of the middle pixel and the target position is at the origin, then the *on* pixels form a cross pattern, as shown by the shaded pixels in the figure. Other pixel patterns which can be generated by the Gaussian dot are represented in Figure 2.7.

FIGURE 2.6 The Gaussian entity function $E_0(x, y) = Ae^{-(x^2+y^2)/(2\sigma^2)}$, represents a small dot target placed in the object plane. It is depicted at the left with a shaded level to indicate a threshold for binary quantization. At the right, the Gaussian dot is represented by a circular contour which is superimposed on the pixel pattern of its binary image. Pixels with their centres within the contour are shaded to indicate that they are above the binary quantization threshold.

FIGURE 2.7 The pixel patterns for six different positions of the binary Gaussian target are shown. The cross in each pattern represents the position of the target. The six patterns, together with their reflections and rotations (25 patterns in all) are the only binary representations possible.

If the target position space D is defined to be the unit square centred at the origin,

$$D = \left\{ (x_0, y_0) : |x_0| \leq \frac{1}{2} \text{ and } |y_0| \leq \frac{1}{2} \right\}. \quad (2.2-7)$$

then the six pattern shown in Figure 2.7, together with their rotations and reflections, are the only possible binary images of the dot. Notice that the sampling window is sufficient for D . In all, there are 25 binary representation for the dot within the unit square position space. These 25 representations form the set of locales $\mathcal{L}$ for the target in D .

The image space for this binary digitizing scheme, is $\mathbf{Q}^K$ with $\mathbf{Q}=\{0,1\}$ and $K=9$. The 25 locales are a small portion of the total size of image space, which has 512 points in it.

Each of the 25 pixel patterns is associated with a locale in D , as depicted in Figure 2.8, where the regions of position space which correspond to each of the six patterns of Figure 2.7 are labelled. Notice the symmetry in the locale pattern. If opposite edges of the unit square are joined, the locale boundaries align perfectly.

Locales on the perimeter of the unit square appear to extend beyond the bounds of the position space, indeed, if the area of the position space is increased, the locale pattern will grow outward by symmetric extension. In order to define the locale regions with some independence from the choice of the position space, locales are considered to extend in a periodic manner beyond the position space (when the sampling window is sufficient), as illustrated in Figure 2.9.

In the next section, a method will be described for drawing the locale pattern for an arbitrary target shape, position space and sampling window.

2.3 Generating Locale Maps

A locale map is a graphical representation of the partitioning of target position space into position equivalence classes. Each region of the locale map is identified with a particular digital representation of the target.

FIGURE 2.8 The regions of target position which correspond to specific binary representation of the Gaussian dot are labelled using the numbers applied to the binary pixel patterns of the previous figure. Each locale in the unit square can be associated with one of the six types of pixel patterns shown in Figure 2.7, with rotation or reflection symmetry

FIGURE 2.9 The locale pattern within the target position space (a square position space is indicated here by a thick box) can be considered to extend periodically beyond the bounds of the position space. Periodicity is with respect to the sampling raster. Note that the locales need not be symmetric within the position space

Labelling each region of the map to indicate which digital representation it corresponds to can be very cumbersome, so locale maps are discussed without reference to the implied labelling scheme. This simplification assumes that each region of the tessellation pattern belongs to a different locale; that is, locales are not comprised of disjoint regions. It will be shown later that disjoint locales are possible but unlikely.

The locale map provides a visual impression of the position estimation uncertainty due to spatial and intensity quantization. Larger regions in a locale map correspond to larger position uncertainty and the distribution of locale sizes provides a visual impression of the quality of available position information.

For example, Figure 2.10 shows a binary image function representing a cross shaped target aligned with the sampling raster. The image function of the target is shown on the left and the locale map for a 5×5 sampling window is shown on the right. By modifying the target slightly, the locale pattern changes dramatically, as illustrated in Figure 2.11 where the size of the target is reduced slightly. Although a smaller target is normally associated with greater position uncertainty, it is seen in this case that a much finer meshwork is obtained, indicating reduced position uncertainty.

In Chapter 3 an analytical method of quantifying position estimation uncertainty from the locale map will be discussed, however, the emphasis here is on a method to obtain a graphical representation of the tessellation pattern of the locales for visual interpretation. The method described is very simple and provides insight into the properties of the locale patterns.

2.3.1 Superpositioned contours

A contour representation of the image function is used to generate the locale map. As an example, the Gaussian dot (Section 1.1.4) with 5 quantization levels is depicted in Figure 2.12a. To generate the locale map, the contour pattern is replicated at offsets corresponding to the centre of each pixel in the sampling window.

FIGURE 2.10 A contour representing the binary image function of a cross is shown at the left. The thick contour line is superimposed on a 5x5 grid of pixels. Using the pixels in this grid to define a sampling window, the locale map for the cross is generated as shown at the right. Notice that the locale map contains large locales because many of the contours are almost coincident with each other. Compare this locale map with the one in the next figure.

FIGURE 2.11 Just as in the previous figure, a contour representing a binary cross is shown at the left and its locale map is shown at the right. The cross in this figure is slightly smaller than the one in the previous figure and the effect is to provide a much finer pattern in the locale map. Viewing the locale map provides a visual impression of the distribution of locale sizes. This locale map indicates higher precision than does the locale map in the previous figure.

In Figure 2.12b, the replicated contour pattern is shown displaced one unit to the right. This displaced-replication is repeated for every pixel in the sampling window, as illustrated in Figure 2.12c. The resultant mesh of contours partitions the position space into locales.

To see that this locale map generation method works correctly, consider that the contours in Figure 2.12a indicate the target positions (x, y) where the pixel value p_0 , associated with $i=0$ and $j=0$, will change its value. Similarly, the contour pattern displaced by $(-i, -j)$ indicates where the pixel value $p_{ij} = I_0(x - i, y - j)$ changes value. The superposition of all such contours show where *any* pixel value changes, and hence the mesh of contours delineate the locales. The inflection about the origin, indicated by the displacements $(-i, -j)$, can be disregarded for radially symmetric targets such as the Gaussian dot.

2.3.2 Periodicity in the locale map

From the construction technique for locale maps, it is obvious that a sampling window containing an infinite grid of pixels will produce a periodic pattern of locales. A finite regular grid will generate a locale pattern which is periodic in its central region but not on its periphery. If the target position space is within the central part of the pattern, as it is when the sampling window is sufficient, then periodicity in the locale map is assured. As the position space is enlarged with respect to the sampling window, non-periodic portions of the pattern enter into the locale map.

2.3.3 Unconventional locale maps

The technique for generating locale maps does not rely on a regularly spaced sampling raster. If $\{p_{ij}\}$ is a set of pixel values obtained by arbitrary spatial sampling, so that each pixel p_{ij} has a corresponding pixel centre (x_{ij}, y_{ij}) , then the locale map is generated by replicating the contours at offsets $(-x_{ij}, -y_{ij})$.

FIGURE 2.12 Five levels of the Gaussian dot target model are represented by the thick concentric contours in (a). The superimposed 3x3 grid of pixels represents the sampling window. The central pixel is the target position space. The pattern of contours is shown replicated at an adjacent pixel centre in (b). Replicating the contour pattern at each pixel in the sampling window, as shown in (c), partitions the position space into regions. The resulting pattern is called a *locale map*, each region in the map belongs to a locale. Using this simple construction technique, a locale map can be generated for any shape of target with any number of gray levels and any sampling window.

Translation invariance is not essential in order to generate a locale map since the locale map can also be generated by superimposing the contours of the individual image functions I_{ij} , without any offset. This principle will be demonstrated in the next section, where a locale map is generated for the important example of a straight binary edge.

As a further generalization of the locale map generation procedure, consider the case of a three-dimensional target position space with coordinates (x, y, z) . The contours of the image function $I_{ij}(x, y, z)$ are surfaces in the three-dimensional target position space and, as before, these surfaces indicate target positions at which the pixel values change. The locale map in this case is simply the union of these surfaces for all pixel indices (i, j) in the sampling window. Due to their notational and computational complexity, position spaces with more than two-dimensions are not considered further.

2.3.4 Locale maps for straight edges

In this section the locale map for a long straight-edge target is constructed. Figure 2.13 depicts a binary edge, indicating the offset and orientation parameters b and θ . The edge orientation θ is specified in a counterclockwise direction, with $\theta=0$ when the edge is horizontal with the dark side of the edge below the bright side. The edge offset b is the (signed) perpendicular distance between the edge and the origin. The offset is positive when the origin is on the bright side of the edge.

The entity function for an edge is determined by an *edge profile* function $h(z)$, which is evaluated at the perpendicular distance $z=(b-i \sin \theta+j \cos \theta)$ from the edge to the centre of a pixel (i, j) :

$$E_{ij}(b, \theta) = h(b - i \sin \theta + j \cos \theta) . \quad (2.3.1)$$

The edge profile function is assumed to increase monotonically from zero to a maximum value h_0 .

FIGURE 2.13 The boundary of an edge is depicted here as a (directed) line. The edge is considered to extend indefinitely in both directions. The dark side of the edge is on the right of the line, the light side is on the left. The location of the edge in the plane is determined by the angle θ of the line counterclockwise from the horizontal, and the signed offset b from the origin to the edge. The offset is negative if the origin is on the dark side of the edge, positive otherwise.

A binary straight edge is obtained by applying a binary quantizer $\lfloor \cdot \rfloor$ to the edge profile function to obtain a unit step function. Let h_t be the quantization threshold, and assume that the edge profile function is defined with $0 < h(0) = h_t < h_0$, then

$$U(z) = \lfloor 1 + (h(z) - h_t)/h_0 \rfloor = \begin{cases} 1, & \text{if } z > 0, \\ 0, & \text{if } z \leq 0. \end{cases} \quad (2.3-2)$$

The image function for this binary edge is

$$\mathcal{I}_{i,j}(b, \theta) = U(b - i \sin \theta + j \cos \theta). \quad (2.3-3)$$

In (b, θ) -position space the image function for a pixel is a sinusoidal contour of amplitude $(i^2 + j^2)^{1/2}$ and phase $\arctan(j/i)$.

One-dimensional Locale Map. If θ is fixed, so that the offset b is the only position parameter, then the image function has the form

$$\mathcal{I}_{i,j} = U(b - b_{i,j}) \quad \text{with} \quad b_{i,j} = (i \sin \theta - j \cos \theta). \quad (2.3-4)$$

Let the target position space for this one-dimensional case be the unit interval $0 \leq b \leq 1$. An interesting discussion of the locale map for this case, which is not illustrated here, can be found in [Dorst84d].

Two-dimensional Locale Map. When both θ and b are treated as unknowns, let the target position space be defined as

$$D = \{ (b, \theta) : |b| \leq \frac{1}{2} \text{ and } 0 \leq \theta \leq 2\pi \}. \quad (2.3-5)$$

This position space includes all edges which intersect the unit-diameter circle at the origin. Using the sampling window

$$W = \{ (i, j) : |i| \leq 2 \text{ and } |j| \leq 2 \}, \quad (2.3-6)$$

the locale map is generated by superimposing the 25 sinusoidal contours of the image function. The resulting mesh of locales is shown in Figure 2.14. The target position space is indicated in the figure by a shaded region. The locale pattern is not periodic because (a) the sampling window cannot be sufficient for an edge that is assumed to be infinitely long, and (b) the image function is not translation invariant. Periodicity in θ is an incidental consequence of the cyclic nature of that parameter.

2.4 The Topology of Locales

Topology is the study of geometrical properties and spatial relations unaffected by continuous change of shape or size. In this section, the topology of locales is considered in both continuous position space and discrete image space. With apologies to the elegant subject of general topology, the discussion will proceed on the basis of intuitive topological principles, primarily: connectedness, adjacency and convexity. There is much that is yet unknown about the topology of locales and this section provides only a few insights into this interesting subject.

The section begins with a discussion of locale connectedness. It is shown by an unrealistic example that locales need not be connected; however, because the entity

FIGURE 2.14 The two parameter position space for a binary edge is partitioned into locales by superimposing scaled and translated versions of the sinusoidal image function (Equation 2.3-3). The locale map for a binary edge with a 5×5 sampling window is shown here. The amplitude of the sinusoid for pixel (i, j) is $b_{ij} = (i^2 + j^2)^{1/2}$ and the phase is $\theta_{ij} = \arctan(j/i)$.

function must be smooth, it is argued that connectedness is a reasonable assumption. Convexity of locales is also argued on the basis of the smoothness of the entity function.

Locale adjacency is discussed to resolve the question of whether adjacency in image space implies adjacency in position space and vice versa. Contrived counter-examples for both conclusions are presented, however, based on the absence of a more realistic counter example, it is conjectured that locale adjacency is usually preserved between the two spaces.

During the study of the position decoding algorithm (described in Chapter 4) it was observed that locales which are close together in image space need not be close together in position space. This effect is discussed with a simple illustration based on the binary Gaussian dot target.

2.4.1 Connectedness

A locale in position space is said to be *connected* if every pair of points in the locale has an unbroken path which joints them and lies entirely within the locale. A disconnected locale is composed of two or more regions which do not share a boundary in the locale map. In image space, where the coordinate system is discrete and locales are points, the definition of connectedness is vacuous.

Figure 2.15 is an illustration of disconnected (or disjoint) locales. This unrealistic image function is shaped like a X and is slightly larger than one pixel. The locale map, with a 5×5 sampling window, is shown on the right of the figure. The shaded locale, which is composed of four disconnected regions, corresponds to the digital image with all pixel values equal to zero except the central one.

The locale corresponding to the digital image with all pixel values equal to zero is also disconnected. This locale is composed of all the regions which make up the apparent background of the locale map.

Another simple example of disconnected locales is illustrated in Figure 2.16. This example uses the concentric circular contours of the Gaussian dot (as in Figure 2.12) with a non-sufficient sampling window consisting of the two pixels $(0,0)$ and $(0,-1)$.

Notwithstanding these demonstrations of disjoint locales, it is conjectured that for realistic image functions and sufficient sampling windows a locale map has few (if any) disconnected locales. This conjecture is based on the principle that the smooth image function is monotonic at the scale of a pixel interval. In order to have a disconnected locale, the image function of some pixel must increase (or decrease) and then return to its original value across the gap between two regions of the locale.

The apex of the image function is, of course, a non-monotonic region of the image function. It is assumed that this region is large enough that other pixels will change value (monotonically) within it, thereby prohibiting disconnected locales. The above illustration of disconnected locales is unrealistic because of the size and abruptness of the shape of the image function.

FIGURE 2.15 The binary image function shown at the upper left has disconnected locales. The locale map is shown at the right. The shaded regions in the map form one disconnected locale which corresponds to the 5x5 binary image described by the array at the lower left.

FIGURE 2.16 The five-level Gaussian dot (Figure 2.12a) can have disconnected locales if the sampling window is not sufficient. The locale map for a sampling window of only two pixels is shown here. The shaded pair of regions comprise a disconnected locale.

2.4.2 Convexity of locales

Vertices in the locale map are formed by the intersection of smooth lines and so all internal angles of locales must be less than 180 degrees. If all locale boundaries are straight lines then all locales are convex polygons. Conversely, if all locales are convex, then the boundaries between them would have to be straight lines.

The smoothing effects of the image acquisition system ensure that the contours of the image function have limited curvature. For dense locale maps, with small locales, the boundaries approximate straight lines and so their curvature will be ignored.

2.4.3 Adjacency

In position space, two locales are said to be adjacent if they share a common edge in the locale map. For example, the four locales which are adjacent to the central locale, in the locale map of the binary Gaussian dot, are shown lightly shaded in Figure 2.17.

In image space, two locales ℓ_1 and ℓ_2 are said to be adjacent if and only if their Hamming distance $\|\ell_1 - \ell_2\|_1$ is one. In Figure 2.17, the digital representations for the central locale and the four surrounding locales are shown inset. Refer to Figures 2.7 and 2.8 for an explanation of the representations. In this example, adjacency in position space corresponds to adjacency in image space since the four surrounding locales each differ from the central locale by one pixel value. In each of the inset digital representations, the pixel which differs in value, shown darkly shaded.

Locale boundaries identify where pixels change value and a boundary between adjacent locales usually corresponds to the value of only one pixel changing by ± 1 , which ensures that the locales are adjacent in image space. However, it is possible for the boundary between adjacent locales to correspond to an image distance of more than one.

Consider the example in Figure 2.18 where many contours of the image function overlap in the locale map. When k contours overlap on a locale boundary, the boundary is said to have *multiplicity* k . Multiplicities greater than 1 do not usually occur unless the image function has a regular geometric shape with straight edges. The Hamming distance between two locales across such a boundary is k . Notice that a boundary of multiplicity k must correspond to k pixel values each changing by ± 1 since the smoothness of the entity function ensure that no single pixel value can jump by more than 1.

Locales which are adjacent in image space are usually adjacent in position space but it is possible to construct a counter-example. In Figure 2.19, an image function is defined which consists of 3 circular contours arranged in an L pattern. When the contours of the image function are superimposed to form a locale map for a 3×3

FIGURE 2.17 Locales are *position-adjacent* if they share an edge in position space and they are *image-adjacent* if the Hamming distance between them is 1. In the binary Gaussian dot example of Figure 2.8, the four locales (of type-4) which are position-adjacent to the central locale (of type-1) are seen to be image-adjacent as well. The locale map is shown at the centre of the figure, with the centre locale striped and the four adjacent locales shaded. For each of the shaded locales a representation of its binary image is inset beside it. For each one, the pixel which differs from that of the central locale is darkly shaded. The orientation of each type-4 locale is indicated by the rotation of its numeric label.

FIGURE 2.18 Contours can overlap in the locale map to produce boundary multiplicity. Here, a 5×5 sampling window and a rectangular binary image function with rounded corners results in a locale map with boundary multiplicities of 1, 2, 3, 4 and 6.

sampling window, as on the right side of the figure, circular contours from different pixels become concentric. This pattern causes adjacent locales in image space to be non-adjacent in position space, as in the case of the two shaded locales which represent the digital images

$$\ell_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \ell_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (2.4-1)$$

Although examples have been constructed to illustrate that adjacency in either position space or image space does not imply adjacency in the other, it is nevertheless a plausible conjecture that *most* pairs of locales which are adjacent in one space are also adjacent in the other. The conjecture is based on the difficulty in devising a realistic counter-example and it implies that the planar structure of the locales in position space is topologically preserved in image space.

In Appendix A.3.2 it is shown that the average number of locales adjacent to each locale in position space is 4. Assuming that locales are adjacent in position space if and only if they are adjacent in image space, this evokes the impression of locales as a loose network in image space with each node of the network having four branches. In an image space of dimensionality $K \gg 4$, this network will be very tenuous. Figure 2.20 depicts a network of locales in image space (albeit crudely in two-dimensions).

FIGURE 2.19 It is possible that two image-adjacent locale are not position-adjacent (but the converse is not true). A binary image function is shown at the upper left, consisting of three separate circular regions. It has a value 1 inside each region and zero elsewhere. The locale map for a 3×3 sampling window is shown at the right. Shading and stripes are used to identify a pair of locales which are image-adjacent but not position-adjacent. Their representations in image space are shown at the lower left, where it can be seen that they have a Hamming distance of 1. Notice that the shaded locale is disconnected.

2.4.4 Scattered locales

Define the set of all locales within a convex volume of image space to be a *locale cluster*. It is possible for a locale cluster to be a scattered set of locales in position space. To illustrate this surprising fact, consider the locale map for the binary Gaussian dot (Section 2.2.5). Select the non-locale point $\vec{q}$ from image space with

$$q = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \quad (2.4-2)$$

and define the locale cluster to be all locales ℓ with $\|\vec{q} - \ell\|_1 \leq 2$. This cluster consists of locales which are rotations and reflections of types 1 and 5, as per the labelling

FIGURE 2.20 Each locale has several other locales adjacent to it, resulting in a network of locales in the multi-dimensional image space. This network is represented two-dimensionally here.

scheme of Figure 2.7, with

$$\text{type-1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{type-2} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.4.3)$$

Figure 2.21 shows this cluster in position space. This example of a scattered cluster is not unrealistic. The effect of scattered clusters is observed during position estimation with the position decoding algorithm described in Chapter 4. (See Figure 4.11 for an example.)

FIGURE 2.21 The set of all locales which are closest to some point in image space is called a *locale cluster*. A locale cluster which is scattered in position space is shown here. This example is based on the Gaussian dot locale map of Figure 2.8. The shaded locales all have unit Hamming distance from the same point in image space.

Chapter 3

BOUNDS ON PRECISION

In this chapter, lower bounds on the mean square error (MSE) for position estimation are studied. Two bounds are discussed; the first is an attainable and determinable bound representing the theoretical limit of precision in digital images, the second is a determinable but unattainable bound called the *lineal bound*, which is particularly easy to compute.

A bound is *attainable* when there exists an algorithm (called a solution) which will achieve the bound. A bound is *determinable* when the value of the bound can be calculated, whether or not a solution to attain it exists. If a bound is attainable, it may be possible to implement the solution so that the value of the bound can be determined by simulation. If the solution is difficult or impossible to implement, the bound might be determined by theoretical analysis of the solution.

The bound on channel capacity established by Shannon for communications theory (see [Shann49] or [Sakri68, s.8-5]) is an example of a well known bound based on an optimal algorithm. In this case, a signal encoding algorithm maximizes the data transmission rate down a noisy channel for a given error rate. For the classical problem of linear signal estimation, the optimal solution is provided by the Wiener-Hopf equation [Sakri68, s.7]. The performance bound based on this solution provides a benchmark in many applications. Another example, in statistical parameter esti

mation, is the so-called information inequality, or Cramér-Rao lower bound, for the variance of an estimator. This bound is determinable but is not always attainable [Bicke77, s.4.3].

Bounds on precision are useful because they can be used to predict performance, based on theoretical principles, to help select system parameters or to investigate the feasibility of a system. Theoretical performance bounds provide universal benchmarks against which the performance of diverse systems can be compared. The theoretical basis of performance bounds often provide useful insights into essential issues of the problem.

The theoretical limit to MSE for position estimation, as developed in this chapter, is determined by three parameters of the locale map. The parameters represent the density, shape and size of the locales in the locale map. This bound is attained by the position decoding algorithm, which is discussed in Chapter 4.

Computation of the three parameters which determine the optimal MSE is computationally difficult so a more easily computed bound is derived, called the *lineal bound*. This bound is formulated in terms of a single parameter, the *lineal density* of locales, which is determined by the *total variation* of the image function. It is shown how the total variation can be easily computed by graphical and analytical methods.

Several examples targets are considered to demonstrate the lineal bound. The dependence of the bounds on the amplitude and size of the target is examined.

3.1 Optimal Position Estimation

A position estimation scheme uses the values of the pixels in the digital image, together with a priori knowledge about the target, to determine an estimate $(\hat{x}, \hat{y})$ for the target position parameters. The values of the pixels are unchanging for the target positioned within a locale $\ell \in \mathcal{L}$, thus the position estimator can be formulated as a mapping T of locales into target positions with $T(\ell) = (\hat{x}, \hat{y})$.

The precision of an estimator is often expressed as its *mean square error* (MSE)

$$\text{MSE} = \sum_{\ell \in \mathcal{L}} \int_{(x,y) \in \mathbf{D}} \left((x - \hat{x}_\ell)^2 + (y - \hat{y}_\ell)^2 \right) d\mu(x,y) \quad (3.1-1)$$

where $\mu(x,y)$ is the a priori probability measure on the target position space. It is assumed henceforth that the probability measure is constant; that is, $dx dy$ can be written in place of $d\mu(x,y)$. This technical detail is made explicit here to emphasize the fact that the choice of target position parameter can affect the MSE. For example, a uniform probability distribution for the target position in polar coordinates results in a non-uniform distribution in rectangular coordinates with a disproportionately large probability of the target being in the vicinity of the origin.

The characteristic function χ_ℓ of a locale ℓ in $\mathbf{D}$ is defined as

$$\chi_\ell(x,y) = \begin{cases} 1 & \text{for } (x,y) \in \ell \\ 0 & \text{otherwise.} \end{cases} \quad (3.1-2)$$

The size (area) of the characteristic function is denoted $|\chi_\ell|$,

$$|\chi_\ell| = \iint_{(x,y) \in \mathbf{D}} \chi_\ell(x,y) dx dy. \quad (3.1-3)$$

If A is the area of $\mathbf{D}$, then $|\chi_\ell|/A$ is the a priori probability that the target will be positioned within locale ℓ . When the characteristic function is used in an integral, the limits of the interval are presumed to be the entire target position space.

The centroid of a locale ℓ is $(\bar{x}_\ell, \bar{y}_\ell)$, with

$$\bar{x}_\ell = \iint x \chi_\ell dx dy \quad \text{and} \quad \bar{y}_\ell = \iint y \chi_\ell dx dy. \quad (3.1-4)$$

Within an individual locale ℓ , the MSE of an estimate $T(\ell) = (\hat{x}_\ell, \hat{y}_\ell)$ is MSE_ℓ ,

$$\begin{aligned} \text{MSE}_\ell &= |\chi_\ell|^{-1} \iint \left((x - \hat{x}_\ell)^2 + (y - \hat{y}_\ell)^2 \right) \chi_\ell dx dy. \\ &= |\chi_\ell|^{-1} \iint \left(\left[(x - \bar{x}_\ell)^2 + (y - \bar{y}_\ell)^2 \right] + \left[(\bar{x}_\ell - \hat{x}_\ell)^2 + (\bar{y}_\ell - \hat{y}_\ell)^2 \right] \right) \chi_\ell dx dy. \\ &= \frac{m_\ell^{(2)} + b_\ell^2}{|\chi_\ell|}. \end{aligned} \quad (3.1-5)$$

The first term $m_\ell^{(2)}$ is the second central moment of the locale. The second term b_ℓ^2 is an error term referred to here as the *skew* of the estimate. Note that the exponent of $m_\ell^{(2)}$ is written in parenthesis to indicate the order of the moment rather than exponentiation. The MSE has a minimum value of $m_\ell^{(2)}$, which is attained by the (unique) optimal estimate which maps each locale into its centroid; $T(\ell) = (\bar{x}_\ell, \bar{y}_\ell)$.

Optimal Position Estimator The *optimal position estimator* maps each locale in the target position space into the locale centroid $(\bar{x}_\ell, \bar{y}_\ell)$. This optimal estimator, $T_0(\ell) = (\bar{x}_\ell, \bar{y}_\ell)$, defined on a position space $\mathbf{D}$ with area A , has a mean square error MSE_0 , determined by the second central moments $m_\ell^{(2)}$ of the locales;

$$\text{MSE}_0 = \frac{1}{A} \sum_{\ell \in \mathcal{L}} |\lambda_\ell| \text{MSE}_\ell = \frac{1}{A} \sum_{\ell \in \mathcal{L}} m_\ell^{(2)}. \quad (3.1-6)$$

A unique optimal position estimator T_0 exists for every target.

Existence of the optimal estimator is assured by the existence of a locale map for the target. Section 2.3.3 shows how to construct a locale map for an arbitrary image function. Uniqueness of the optimal estimator is a direct consequence of uniqueness of the locale centroid and Equation 3.1-5.

An example of a sub-optimal estimator with non-zero skew is illustrated in Figure 3.1. The position estimates $(\hat{x}_\ell, \hat{y}_\ell)$ for the centre of mass estimate (COM) (Section 5.2.2) applied to the Gaussian dot target are shown superimposed on the locale map for the target. Several of the estimates clearly depart from the locale centroid, contributing to the skew of the position estimate.

The optimal position estimator for straight binary edge segments is discussed in [Dorst86s]. An implementation of the optimal position estimator is discussed in Chapters 4 and 5.

FIGURE 3.1 The location of the position estimates (indicated by small circles) obtained with the centre of mass (COM) position estimator for a binary circle are displayed here superimposed on the locale pattern within a unit pixel square. Two sources of estimation error are evident: the position uncertainty due to the area of each locale, and a skew error introduced when the position estimates are not at the centroid of their respective locale. The estimates for the locales of type-5 (c.f. Figure 2.8) are most obviously skewed.

3.2 Lower Bounds on Estimation Errors

Consider Figure 3.2, which depicts the dependence of MSE on target size. The upper region of the figure (above curve (A)) corresponds to position estimation precision which is *realizable* with practical algorithms and available technology. The MSE of any practical position estimation scheme will be greater than the theoretically optimum MSE_0 defined in Equation 3.1-6 (represented by curve (B) in the figure). As position estimation algorithms and available technology improve, the boundary of realizable precision (A) moves downward toward the theoretical limit (B). The theoretical limit provides an unchanging benchmark against which the performance of realizable position estimation systems can be compared.

Although the optimal precision (B) cannot be realized, it is *attainable* by the (unique) optimal position estimator T_0 . Precision below curve (B) is *not attainable*. The regions between curves (A) and (B) corresponds to precision which is attainable

FIGURE 3.2 The three curves shown here depict the limit to realizable precision (A), the theoretically optimal precision MSE_0 (B) and the lineal bound on precision (C). Practical limitations such as external noise and algorithm sub-optimality constrain realizable precision to the region above curve (A). The optimal position estimator attains the theoretical bound on precision (B) in the absence of noise. The lineal bound (C) is an easily computed approximation to (B). The lineal bound lies strictly below the optimal bound, within the region of unattainable precision (vertically striped). The region of attainable but unrealizable precision, between curves (A) and (B), represents the gap between theory and practice. The gap between (B) and (C) represents the tightness (or looseness) of the lineal bound.

but unrealizable. The size of this region represents the *tightness* (or looseness) of the theoretical bound with respect to the realizable bound on precision.

An expression for the value of the optimal precision (B) is derived in this section, however it is computationally difficult to determine. The *lineal bound*, derived in Section 3.3.4, provides an alternate bound on precision (C), which is an easily computed lower bound on the optimal precision.

Notice that a lower bound on MSE_0 can be used as a decision boundary to state when a system *cannot* achieve a particular level of performance (*i.e.*, the precision is unattainable) but it does not provide a reliable indication of when a system *can* achieve a particular level of performance (*i.e.*, is realizable). An upper bound on MSE_0 is of little use as a decision boundary since it does not permit a statement regarding the attainability or the realizability of performance.

The tightness of the optimal precision (B) to the realizable bound (A) is not an issue here, however, the tightness of the lineal bound (C) to the optimal precision (B) is. It is desirable to have the lineal bound as tight as possible to the optimal precision, while still having computational simplicity. The tightness of the lineal bound is investigated in Chapter 5 by comparing it to the performance of the optimal position decoding algorithm.

3.2.1 Dependence on locale map parameters

In this section, it is shown that the optimal MSE can be determined from three parameters of the locale pattern; namely, the locale density N_a , a locale shape parameter S , and a locale size parameter D .

Area-Density of Locales N_a Let $\mathcal{L}$ be the set of locales in target position space, let $N = |\mathcal{L}|$ be the number of locales in $\mathcal{L}$ and let $A = \sum_{\ell \in \mathcal{L}} |\chi_\ell|$ be the total of the sizes of all locales in $\mathcal{L}$. The *area-density of locales* is defined to be $N_a = N/A$.

Locale Shape Parameter S Let the *shape coefficient* of an individual locale ℓ be $s_\ell = m_\ell^{(2)} / |\chi_\ell|^2$, where $m_\ell^{(2)}$ is the second central moment of ℓ (Equation 3.1-5) and $|\chi_\ell|$ is the size of ℓ . The *shape parameter* S for the set $\mathcal{L}$ of locales in target position space is defined as

$$\frac{1}{S} = \frac{1}{N} \sum_{\ell \in \mathcal{L}} \frac{1}{s_\ell}. \quad (3.2-1)$$

Locale Size Parameter D The *size parameter* D for the set $\mathcal{L}$ of locales in target position space is defined as

$$D = \frac{1}{N} \sum_{\ell \in \mathcal{L}} N_a^2 s_\ell \left(|\chi_\ell| - \frac{S}{N_a s_\ell} \right)^2, \quad (3.2-2)$$

where the locale density N_a , shape coefficient s_ℓ and shape parameter S are as defined above.

The terminology *shape coefficient* is used for s_ℓ because, for a two-dimensional target position space, locales with the same shape will have the same value for s_ℓ ,

independent of the locales sizes (as shown in Appendix A.4). The size parameter D is defined as a weighted mean square deviation of the locale sizes from the value $S/(N_a s_\ell)$. It will be shown below that the optimal mean square error MSE_0 is minimized when the shapes and sizes of locales happen to satisfy

$$|\lambda_\ell| = \frac{S}{N_a s_\ell}, \quad (3.2-3)$$

in which case $D = 0$. Both S and D are invariant with scale changes in target position space. The shape parameter is invariant because the coefficients s_ℓ are shape invariant. The size parameter is invariant because N_a and $|\lambda_\ell|$ vary inversely with each other as the scale changes.

Notice that Equation 3.2-3 imposes the constraint that all locales of the same shape must have the same size but not vice versa. If all locales have the same shape and size, then $|\lambda_\ell| = 1/N_a$. This restrictive condition requires that the locales form a *monohedral tiling* in position space [Grunb86].

Let $\varepsilon_\ell = |\lambda_\ell| - S/(N_a s_\ell)$. Substituting this into Equation 3.1-6 (using $MSE_\ell = s_\ell |\lambda_\ell|$) gives the desired expression for the theoretical optimum bound on position estimation precision, as follows (see Appendix A.5 for more details).

The Optimal Mean Square Error The optimum mean square error is given by the expressions

$$\begin{aligned} MSE_0 &= \frac{1}{A} \sum_{\ell \in \mathcal{L}} s_\ell |\lambda_\ell|^2, \\ &= \frac{S}{N_a} + \sum_{\ell \in \mathcal{L}} s_\ell \varepsilon_\ell^2, \\ &= \frac{(S + D)}{N_a}. \end{aligned} \quad (3.2-4)$$

The parameter D will be zero if and only if Equation 3.2-3 is satisfied.

In this simple yet important relationship, the denominator depends only on the density of locales (irrespective of locale shape and size) while the numerator depends only on the locale shape and size parameters (irrespective of any scale applied to the

FIGURE 3.3 Periodicity of the locale pattern allows the left and right sides of a unit square position space to be equated to form a cylinder. The locale boundaries on the opposite edges align perfectly so that the first and last (partial) locales along a horizontal line segment are treated as one complete locale. The *lineal density* of locales is the number of locales along the unit length circumference of the line segment around the cylindrical position space. The lineal density may change with the vertical position of the line segment, it is averaged to obtain the *average lineal density* N_l , which is equal to the number of locales on the toroidal surface formed by equating the ends of the cylinder (not depicted)

size of locales). The positive terms $s_t \varepsilon_t^2$ are zero when Equation 3.2-3 is satisfied, thereby minimizing MSE_0 .

Unfortunately, it is difficult to evaluate the three parameters N_a , S and D for an arbitrary locale map and so, in the next section, the alternative lineal bound is examined.

3.2.2 Lineal density and the directional MSE

Consider a unit square target position space $\mathbf{D}$ and a unit length horizontal line segment $\Lambda(y_0)$

$$\Lambda(y_0) = \{ (x, y) \in \mathbf{D} : y = y_0 \} . \quad (3.2.5)$$

as depicted in Figure 3.3. To define the *lineal density* $N_l(y_0)$ of locales along $\Lambda(y_0)$, join the left and right edges of $\mathbf{D}$ so that $\Lambda(y_0)$ becomes a circle around a cylinder. (The reason for joining the edges will become apparent later in this section.) The lineal density of locales along $\Lambda(y_0)$ is defined as the number of locales intersected by the line segment around the cylinder. A more formal definition of lineal density follows.

Lineal Density of Locales $N_l(y_0)$ For a sufficient sampling window (Section 2.2.1) and a translational target model (page 33), the locales in the target position space $\mathbf{D}$ can be extended periodically outside of $\mathbf{D}$ (Section 2.2.4 and Figure 2.8). Let $\Lambda(y_0)$ be a line of length L , parallel to the x -axis and with a y -intercept of y_0 . Let $N(L)$ be the number of locales intersected by $\Lambda(y_0)$ along its length. The *lineal density* of locales in the direction of $\Lambda(y_0)$ is defined as

$$N_l(y_0) = \lim_{L \rightarrow \infty} \frac{N(L)}{L}. \quad (3.2-6)$$

The *average lineal density* of locales is the integral of the lineal density with respect to the offset y_0 ;

$$N_l = \int_{-\frac{1}{2}}^{\frac{1}{2}} N_l(y) dy. \quad (3.2-7)$$

The limits of the integral in the above equation are arbitrary, as long as they span a unit interval, since the locale pattern is extended periodically. If $\Lambda(y_0)$ is a unit length line segment which does not begin or end on a locale boundary, and if N is the number of locales intersected by it, then $N_l = N-1$. On the other hand, if $\Lambda(y_0)$ does begin and end on locale boundaries then, depending on which locales the end points belong to, the lineal density may be one more or one less than $N-1$. The formal definition of lineal density given above resolves this ambiguity, albeit in a somewhat pedantic manner. If $\Lambda(y_0)$ is considered to be a unit perimeter circle, as discussed earlier, then the lineal density of locales is always *equal* to the number of locales along it.

The portion of $\Lambda(y_0)$ which lies entirely within a locale ℓ is denoted $\Lambda_\ell(y_0)$, as in Figure 3.4. The lineal density of locales is the reciprocal of the average length of these subsegments.

Consider a position estimator $T(\ell) = (\hat{x}_\ell, \hat{y}_\ell)$. Let the centroid of locale ℓ be denoted as (x_ℓ, y_ℓ) and let the mid point of the segment $\Lambda_\ell(y_0)$ be denoted as $(x_\ell(y_0), y_0)$. Within the individual locale then, the *directional mean square error* in the x -direction,

FIGURE 3.4 Symbols used in the text are illustrated. The portion $\Lambda_\ell(y_0)$ of a line segment within an individual locale ℓ is shown as a thick horizontal line. Its mid-point $x_\ell(y_0)$ may not align vertically with the locale centroid (x_ℓ, y_ℓ) . The area of the locale $A_\ell = |\lambda_\ell|$ is shaded. The lineal bound on precision is based on the estimator $T_x(\ell) = (x_\ell(y_0), y_0)$, which is the minimum variance estimator given that the target lies within ℓ and on the line $\Lambda(y_0)$.

which is denoted as $\text{MSE}x_\ell$, can be expressed as

$$\begin{aligned}
 \text{MSE}x_\ell &= |\lambda_\ell|^{-1} \int (x - \hat{x}_\ell)^2 \lambda_\ell dx dy \\
 &= |\lambda_\ell|^{-1} \int \left((x - \bar{x}_\ell)^2 + (\bar{x}_\ell - \hat{x}_\ell)^2 \right) \lambda_\ell dx dy, \\
 &= |\lambda_\ell|^{-1} \int_y \int_{\Lambda_\ell(y)} \left((x - x_\ell(y))^2 + (x_\ell(y) - \hat{x}_\ell)^2 \right) dx dy + b_{\ell,x}^2, \\
 &= \frac{1}{12} |\lambda_\ell|^{-1} \int |\Lambda_\ell(y)|^3 dy + \epsilon_{\ell,x}^2 + b_{\ell,x}^2, \\
 &\geq \frac{1}{12} |\lambda_\ell|^{-1} \int |\Lambda_\ell(y)|^3 dy.
 \end{aligned} \tag{3.28}$$

The term $b_{\ell,x}^2$ is the skew of the estimate in the x -direction caused by the deviation of the estimate $(\hat{x}_\ell, \hat{y}_\ell)$ from the locale centroid (x, y) . For an optimal estimate, this term is zero. The term $\epsilon_{\ell,x}^2$ arises from the lateral difference between the locale centroid and the mid-point of the line segment. It expresses the lack of left-right (x -axis) symmetry in the locale.

Averaging $\text{MSE}x_\ell$ over all locales gives $\text{MSE}x$.

$$\text{MSE}_x = \sum |A_\ell| \text{MSE}_{x_\ell} . \quad (3.2-9a)$$

$$\geq \frac{1}{12} \int \sum |\Lambda_\ell(y)|^3 dy , \quad (3.2-9b)$$

$$\geq \frac{1}{12} \int \left(\sum \frac{1}{N_\ell(y)} \right)^3 dy = \frac{1}{12} \int \left(\frac{1}{N_l(y)} \right)^2 dy , \quad (3.2-9c)$$

$$\geq \frac{1}{12N_l^2} . \quad (3.2-9d)$$

The last two steps use the fact that, given a set of positive numbers X with mean $\bar{x}$, and a function f which is convex upwards, the average of $f(x)$ for all $x \in X$ is greater than or equal to $f(\bar{x})$. Convex upwards means that $f(x_1) + f(x_2) \leq 2f(\frac{1}{2}(x_1 + x_2))$. To get Equation 3.2-9c, use $x \equiv \Lambda_\ell(y)$ and $f(x) \equiv |x|^3$ and to get Equation 3.2-9d, use $x \equiv N_\ell(y)$ and $f(x) \equiv x^{-2}$.

Equation 3.2-9 gives a bound for the directional MSE in terms of the lineal density of locales. In the next section, bounds are discussed for the lineal and area locale densities themselves.

3.3 Bounds on Locale Density

Determination of the exact number of locales in the position space of an arbitrary target is a difficult problem. This enumeration problem has been considered for the specific case of binary edges [Dorst84d] [Beren88] but the solution is complex and cannot be easily generalized to other target forms.

This section discusses two alternative approaches. The first considers locales in image space, since the number of locales in image space and in position space are the same. This approach has limited success. The lineal density of locales is then considered. This more successful approach uses a measure of variability in the image function, referred to as the *total variation*, to estimate the number of locales per unit length. It is shown that the total variation is easily determined for arbitrary image functions and that it provides a bound on the lineal density of locales.

3.3.1 The area density of locales

Recall that the digital image of a target is represented by a set of K pixel values from a sampling window $\mathbf{W}$ (Section 2.2.1). Each such set of pixel values is a point in K -dimensional image space $\mathbf{Q}^K$. The set $\mathcal{L}$ of all locales is a subset of image space, and so a trivial bound on the total number of locales is given by $B_1 = |\mathbf{Q}^K|$.

This bound is not very satisfying because it is too loose. For example, the binary Gaussian dot of Section 2.2.5 has only 25 locales but the bound B_1 is 512.

A tighter bound can be established by finding the smallest box which encloses all of the locales in image space. This K -dimensional box has sides of length c_k , determined by the range of values for the pixels $p_k = (i_k, j_k)$ over the position space $\mathbf{D}$.

$$c_k = \max_{(x,y) \in \mathbf{D}} \mathcal{I}_0(x-i_k, y-j_k) - \min_{(x,y) \in \mathbf{D}} \mathcal{I}_0(x-i_k, y-j_k) + 1. \quad (3.3.1)$$

The number of K -dimensional points within this box is $B_2 = c_1 c_2 \cdots c_K$, which provides a new bound on the number of locales.

$$|\mathcal{L}| \leq B_2 = \prod_{1 \leq k \leq K} c_k \leq B_1. \quad (3.3.2)$$

Unfortunately, B_2 may not be much tighter than B_1 . For the binary Gaussian dot, B_2 is equal to 256, which is still half the size of B_1 and ten times greater than the actual number of locales.

In search of a still tighter bound, the topology of locales in image space might be taken into account. For example, if the average number of locales adjacent to each locale in image space is n then the average locale density is less than $n/(2K)$ (as discussed further in Appendix A.3.1), which gives a tighter bound B_3 .

$$|\mathcal{L}| \leq B_3 = \frac{n}{2K} B_2. \quad (3.3.3)$$

Applying this bound to the Gaussian dot, using $n = 4$ and $K = 9$, gives a bound of $B_3 = 57$. This bound is much tighter than the others but it depends on the heuristic assumption that $n = 4$.

FIGURE 3.5 Estimating the total number of locales in image space is depicted. Method-1 simply counts the number of points in the entire space. Method-2 determines the smallest box in K -dimensional image space which encloses all the locales. Method-3 estimates the average density of locales in image space, refining the bound of method-2. The actual number of locales is less than the estimates obtained by any of these methods.

The three bounds B_1 , B_2 and B_3 on the number of locales are depicted in Figure 3.5. Improving these bounds significantly, while maintaining generality with regard to the form of the target, is difficult. Rather than counting locales in an area, it is more effective, as will be seen later in this section, to count locales along the length of a line segment. Before doing so, the concept of total variation is introduced.

3.3.2 Total variation

The density of locales in position space is determined by the density of the meshwork in the locale map constructed from the contour lines of the image function (Section 2.3). Since contours in the locale map represent changes in the image function,

the variations in the image function will determine the density of locales. The idea is to establish the average change in the pixel values as the image function is translated in a given direction. This measure of change in the image function is called *total variation* V and will be used to provide a bound on the lineal density of locales.

Without regard to rigor, the *total variation* in a direction $\vec{x}_0$ could be expressed as an integral of the magnitude of the derivative of the digital image:

$$V = \frac{1}{A} \int_{\mathbf{D}} \left| \frac{\partial}{\partial \vec{x}_0} \mathcal{I}_{\vec{x}} \right| d\vec{x}, \quad (3.3-4)$$

where A is the area of the position space $\mathbf{D}$. This heuristic definition expresses the essential meaning of total variation: a formal mathematical treatment of total variation for vector valued functions can be found in [Rudin64].

For simplicity, assume that the direction $\vec{x}$ is parallel to the x -axis. Furthermore, the position space $\mathbf{D}$ is assumed to be the unit pixel square centred at the origin, the sampling window $\mathbf{W}$ is assumed to be sufficient (Section 2.2.1), and the image function $\mathcal{I}$ is assumed to be translation invariant (Section 2.1.4).

The *total variation* V will be defined as the average of the *lineal variation* $V(y_0)$, which is defined next.

A unit length horizontal line segment $\Lambda(y_0) \subset \mathbf{D}$ is divided into $2M$ equal-length pieces, each of length $\Delta = 1/(2M)$. The *lineal variation* $V(y_0)$ is defined as the sum of the magnitude change in the digital image $\mathcal{I}$ across the $2M$ pieces of $\Lambda(y_0)$ as M gets very large.

$$\begin{aligned} V(y_0) &= \lim_{M \rightarrow \infty} \sum_{m=-M}^{M-1} \left\| \mathcal{I}_{(m\Delta, y_0)} - \mathcal{I}_{((m+1)\Delta, y_0)} \right\|_1, \\ &= \lim_{M \rightarrow \infty} \sum_{m=-M}^{M-1} \sum_{(i,j) \in \mathbf{W}} |\mathcal{I}_0(m\Delta - i, y_0 - j) - \mathcal{I}_0((m+1)\Delta - i, y_0 - j)|. \end{aligned} \quad (3.3-5)$$

The limit operator in these equations is a mathematical formality to ensure that Δ is adequately small. The index i , which increments the x -axis in unit steps, and

the index m , for the small segments of length Δ , can be combined using $m\Delta + i = (m + 2Mi)\Delta$.

$$\begin{aligned} V(y_0) &= \lim_{M \rightarrow \infty} \sum_{m=-\infty}^{\infty} \sum_j |\mathcal{I}_0(m\Delta, y_0 - j) - \mathcal{I}_0((m+1)\Delta, y_0 - j)|, \\ &= \sum_j v(y_0 - j), \end{aligned} \quad (3.3-6)$$

where $v(y_0 - j)$ is called the *variation* of the image function $\mathcal{I}_0$ along the horizontal line with y -intercept of $y_0 - j$. Explicit reference to the range of the indices can be eliminated since the image function is assumed to be constant outside of the sampling window.

The total variation V is obtained by averaging the lineal variation $V(y)$.

$$\begin{aligned} V &= \int_{-\frac{1}{2}}^{\frac{1}{2}} V(y) dy, \\ &= \int_{-\frac{1}{2}}^{\frac{1}{2}} \sum_j v(y - j) dy, \\ &= \int v(y) dy. \end{aligned} \quad (3.3-7)$$

The three quantities – total variation, lineal variation and variation – should not be confused. It is the *total variation* V which will provide the desired bound on precision. Computation of the lineal variation $V(y)$ is a necessary formal step and may be used to illustrate the dependence of precision on the position parameters. The variation v is used for notational convenience.

3.3.3 Determining the total variation

Since the image function $\mathcal{I}_0(x, y)$ is quantized, the variation $v(y_0 - j)$ can be determined from a simple contour representation of the function, as in Figure 3.6. The variation $v(y_0 - j)$ for each horizontal line is the number of contours it intersects. The lineal variation $V(y_0)$ is the sum of the variations of all of the horizontal lines in the grating covering the contour pattern. Thus, the lineal variation of an image function

FIGURE 3.6 An image function such as a Gaussian dot is represented by a set of contour lines (solid circles). The lineal variation in the horizontal direction is obtained by counting the intersection points (marked with stars) of the contour lines with the overlaid grating of horizontal lines. The lines in the grating are placed at unit (pixel) vertical spacing.

along a line segment is the number of points in the intersection of two sets of lines; namely, the set of contour lines of the image function, and the set of grating lines $\Lambda(y_0 - j)$. This intersection set is depicted with stars in the figure.

Equation 3.3-6 accommodates very general forms of the image function but it can be greatly simplified if the image function is presumed to be *unimodal* along each line segment; that is, if the image function rises monotonically along $\Lambda(y_0 - j)$ from zero to a maximum value $M_j(y_0)$, then decreases monotonically again to zero, as depicted in Figure 3.7a. In this case, the lineal variation of the image function along $\Lambda(y_0)$ is given by

$$\begin{aligned} V(y_0) &= 2 \sum_j \max_{(x, y_0) \in D} |\mathcal{I}_0(x, y_0 - j)|. \\ &= 2 \sum_j M_j(y_0). \end{aligned} \quad (3.3-8)$$

This provides a straightforward method of analytic computation of the lineal total variance. It is easily generalized to image functions exhibiting *multimodal* characteristics, as in Figure 3.7b. If the local maxima along each segment $\Lambda(y_0 - j)$ are labelled in sequence as $M_j^{(0)}, M_j^{(1)}, \dots, M_j^{(k_j)}$, and their respective intervening local minima

FIGURE 3.7 Examples of *unimodal* (a) and *multimodal* (b) functions. A unimodal function increases monotonically from zero to a maximum value M and then decreases monotonically to zero. A multimodal function has intervening local extrema such as $m^{(1)}$ shown in (b). The total variation of multimodal functions is computed as the heights of the first and last maxima ($M^{(0)} + M^{(1)}$) plus the differences between successive extrema ($M^{(0)} - m^{(1)}$ and $M^{(1)} - m^{(1)}$).

are labelled as $m_j^{(1)}, m_j^{(2)}, \dots, m_j^{(k_j)}$, then the lineal variation is given by

$$\begin{aligned} V(y_0) &= \sum_j \left(M_j^{(0)} + \sum_{i=1}^{k_j} \left(M_j^{(i-1)} - m_j^{(i)} + M_j^{(i)} - m_j^{(i)} \right) + M_j^{(k_j)} \right) \\ &= 2 \sum_j \left(M_j^{(0)} + \sum_{i=1}^{k_j} \left(M_j^{(i)} - m_j^{(i)} \right) \right). \end{aligned} \quad (3.3-9)$$

In the above equation, the notation has been simplified by removing (y_0) .

The total variation V for the unimodal case is obtained from Equation 3.3-8 as

$$V = 2 \int M_0(y) dy. \quad (3.3-10)$$

A corresponding integral can be obtained for the multi-modal case from Equation 3.3-9.

3.3.4 The lineal bound

The lineal density of locales $N_l(y_0)$ is defined in Section 3.2.2 by counting locales along a line segment $\Lambda(y_0)$, just as the lineal variation $V(y_0)$ is in Section 3.3.2, however,

they need not give exactly the same value since they count them in different ways. The lineal variation counts locale boundary multiplicity (Section 2.4.3) while the lineal density counts distinct locales. There are two sources of discrepancy between N_l and V : locale boundary multiplicity and concave locales.

If the multiplicity of a locale boundary is greater than one, then the lineal variation will be greater than the lineal density. When a locale is not convex, it may be possible for $\Lambda(y_0)$ to enter and exit the *same* locale more than once, thereby causing the lineal variation to be greater than the lineal density. These possibilities force the inequalities

$$V(y_0) \geq N_l(y_0) \quad \text{and} \quad V \geq N_l. \quad (3.3-11)$$

Applying the second inequality to Equation 3.2-9d gives a lower bound on the directional mean square error MSE_x in terms of the total variation.

$$\begin{aligned} \text{MSE}_x &\geq \frac{1}{12N_l^2} \cdot \\ &\geq \frac{1}{12V^2} \cdot \end{aligned} \quad (3.3-12)$$

This bound is referred to as the *lineal bound* on geometric precision, as defined below.

The Lineal Bound on Geometric Precision The directional mean square error MSE_x , for all position estimation schemes applied to a given target, is bounded below by $1/(12V^2)$, where V is the total variation of the image function of the target.

Several bounding conditions were invoked to arrive at the lineal bound. Each condition potentially reduces the tightness of the bound. The bounding conditions and the equations in which they are used are summarized below.

1. Lack of left-right symmetry of individual locales introduces a non-zero bounding element $\epsilon_{l,x}^2$ in Equation 3.2-8.
2. Variations in the lengths $|\Lambda_l(y)|$ for different l make the mean of their cubes strictly greater than the cube of their mean, in Equation 3.2-9c.

3. Changes in the lineal density of locales $N_l(y)$ with the value of y make the mean square reciprocal greater than the reciprocal mean square in Equation 3.2-9d.
4. Multiplicity of locale boundaries and non-convexity of locales results in the total variation being strictly greater than the averaged lineal locale density in Equation 3.3-12.

Notwithstanding these stated conditions, the bound on the directional mean square error MSE_x given by Equation 3.3-12 is rigorous and provides an instructive and practical tool for the analysis of geometric precision in digital images. The tightness of the bound is investigated by simulations employing the position decoding algorithm (Section 5).

The optimal mean square error MSE_0 , expressed in terms of the three parameters N_a , S and D , is both determinable and attainable. It is attained by the position decoding algorithm and it may be determined either by implementation of that algorithm or by (arduous) evaluation of the three parameters from the locale map.

3.4 Examples

Examples of circular and straight-edge image functions are presented in this section to illustrate the lineal bound on position estimation precision and to explore its dependence on target size and amplitude. The term *target size* refers to the diameter of the circular targets and the length of the edge targets. The *target amplitude* is the difference between the background level surrounding the target and the brightest spot on the target. It is presumed that the background is uniformly zero.

The binary disk and the binary edge (see Figure 2.1) are presented as examples because they are realistic, frequently-used targets. In the image processing literature, circles and edges receive more attention than any other pattern, perhaps because they occur so abundantly in scenes from industrial applications. Binary image functions

illustrate the principles of the locale map and linear bound particularly well because of their simplicity. The grey-level image functions for circular targets and edges are also considered.

The binary disk and the Gaussian dot targets are examples of translational target models (page 33). They are analyzed using a sufficient sampling window and a unit square position space. The binary and grey-level edges, on the other hand, are not translational models. In fact, they are not even translation invariant and they do not have a sufficient sampling window. Edges provide an example of an abstract target position space and illustrate how the theory of locales for translational models can be extended to more general cases.

3.4.1 Binary disk

A binary disk is represented by a circular contour of radius r (Figure 3.8). It has a total variation (Equation 3.3-10)

$$\begin{aligned} V &= 2 \int_{-r}^r dy, \\ &= 4r. \end{aligned} \tag{3.4-1}$$

Recall, from the lineal bound in Equation 3.3-12, that the directional mean square error MSE_x satisfies $\text{MSE}_x \leq 12V^2$, so the root mean square error RMSE_x in position estimation is bounded by $8\sqrt{3}r$ for binary disk targets.

The lineal variation $V(y_0)$ is determined by Equation 3.3-8 to be the piecewise linear function

$$V(y_0) = 2(\lfloor r - y_0 \rfloor + \lfloor r + y_0 \rfloor). \tag{3.4-2}$$

where $\lfloor \cdot \rfloor$ represent integer truncation. It is often true, as it is in this case, that the total variation V is easier to determine, is a better behaved function and is more useful than the lineal variation.

2

FIGURE 3.8 The image function for a binary disk is represented by a single circular contour of radius r . The total variation V is a linear function of r with slope $V/r=4$.

3.4.2 Gaussian dot

The binary disk can be generalized to other circularly symmetric functions such as the Gaussian dot, described by the image function

$$I_0(x, y) = [Ae^{-(x^2+y^2)/(2\sigma^2)}]. \quad (3.4-3)$$

The target size is determined by the parameter σ and its amplitude is determined by A .

The image function for the Gaussian dot is represented by a set of $N=[A]$ concentric circular contours (see Figure 3.9). The n^{th} contour is the circle defined by $I_0(x, y)=n$, with the radius r_n given by $r_n^2 = 2\sigma^2 \ln(A/n)$. Considering the image function as a stack of binary disks, the total variation is computed as

$$\begin{aligned} V &= 4 \sum_{n=1}^N r_n, \\ &= 4\sqrt{2}\sigma \sum_{n=1}^N \sqrt{\ln(A/n)}, \\ &\approx 5.0\sigma(A-1). \end{aligned} \quad (3.4-4)$$

FIGURE 3.9 The Gaussian image function $I = [Ae^{-(x^2+y^2)/(2\sigma^2)}]$ has concentric circular contours with radii $\{r_n\}$ determined by $r_n^2 = 2\sigma^2 \ln(A/n)$ for $1 \leq n \leq [A]$. The total variation $V = 4 \sum r_n$ turns out to be essentially linear with target contrast A , with $V \approx 5\sigma(A-1)$.

Notice that V varies linearly with the target size σ . Computations have verified that V is also linear with the target amplitude A , with $V \approx 5\sigma(A-1)$, as shown in Figure 3.10. Asymptotic linearity, as A gets large can be demonstrated analytically by replacing the summation in the above equation by an integral.

$$\begin{aligned}
 V_{\infty} &= 2 \int_{-\infty}^{\infty} Ae^{-x^2/(2\sigma^2)}, \\
 &= 2\sqrt{2\pi}\sigma A, \\
 &= 5.01\sigma A.
 \end{aligned} \tag{3.4-5}$$

3.4.3 Binary edge segment

The image function and the locale pattern for the binary edge were introduced in Section 2.3.4. The edge is an important shape in digital image processing but it is anomalous with respect to the simplifying assumptions made for the presentation of the locales theory. The assumptions of translation invariance of the image function, sufficiency of the sampling window and finite support of the image function do not

FIGURE 3.10 The total variation V for the Gaussian dot, as determined by Equation 3.4-4, is closely approximated by $5\sigma(A-1)$.

hold for the binary edge. The locales approach to geometric precision still applies to cases such as this since these predicates are used only for convenience and are not essential to the theory.

The image of a binary edge is modelled as a long straight edge viewed through a square sampling window W . This window is not sufficient because the edge extends indefinitely. If K^2 is the number of pixels in the sampling window then its width K defines the length, or size, of the edge segment. The coordinates of the position space for the binary edge are (b, θ) , where b is the distance from the edge to the origin and θ is its orientation counterclockwise from horizontal.

The binary edge locale map. The target position space $\mathbf{D}$ for the binary edge is defined as

$$\mathbf{D} = \left\{ (b, \theta) : |b| \leq \frac{1}{2}, 0 \leq \theta < 2\pi \right\}. \quad (3.4-6)$$

There is considerable arbitrariness in the choice of target position space. The analysis in [Dorst86s] for example, uses an alternative position space $\mathbf{D}' = \{(b, \theta) : 0 \leq b < \cos \theta, 0 \leq \theta \leq \pi/4\}$.

FIGURE 3.11 The locale pattern in (b, θ) space for an edge is constructed from sinusoidal contours $b_{ij}(\theta) = j \cos \theta - i \sin \theta$, with the indices (i, j) determined by the sampling window W . This figure was generated using the 5×5 sampling window with $|i|, |j| \leq 3$. The position space $D = \{(b, \theta) : |b| \leq \frac{1}{2}, 0 \leq \theta < 2\pi\}$ (shaded region) includes all edges which intersect the unit diameter circle about the origin of the image plane. Only half of the position space is shown ($0 \leq \theta < \pi$) since the locale pattern in the other half is identical.

The locale map for an edge of length $K=5$ is illustrated in Figure 3.11. It is constructed from the contours of the image function, which are sinusoidal curves in (b, θ) coordinates given by

$$\begin{aligned} b_{ij}(\theta) &= j \cos \theta - i \sin \theta, \\ &= r_{ij} \sin(\alpha_{ij} - \theta), \end{aligned} \quad (3.4-7)$$

where $r_{ij} = i^2 + j^2$ and $\alpha_{ij} = \arctan(i, j)$. The indices i and j are determined by the sampling window $W = \{(i, j) : |2i^-| < K, |2j^-| < K\}$. (The notation i^- indicates a number less than i but close to it.) In this case it has the effect of skewing the window into the first quadrant when K is even. When K is odd, it has no effect and the window is symmetric about the origin.

The locale pattern of an edge differs in an important way from those of a binary disk, a Gaussian dot and many other target forms; the existing locale boundaries do not change as the edge is increased in size. This is because as pixels are added to the sampling window more sinusoidal contours are added to the locale map of the edges.

For a translational target model such as the Gaussian dot, any change in the target size will change all of the contours and the locale pattern will be drastically altered.

This unique characteristic of straight edges perhaps explains why detailed analysis of the locales has been done only for binary edges [Dorst84d] [Beren88]. These analyses would be difficult to perform for the more variable locale patterns of the Gaussian dot.

The total variation in b . The lineal variation $V_b(\theta_0)$ along a line segment $\Lambda_{\theta_0} = \{(b, \theta) \in \mathbf{D} : \theta = \theta_0\}$ is given by the number of contours crossing its unit length (Section 3.3.2), which can be expressed as

$$V_b(\theta_0) = \sum_{(i,j) \in W} U\left(\frac{1}{2} - |b_{ij}(\theta_0)|\right). \quad (3.4-8)$$

It turns out to be just as easy to find the total variation V_b , which is the number of locale boundaries along Λ_{θ_0} averaged with respect to θ_0 , as it is to find $V(\theta_0)$. The total variation V_b is

$$\begin{aligned} V_b &= \frac{1}{2\pi} \int_0^{2\pi} V(\theta) d\theta, \\ &= \frac{1}{2\pi} \sum_{(i,j) \in W} \int_0^{2\pi} U\left(\frac{1}{2} - |r_{ij} \sin(\alpha_{ij} - \theta)|\right) d\theta, \\ &= \frac{1}{2\pi} \sum_{(i,j) \in W} 4 \arcsin\left(\frac{1}{2r_{ij}}\right). \end{aligned} \quad (3.4-9)$$

Using this relationship, the total variation V_b is computed as a function of the width K of the sampling window. As shown in Figure 3.12, this relationship is essentially linear with slope $V_b/K \approx 1.12$. This result is in close agreement with [Beren88, Eq. 28] and [Dorst84d, s.IV-C].

It is interesting to see how this compares with the total variation obtained for the alternate position space $\mathbf{D}' = \{(b, \theta) : 0 \leq b < \cos \theta, 0 \leq \theta \leq \pi/4\}$ from [Dorst86s]. This position space corresponds to edges which are oriented in the first octant and have a y -intercept between 0 and 1. Notice that $\mathbf{D} \subset \mathbf{D}'$. The lineal variation $V'(\theta_0)$ on the line segment $\Lambda'_{\theta_0} \subset \mathbf{D}'$ is given by the number of indices $(i, j) \in W$ which satisfy

FIGURE 3.12 The total variation in the b and θ direction for a binary edge is plotted as a function of the edge length K . Precision in position b is a linear function of K , while precision in orientation θ is a quadratic function of K .

$0 \leq b_{i,j}(\theta_0) < \cos \theta_0$, normalized by the length of the line segment $|\Lambda'_{\theta_0}| = \cos \theta_0$. For each index i there is a unique j_{θ_0} with $(i \tan \theta_0) \leq j_{\theta_0} < (i \tan \theta_0 + 1)$, for which $0 \leq b_{i,j}(\theta_0) < \cos \theta_0$. Consequently, for every θ_0 there are K suitable indices (i, j_{θ_0}) in W and so the lineal variation is given by

$$V'(\theta) = \frac{K}{\cos \theta} \quad \text{for } 0 \leq \theta < \frac{\pi}{4}. \quad (3.4-10)$$

The total variation V'_b for this alternate position space is $(4/\pi) \int_0^{\pi/4} V'(\theta) d\theta = 1.12K$. The fact that $V_b \approx V'_b$ suggests that the locale density is distributed quite uniformly over the position space, despite the variations in locale size and shape.

The interaction of a straight-edged target with a regular sampling raster results in significant fluctuations in the lineal variation as a function of b and θ . For example, edges oriented at multiples of $\pi/4$ have particularly large position uncertainty, both in the b and the θ direction. This behavior, which results from the "eyes" in the locale pattern, might be obscured in realistic situations by the interaction of external noise with the uncertainty caused by quantization. Simulation studies have demonstrated this effect, as reported in Sections 5.5.3 and 5.5.4. This is an example of a situation in which external noise may actually *improve* the position estimation precision.

The total variation in θ . Since the position space $\mathbf{D}$ is defined with $|b| \leq \frac{1}{2}$, each sinusoidal contour, defined by the curve $b_{ij}(\theta) = r_{ij} \sin(\alpha_{ij} - \theta)$, is assured to cross a horizontal line Λ_{b_0} in $\mathbf{D}$ exactly 2 times unless $r_{ij} = 0$. There is a total of K^2 contours, so the lineal variation and the total variation are given by

$$V(b) = \frac{2(K^2 - 1)}{|\Lambda_{b_0}|} = \frac{(K^2 - 1)}{\pi} \quad (3.4-11)$$

and

$$V = \int V(b) db = \frac{(K^2 - 1)}{\pi}, \quad (3.4-12)$$

as illustrated in Figure 3.12.

Notice that the total variation for orientation depends on the *square* of the length of the binary edge. The simulation studies using noisy data indicate that this quadratic relationship persists in the presence of noise (Equation A.6-3).

Grey-level edges. The image function $\mathcal{I}_{ij}$ for an edge with M grey-levels is similar to that of the binary edge (see Section 2.3.4);

$$\mathcal{I}_{ij}(b, \theta) = \lfloor h(b + i \sin \theta - j \cos \theta) \rfloor, \quad (3.4-13)$$

where $\lfloor \cdot \rfloor$ denotes integer truncation and $h(x)$ is a monotonic edge profile function. If $h(x_m) = m$ for $1 \leq m \leq M$, then the image function can be expressed as the sum of M binary-edge image functions;

$$\mathcal{I}_{ij}(b, \theta) = \sum_{m=1}^M U(b + i \sin \theta - j \cos \theta - x_m). \quad (3.4-14)$$

The locale map generated from such an image function is a composite of M replicas of the binary edge locale map displaced by the offsets x_m in the b direction. As a consequence, the total variation is approximately M times that of the binary edge; that is, the total variation for grey-level edges is proportional to the number of grey levels. This is corroborated in [Klaas75, Fig. 5].

In summary, the total variation has been discussed for the binary disk, the Gaussian dot, and edge segments. The total variation V is related to the mean square

position estimation errors in the x -direction MSE_x by the lineal bound, expressed in Equation 3.3-12 as $\text{MSE}_x \geq 1/(12V^2)$. The lineal bounds for the above examples are shown below.

Target	Lineal Bound
Binary Disk	$4\sqrt{12}r \approx 13.8r$
Gaussian Dot	$5\sqrt{12}\sigma(A-1) \approx 12.2(A-1)\big _{\sigma=\frac{1}{\sqrt{2}}}$
Edge Offset	$1.12\sqrt{12}KM \approx 3.88KM$
Edge Orientation	$\sqrt{12}(K^2-1)M/\pi \approx 1.10M(K^2-1)$

Chapter 4

POSITION DECODING

A new position estimation technique, referred to as *position decoding*, is introduced in this chapter. The discussion begins with some motivation for the concept of position decoding and the choice of its title. The reference to decoding is explained by analogy with the general problem of digital signal coding and decoding in communications theory. Position decoding is performed by minimizing a cost function to obtain *feasible regions* for the target position.

A simple data structure called a *truth map* is central to the implementation of position decoding. It is a two-dimensional cost function implemented as a discrete array. A technique is developed for iteratively improving the resolution of the truth map, thereby dramatically improving computational efficiency.

A basic form of position decoding is described first, so as to clearly illustrate the underlying principles of the algorithm. Following this, position decoding with noisy data is discussed and the more subtle aspects of the algorithm are explained.

There are many avenues for pre-processing of pixel data and enhancing the basic algorithm. Several of the most important are discussed briefly in the context of preparing position decoding for implementation in a real application. In the final section of this chapter, position decoding is evaluated in terms of its uniqueness, its flexibility and stability, its optimality, and its computational complexity.

4.1 Purpose of the Algorithm

The purpose of the position decoding algorithm is to illustrate and explore aspects of the theoretical formalism of locales. The algorithm demonstrates that the optimal position estimator can, in fact, be implemented. By using the algorithm in a simulation environment, the gap between the lineal bound and the optimal bound on precision can be examined. The flexibility of the position decoding algorithm makes it possible to use different targets and to examine the effects of target parameters on position estimation precision. The algorithm is robust enough to allow preliminary investigations of the effects of noise.

The theory of locales has shown that the locale centroid is the optimal position estimate (Equation 3.1-5). It is one thing to postulate an optimal solution, but quite another to implement it, efficiently or otherwise. Position decoding demonstrates that this estimate can be realized in a practical and efficient manner. The algorithm is soundly based on the theory of locales and provides the optimum position estimate in the absence of noise. Its optimality is shown in the description of the algorithm and later is demonstrated by simulation in Section 5.5.4.

The lower bound on attainable precision, as provided by the lineal bound of the locale theory, is not a tight bound; there is a gap between realizable precision and that given by the bound. Position decoding will help to quantify this gap. As a realization of the optimal position estimate, the performance curves (position estimation errors) for position decoding define the boundary between *attainable* and *unattainable* precision, as illustrated in Figure 3.2. The performance of position decoding in the absence of noise thus provides a tight bound on the available geometric precision. While this may be superior to the lineal bound of the locales theory, it is much more difficult to compute. Both the lineal bound and that provided by position decoding are products of the same approach to the analysis of precision, namely, the locales theory. Notice that, as discussed in Section 3.2, an upper bound on attainable precision cannot be used as a decision boundary for realizability, whereas a lower bound can.

In terms of practical implementation, the objective here is to show only that the algorithm, or others derived from it, may have some potential for use in high precision position estimation applications. In the presence of noise, the algorithm compromises optimality to some extent to avoid distracting complications, but the basic qualities of superior precision, flexibility and robustness are retained.

The position decoding software was initially written to strictly implement the optimal position estimate in the absence of noise. The success of early simulation studies invited the application of the algorithm to noisy data. This required very few modifications to the basic algorithm and the modified algorithm retained the property of optimality as the noise level went to zero.

The theory of locales, as developed in Chapters 2 and 3, has not been extended in a formal way to noisy data. The position decoding algorithm, with its sound theoretical basis and tolerance to noise, is a tool for investigating locales and optimal position estimation in the presence of noise. Its use as an investigative tool is demonstrated in the simulation studies of Chapter 5.

4.2 Position Decoding Generalities

This section shows the reason for the use of the term *position decoding*, which is drawn from the field of communications theory. The term is intended to evoke the impression that the digital image is an encoded message and that the target position is determined by decoding the message. A perfect message decoding scheme will lead to the optimal position estimator.

4.2.1 Position decoding

In an abstract sense, a digital image is an encoding of information in a scene. Various types of information are encoded, such as the textural information identified by image segmentation procedures, the contextual information identified by image

understanding procedures and the geometric information extracted by position estimation algorithms. The nature of a digital image may range from an alphanumeric display to a diffraction pattern or an aerial photograph but the purpose is always to represent information. It is the task of image processing and analysis to extract this information. To a great degree, the success of this task depends on an understanding of how the information is encoded in the image in the first place.

The theory of locales has shown that the position of a target can be specified in terms of membership in an equivalence class of positions. Each equivalence class, or locale, corresponds to a unique digital representation of the target. The functional relationship between the target position and the corresponding digital representation is a *position encoding* scheme determined by the properties of the imaging system. The objective of any position estimation procedure is to try to perform some sort of inverse of the encoding scheme. The *position decoding* algorithm, in the absence of noise, performs this inverse exactly, based on a priori knowledge of the target and encoding scheme, to obtain the locale which contains the target position.

A description of the functional form of the imaging processes can be quite involved (see [Chamb78] or [Purll83] for example). Imaging sensor characteristics, lens distortions, electronic effects, transmission noise and quantization each contribute to the encoding process. All the characteristics of the imaging system are presumed to be incorporated into the *target model* used for position decoding.

Determining the target locale from the observed image provides a region, rather than a single point, for the target position. Position decoding uses a representative position from the locale as the actual position estimate. The best choice is given by the *locale centroid*, as explained in Section 3.1 and [Dorst86s].

4.2.2 Analogy with signal decoding

The analogy between position decoding and the decoding of messages from noisy digital communication channels is instructive. Besides commonality of terminology,

the analogy provides insights into the inherent redundancy of image encoding and the potential for optimal decoding and error correction in the presence of noise. Without a rigorous formalism for locales in the presence of noise, the proven techniques of digital signal decoding provide useful heuristics for position decoding.

In digital communications, a message consisting of a string of characters is encoded as a string of bits and is then transmitted. The encoding is done, not to scramble or obscure the message (encryption is used for that purpose), but rather to enhance the differences between messages and hence achieve error tolerance. A relatively small repertoire of messages is translated into a subset of a much larger code space. Messages are assigned to points in the code space which are as distant from each other as possible. If this distance is sufficiently large then the received signal will look very similar to the transmitted code even when some noise is added. In this case, errors can be corrected by inferring the intended code to be the one closest to the received signal. If the codes are packed too densely in the code space, then noise may cause ambiguity as to the intended code, and so errors may not be correctable. If an error can cause one code to be changed into another, then there is no way of knowing that an error occurred at all.

Decoding an encoded message received over a noisy digital communications channel requires two steps (which may be combined). The first step is to recover the transmitted code from the corrupted received signal based on the noise statistics of the channel. This is the error detection and correction step. It results in an inferred code; that is, it yields an estimate of what code was actually sent. The second step is to interpret the inferred code to obtain the transmitted message. The interpretation is based on inverting the message encoding scheme.

The analogy between signal decoding and position decoding is depicted in Figure 4.1. The encoding of the target is an idealized digital representation based on the model of the target. Errors due to noise and imperfections in the target model corrupt the digital representation, yielding the received or observed image. An inferred digital image is derived from the observed image using a target model. This is the

FIGURE 4.1 A target's position is encoded in an ideal digital representation of the image, based on a target model. The observed digital image is a corrupted version of the ideal one (upper right). The target position is found by deducing the most likely ideal image and then applying the inverse of the encoding process to it to obtain a target locale (lower middle). The centroid of the locale is computed to obtain the final position estimate.

most difficult step in that it must accommodate the vagaries of data in the real world and effectively perform error detection and correction. Note that the inferred image is an ideal image for the target at *some* position, just as the inferred code for signal decoding is some legal transmission code. The inferred image is interpreted to obtain its corresponding locale, which provides the final position estimate for the target.

4.3 Basic Position Decoding

Position decoding is based on the ideal of intersecting feasible regions. Each pixel in the digital representation of a target has an associated feasible region, as defined in this Section. It is shown that by intersecting all of these feasible regions the locale corresponding to the actual target position is obtained. Knowing the locale in which the target is positioned makes it possible to obtain the optimal position estimate.

From the size and shape of the locale, it is also possible to determine the uncertainty in position caused by quantization.

4.3.1 Intersection of feasible regions

Let $\{\tilde{p}_k\}$ be a set of pixels from an observed digital image of the target and let $\{(i_k, j_k)\}$ be the set of corresponding pixel indices. The *feasible region*, denoted $R_{\tilde{p}_k}$, for the value of a pixel is defined as the region of target positions for which the image function is equal to the pixel value. Formally, the feasible region $R_{\tilde{p}_k}$ for pixel index (i_k, j_k) with the observed pixel value $\tilde{p}_k$, is defined as a subset of the target position space $\mathbf{D}$ with

$$R_{\tilde{p}_k} = \{(x, y) \in D : \mathcal{I}_{i_k, j_k}(x, y) - \tilde{p}_k = 0\}, \quad (4.3-1)$$

where $\mathcal{I}_{i_k, j_k}(x, y)$ is the pixel's image function as defined in Section 2.1.5.

Figure 4.2 shows an image function represented by a set of contours. One contour interval, corresponding to a pixel value $\tilde{p}_k$, is shaded. On the right side of the figure, the feasible region for this pixel value is shown (darkly shaded) as the restriction of the contour interval to the position space $\mathbf{D}$ (indicated by a square in the figure). The position of the circular contours of the image function, relative to the position space, depends on the position (x_{ij}, y_{ij}) of the pixel p_k (Equation 2.1-3).

The feasible region $R_{\{\tilde{p}_k\}}$ for a set of pixels $\{\tilde{p}_k\}$ (such as that defined by the sampling window $\mathbf{W}$) is given by the intersection of the feasible regions of the individual pixels;

$$R_{\{\tilde{p}_k\}} = \bigcap_{\tilde{p} \in \{\tilde{p}_k\}} R_{\tilde{p}}. \quad (4.3-2)$$

The intersection of two feasible regions is depicted in Figure 4.3, where set intersection is denoted with a $+$ sign. The feasible regions are intersected until the target locale is obtained.

FIGURE 4.2 The target model, or image function, may be represented by a set of contours such as the concentric circles shown at the left. An observed value for a single pixel in the digital image of the target will correspond to some contour interval of the image function, such as the shaded region in the figure. The target position space (a unit pixel square) is identified by the small dark square on the left. The feasible region for the observed pixel value is the restriction of the shaded contour interval to the position space as shown (enlarged) by the dark region on the right.

Obtaining the Target Locale. When there is no noise in the digital image, the intersection of the feasible regions of all pixels in the sampling window W is exactly equal to the target locale ℓ ;

$$\ell = \bigcap_{(i_k, j_k) \in W} R_{\tilde{p}_k} . \quad (4.3-3)$$

In the absence of noise, the target locale ℓ is a subset of every feasible region $R_{\tilde{p}_k}$ and so $\ell \subseteq R_{\{\tilde{p}_k\}}$ for any pixel set $\{\tilde{p}_k\}$. When the pixel set includes all pixels in the sampling window W , then $R_{\{\tilde{p}_k\}} \subseteq \ell$ since all points in $R_{\{\tilde{p}_k\}}$ will have exactly the same digital image.

4.3.2 Locale centroid and size

It has been shown (see Equation 3.1-5) that the optimal position estimate is given by the centroid of the locale. Since the intersection of feasible regions provides the locale of the target, it is a simple matter to obtain the optimal position estimate by computing the locale centroid.

FIGURE 4.3 Each pixel in the digital image of a target has an associated feasible region in position space. These regions are intersected, as depicted here, by summing the cost functions. The cost function is defined with the value 0 corresponding to *true* and all positive values corresponding to *false* so that arithmetic summation results in a logical AND operation.

The position estimation uncertainty caused by quantization is determined by the size of the locales. It is useful to be able to estimate the position uncertainty in conjunction with the target position. Position uncertainty may be expressed, for example, as the area of the locale or as the width and height of the locale. Notice that the position uncertainty can be determined without knowledge of the actual target position; this is made possible by the information which is implicitly provided by the target model about the target shape.

4.3.3 The truth map

Feasible regions are represented in the position decoding computer program by a two-dimensional array of cells referred to as a *truth map*. A truth map $T(n, m)$ represents a grid overlaying the target position space. Each cell (n, m) in the truth map corresponds to a single target position (x_n, y_m) given by

$$\begin{aligned} x_n &= A_x + S_x n, \\ y_m &= A_y + S_y m, \end{aligned} \tag{4.3-4}$$

where (A_x, A_y) and (S_x, S_y) are the truth map *offset* and *scale* constants respectively.

The truth map is not unlike a conventional truth table. For a pixel value $\hat{p}_k$ the

the truth map is defined in terms of the feasible region $R_{\tilde{p}_k}$ as

$$T(n, m) = \begin{cases} 0 & \text{if } (x_n, y_m) \in R_{\tilde{p}_k} \\ 1 & \text{otherwise} \end{cases} \quad (4.3-5)$$

The value 0 corresponds to *true* and values greater than 0 correspond to *false* so that the logical AND operation on feasible regions can be performed by arithmetic addition. The feasible region for a pixel is represented in the truth map by the area of true cells. To compute the feasible region for a set of pixel values $\{\tilde{p}_k\}$, each pixel is selected in succession, the truth map is evaluated for it according to the above relation using an auxiliary truth map T' for storage, and the auxiliary truth map is then added to the main truth map T . If $T^{(k)}$ is the main truth map at the start of the k^{th} iteration (*i.e.*, after processing the pixel values $\tilde{p}_0, \dots, \tilde{p}_{k-1}$) then T' is computed using $\tilde{p}_k$ according to Equations 4.3-5 and 4.3-1, and then combined with $T^{(k)}$ to obtain the new main truth map $T^{(k+1)}$ as

$$T^{(k+1)}(n, m) = T^{(k)}(n, m) + T'(n, m). \quad (4.3-6)$$

An important characteristic of the algorithm is that the feasible regions are computed *pixel-wise* rather than *position-wise*. Position-wise evaluation would completely determine each cell of the truth map by testing whether or not $(x_n, y_m) \in R_{\tilde{p}_k}$ for *all* of the pixel values before proceeding to the next cell. Position-wise evaluation is used, for example, in template matching [Gonsa76] and sequential similarity detection algorithms [Barne72]. Pixel-wise evaluation, on the other hand, computes the entire truth map for each pixel before proceeding to the next. Position decoding uses pixel-wise evaluation because it provides opportunity for improved computational efficiency and robustness to erroneous data. The Hough transform (see Section 4.6.1) is an example of another algorithm which performs pixel-wise evaluation and uses a data construct similar to the truth map.

The use of an auxiliary truth map may at first appear to be superfluous but, in conjunction with pixel-wise evaluation, it is essential to permit dilation of the feasible regions, as discussed next.

FIGURE 4.4 Small locales may be missed during truth map evaluation. Each dot in this figure represents the evaluation point for a cell of the truth map. The cells are delineated here by dashed lines. The shaded area is a small locale. None of the cells of the truth map have their centre within the locale and so the truth map will be false everywhere and position decoding will fail to converge on a position estimate. This problem is avoided by dilating the feasible regions prior to intersecting them.

4.3.4 Dilation of feasible regions

The truth map provides only an approximate representation of the actual feasible regions since it is evaluated by sampling the image function at discrete points. Each cell of the truth map is evaluated at a single point (its centre) and for a small locale it may happen that none of the cell centres lie within the locale, as depicted in Figure 4.4. In this case, the entire truth map would be false and the position decoding algorithm would fail completely.

One way to avoid this dilemma is to proceed on the assumption that the target locale is within or adjacent to the cell(s) with the lowest value and use the area of minimum cell values to represent the locale. There is a more effective method, however, of ensuring that the target locale is detected during truth map evaluation. The feasible region for each pixel can be *dilated* [Haral87] in the auxiliary truth map T' prior to incorporation into the main truth map. The specific dilation procedure implemented for position decoding is discussed in Appendix B.2.

FIGURE 4.5 The feasible region for a pixel value is depicted by the darkly shaded contour interval. Those cells of the truth map with centres within the feasible region will be evaluated as true. Their cells are shown with a dark outline and crosshatching. They do not cover the feasible region entirely so they are *dilated*, to become the striped region, by including all cells which are vertical, horizontal or diagonal neighbours. The dilated region completely covers the feasible region.

Assuming that feasible regions for individual pixels are large relative to the scale of the truth map, the intersection of dilated feasible regions will always contain the locale of the target, no matter how small the locale is. Figure 4.5 shows dilation of a feasible region. Dilation is performed by applying a maximizing filter to the auxiliary truth map. The maximizing filter replaces the centre cell of a 3×3 window (non-recursively) with the maximum of the 9 cells in the window. Dilation ensures that any cell $T'(n,m)$ containing a point of the locale of the target will be a true cell for each pixel and thus the *true* cells of the final truth map will cover the target locale.

As shown in Figure 4.6, feasible regions may fail to cover the target locale completely after dilation if they are too narrow to begin with. Narrow regions can occur where an image has a very steep intensity gradient. Steep intensity gradients are rare in typical digital imagery due to the smoothing effects of the image acquisition system. Where steep gradients do exist, noise and the difficulties of target modelling may be more important than the problem of region dilation. Too narrow regions may

FIGURE 4.6 In the event that the feasible region for an individual pixel is very narrow (shaded region), it is possible that the dilated region (striped) will not completely cover the feasible region. This situation can arise when the image function of the target has a very steep intensity gradient.

be detected by observing that the auxiliary truth map T' has no true cells, or it can be avoided by increasing the resolution of the truth map so that the initial scale S_x and S_y are smaller than the reciprocal of the intensity gradient.

4.3.5 Iterated truth map resolution

If the truth map has $N \times M$ cells and the sampling window has K pixels then evaluation of the truth map requires that the image function in Equation 4.3-1 be computed (NMK) times. To achieve a precision in position estimation on the order of 10^{-2} pixels, the resolution of the truth map must be fine enough to resolve very small feasible regions; at least 500 cells on a side would be needed. With $K = 10$ pixels in the sampling window, this implies that 2,500,000 evaluations of the image function are required to determine the truth map. This can be reduced to just 2500 by using iterated truth maps, as described next.

A truth map with only 12 cells on a side may be evaluated initially to provide a coarse coverage of the target position space. Using the method of feasible region

FIGURE 4.7 The truth map is evaluated at a low resolution (left) and the region of *true* cells is then re-evaluated at a refined resolution (right). This process is iterated until the *true* region spans a substantial portion of the locale map, or until a pre-specified resolution is achieved. Truth map cell values are represented here by shades of grey, with white indicating zero and darker shades indicating more positive values. Only one cell contains zero on the left but on the right, after refinement, the shape of the target locale can be discerned.

dilation, the locale containing the true target position is assured of being covered by the true cells of this truth map but estimation of the locale centroid is not precise with such a low resolution truth map. The resolution can be iteratively improved by selecting the cells within the smallest rectangular area covering the true cells and defining a new scale and offset for the truth map so that the *resized* truth map will cover the rectangle, as depicted in Figure 4.7.

The process of resizing is repeated until either the region of true cells spans a good portion of the truth map or until a suitably fine resolution is obtained. As implemented for this study, resizing is terminated when S_x or S_y is less than 10^{-3} .

The span of the true region of the main truth map is determined after the auxiliary truth maps T' for all of the pixels have been combined into the main truth map. The four values

$$\begin{aligned}
N_H &= \max \{ n : T(n, m) = \text{true} \} + 1 \\
N_L &= \min \{ n : T(n, m) = \text{true} \} + 1 \\
M_H &= \max \{ m : T(n, m) = \text{true} \} + 1 \\
M_L &= \min \{ m : T(n, m) = \text{true} \} + 1
\end{aligned} \tag{4.3-7}$$

are used to determine whether or not resizing of the truth map is required. A truth map which is $N \times M$ cells wide will be resized when

$$(N_H - N_L + 1) < c_r N \quad \text{or} \quad (M_H - M_L + 1) \leq c_r M, \tag{4.3-8}$$

where c_r is the resizing parameter. The value $c_r = 0.3$ was selected on the basis of *ad hoc* preliminary investigations. Note that c_r must be in the range

$$\frac{1}{N} < c_r < \frac{N-2}{N} \quad \text{and} \quad \frac{1}{M} < c_r < \frac{M-2}{M}. \tag{4.3-9}$$

If c_r is too small, resizing will never occur because the individual truth map cells will be too large to satisfy the resizing condition. If c_r is too close to unity, the resizing calculations will yield the same offset and scale factors as the original truth map, resulting in an endless resizing loop.

Resizing is performed by defining new scale and offset parameters (S'_x, S'_y) and (A'_x, A'_y) , using

$$\begin{aligned}
S'_x &= S'_y = \max \{ (N_H - N_L + 1), (M_H - M_L + 1) \} / (N - 1), \\
A'_x &= A_x + N_L S_x, \quad \text{and} \\
A'_y &= A_y + M_L S_y,
\end{aligned} \tag{4.3-10}$$

where (S_x, S_y) and (A_x, A_y) are the old scale and offsets respectively.

Equation 4.3-7 ensures that the resized truth map overhangs the true cells of the old truth map by one cell all around its perimeter. This was done in early versions of the position decoding algorithm as a simpler alternative to feasible region dilation. Later, feasible region dilation proved to be a better method of ensuring coverage of the locale by the truth map, despite its computational complexity. Nevertheless,

FIGURE 4.8 The centroid of the target locale is estimated by the centroid of the true region of the truth map. This estimate can be in error by half a cell width. As an example, an oblong locale (shaded) is shown, covered by a dilated region of true cells (striped). The centroid of the locale and the centroid of the true region are indicated with white dots, which are about half a cell width apart. The black dots and faint lines indicate the centre and boundaries of the truth map cells.

Equation 4.3-7 was retained for locale resizing because it adds to the robustness of the algorithm when noise is present.

The region defined by the true cells of the truth map provides an approximation to the target locale. The blockiness of this region causes some error in the estimation of the locale centroid. A rough (heuristic) estimate of the magnitude of this error is one half of the size of a cell of the truth map, as illustrated in Figure 4.8. This error is added to the fundamental position estimation error. A locale which spans n cells represents a fundamental position uncertainty in the x direction of about nS_x , where S_x is the scale of the truth map. This uncertainty is increased by about $\frac{1}{2}S_x$ (i.e., by a factor of about $1/2n$) due to errors in the locale centroid estimation. For example, a locale which spans about $1/3$ of a truth map having 15 cells on a side can introduce 10% additional error to the position estimate.

4.3.6 Partial evaluation

Savings in computational complexity can be achieved by using a technique of early termination of the evaluation of cells of the truth map. On the k^{th} iteration (Equation 4.3-6) only those cells with $T^{(k)}(n, m) = 0$ need to be evaluated since all others fail to be in the feasible region of some of the pixels $\hat{p}_0, \dots, \hat{p}_{k-1}$.

Later, in Section 4.6.4, it is shown that this technique can reduce the computational effort by as much as one third.

Other methods of reducing the computational effort are available but, because of their complexity, limited benefit, or disadvantageous side effects, they are omitted from the implementation of the algorithm used for this study. For example, truth map resizing (Equation 4.3-10) can be made more efficient by rescaling the x and y axis independently, but this destroys redundancy in the results, which is used to help monitor the integrity of the software. Another example is the possibility of resizing the truth map, before *all* pixels have been processed, whenever the feasible region in the main truth map becomes small. Doing this would complicate the software and make it difficult to verify consistency of the feasible regions as determined at various truth map resolutions, again compromising the capabilities for integrity checking.

4.4 Position Decoding with Noise

When there is noise in the image of the target, the method of intersecting feasible regions may fail to provide the correct locale, or any locale at all for that matter. To avoid such difficulties, the objective must be to find those truth map cells which contain the minimum value, rather than to seek the cells containing the value 0. Stated differently, the objective must be to find the locale(s) for which the observed digital image is most likely, and the truth map must indicate the likelihood or a region.

4.4.1 Noise and the likelihood function

For a given target position (x, y) , let the probability of a pixel (i_k, j_k) having the observed value $\hat{p}_k$ be $P_k(\hat{p}_k; x, y)$. The probability of obtaining an observed set of pixel values $\{\hat{p}_k\}$ when the target is at (x, y) is then given by the joint distribution

$$P(\{\hat{p}_k\}; x, y) = \prod_k P_k(\hat{p}_k; x, y), \quad (4.4-1)$$

assuming that the pixel values are statistically independent. For a set of observed pixel values $\{\hat{p}_k\}$ the *maximum likelihood estimate* (MLE) for the target position is obtained by maximizing $P(\{\hat{p}_k\}; x, y)$ as a function of x and y .

The function $P(\{\hat{p}_k\}; x, y)$, as function of x and y , is not a probability distribution; it is referred to as a (joint) *likelihood function*. Maximizing this likelihood function does not assure that the most probable position has been selected, nor that the mean square error has been minimized. Nevertheless, optimal estimation in the presence of noise is often defined as the MLE. This practice is encouraged by the fact that, for a Gaussian distribution, the MLE has the minimum mean square error and, if the a priori target position is uniformly distributed, the MLE has maximum probability [Bick77]. Accordingly, the MLE is considered the *optimal estimate*.

In Figure 4.9 the likelihood function $P_k(\hat{p}_k; x, y)$ is depicted for noise added after quantization. Notice that the function P_k is constant on the contour intervals of the image function; consequently, the joint likelihood function $P(\{\hat{p}_k\}; x, y)$ is constant for all points (x, y) in the same locale since all such points yield the same (pre-noise) pixel values. When noise is added before quantization, the likelihood function need not be piecewise constant and may vary continuously within a locale.

The position decoding algorithm, as used here, presumes that the likelihood function is constant on contour intervals of the image function. This assumption is made because it simplifies computation, is consistent with the analogy between the algorithm and signal decoding in communications theory, and perhaps most importantly, it focuses the investigation on the target model and the characteristics of locales rather than the noise statistics.

FIGURE 4.9 When noise is added after intensity quantization has been performed, each contour interval of the image function has a constant likelihood value. The likelihood function for a single pixel is depicted here, with darker areas being less likely. When noise is added before quantization, the likelihood may vary continuously.

When the likelihood function is presumed constant on contour intervals of the image function, it can be written as $P_k(\tilde{p}_k; p_k)$ with $\tilde{p}_k$ being an observed pixel value and p_k being the pixel value obtained from a noise-free image function. For a set of observed pixel values $\{\tilde{p}_k\}$ the joint likelihood function is given, as in Equation 4.4-1, by the product of the individual likelihood functions and so it too can be expressed as a function of the pixel values, as $P(\{\tilde{p}_k\}; \{p_k\})$. The joint likelihood function assigns a likelihood to every locale (*i.e.*, every legitimate digital representation in image space of the target). Note that every set of image function values $\{p_k\}$ must be a locale, whereas, a set of observed pixel values $\{\tilde{p}_k\}$, may be any point in image space.

4.4.2 The cost function and feasible regions

A cost function C_k associated with the observed pixel value $\tilde{p}_k$ is defined in terms of the log of the likelihood function by

$$C_k(x, y; \tilde{p}_k) = -\log \left(\frac{P_k(\tilde{p}_k; x, y)}{P_k(\mathcal{I}_{k,jk}(x, y); x, y)} \right). \quad (4.4-2)$$

For a set of pixel values $\{\tilde{p}_k\}$, the overall cost function is given by

$$C(x, y; \{\tilde{p}_k\}) = \sum_k C_k(x, y; \tilde{p}_k). \quad (4.4-3)$$

As with the likelihood function, the cost functions C_k are constant on the contour intervals of the image function, and the joint cost function C is constant on the locales of the target. It is presumed that each C_k is minimal where $\tilde{p}_k = \mathcal{I}_{i_k j_k}(x, y)$ and increases as the image function differs from $\tilde{p}_k$.

Determining the exact cost functions is difficult but, as an alternative, the cost functions can be approximated by arbitrary functions which attain their maxima at the same values of (x, y) as the cost functions do. Simple, computationally efficient functions such as (the negatives of) $|\tilde{p}_k - \mathcal{I}_{i_k j_k}(x, y)|$ and $(\tilde{p}_k - \mathcal{I}_{i_k j_k}(x, y))^2$ accomplish this. Alternatively, simple table-lookup procedures may be used. The implementation of position decoding used for this study uses the first approximating function,

$$C(x, y) = \sum_k |\tilde{p}_k - \mathcal{I}_{i_k j_k}(x, y)|, \quad (4.4-4)$$

which corresponds to the l_1 or *Hamming* norm in image space. The l_1 norm has the distinct advantage of requiring only one subtraction to compute per pixel.

The feasible region $R_{\tilde{p}_k}(c_k)$ is dependent upon a threshold parameter c_k ,

$$R_{\tilde{p}_k}(c_k) = \{ (x, y) \in D : C_k(x, y; \tilde{p}_k) \leq c_k \}. \quad (4.4-5)$$

The feasible region for a set of pixel values $\{\tilde{p}_k\}$ is defined as

$$R_{\{\tilde{p}_k\}}(c) = \left\{ (x, y) \in D : \sum_k C_k(x, y; \tilde{p}_k) \leq c \right\}. \quad (4.4-6)$$

The region of maximum likelihood (or minimal cost) is given by the smallest value of c such that $R_{\{\tilde{p}_k\}}(c)$ is not empty. When the cost function is continuous, this reduces to the conventional maximum likelihood estimate which may provide a single point rather than a region.

In the presence of noise it is not always possible to obtain a feasible region for a set of pixels by a sequence of set intersections, as it is when there is no noise. With

noise, it is necessary to consider unions of intersections of feasible regions, since

$$R_{\{\tilde{p}_k\}}(c) = \bigcup_{\sum c_k = c} \left\{ \bigcap_k R_{\tilde{p}_k}(c_k) \right\}. \quad (4.4-7)$$

The region determined by $R_{\{\tilde{p}_k\}}(c)$ will be referred to as the *aggregate feasible region*. The use of the term is motivated by the fact that the region must be a union of locales. Noise tends to cause aggregation of locales, resulting in increased position uncertainty.

4.4.3 The truth map revisited

When there is noise, the binary *true* and *false* logic of the truth map must be changed to integer or floating-point values. The cell values $T'(n, m)$ for the auxiliary truth map (Equation 4.3-5) are defined by the cost function as

$$T'(n, m) = C_k(x_n, y_m; \tilde{p}_k). \quad (4.4-8)$$

The feasible region dilation procedure is applied to this truth map according to Appendix B.2. The main truth map is updated, as before, using

$$\begin{aligned} T^{(k+1)}(n, m) &= T^{(k)}(n, m) + T'(n, m), \\ &= \sum_{i=0}^k C_i(x_n, y_m; \tilde{p}_i). \end{aligned} \quad (4.4-9)$$

4.4.4 Scattered locales

Locales are defined in both position space and image space in terms of a noise-free digital image. When noise is present, the observed pixel values $\{\tilde{p}_k\}$ define a point $\tilde{p}$ in image space which may or may not correspond to a locale. If it does correspond to a locale then this locale will have the maximum likelihood and all others will have a lower likelihood. In this case, the target locale can be obtained by a series of intersections of feasible regions. as for the noise-free situation, by applying

Equation 4.4-7 with $c = c_k = 0$. The optimal position estimate is then obtained as the centroid of this locale in position space.

Locales are sparsely distributed in image space (see Section 3.3.1) hence $\vec{p}$ is not assured of being a locale. If $\vec{p}$ is not a locale then those locales closest to it (in the image space l_1 -norm) will have the greatest likelihood (lowest cost). In communications theory, signal codes are designed to be well separated in their code space but this is not possible for digital representations of images. It has been shown (Section 2.4.3) that locales form a sparse connected mesh in image space. As a result, there are often several locales which are maximally likely, none of which have a cost of zero. As is depicted in Figure 4.10, these locales need not be adjacent. The union of these locales, which generates the aggregate feasible region, may consist of locales which are scattered in position space (albeit within the vicinity of each other). This effect can be seen in Figure 4.11, which shows an example of scattered locales based on data which was obtained during a simulation test of position decoding. Note that scattered locales are a result of noise and are not caused by the finite resolution of the truth map. The centroid of the aggregate locale is a good estimate of the target position, but it may not be the optimal one, as discussed next.

4.4.5 Locale weighting

When the locales of maximum likelihood are scattered, the question arises: what point in position space should be used as the estimate for the target position? One approach is to treat the most likely locales as a single aggregate locale and compute the centroid of this aggregate locale as the position estimate. Since any target position in the aggregate locale is equally likely, their union defines a region of probabilistically equivalent target positions. The centroid of this region provides the minimal mean square error position estimate *if* the target is known to be within the region.

It is possible however, that the target is not actually positioned in the aggregate locale, but rather, in some less likely locale. When this is the case, the centroid of

FIGURE 4.10 In the presence of noise, observed pixel values may not correspond to a "proper" digital representation of the target; that is, a locale in image space. Image space is depicted here (schematically) as an array of points with locales shown shaded. Adjacent locales are connected by a line, representing the meshwork of locales in image space. The larger dot represents an observed point which has three locales equidistant from it. These locales have maximum likelihood and are not adjacent to one another.

the aggregate locale does not provide the minimal mean square error. A more general solution is provided by the position estimate $(\hat{x}, \hat{y})$ with

$$\hat{x} = \sum_{\ell} \iint x_{\ell} e^{-C(x, y; \{\tilde{p}_k\})} \chi_{\ell} dx dy \quad \text{and} \quad \hat{y} = \sum_{\ell} \iint y_{\ell} e^{-C(x, y; \{\tilde{p}_k\})} \chi_{\ell} dx dy \quad (4.4-10)$$

where the summation is over *all* locales, (x_{ℓ}, y_{ℓ}) is the centroid of locale ℓ , $C(x, y; \{\tilde{p}_k\})$ is the cost function for target position (x, y) given the observed pixel values $\{\tilde{p}_k\}$, and χ_{ℓ} is the characteristic function of the locale ℓ . This estimate is simply the centre of mass of the likelihood function in position space. It is a theoretically interesting estimate but it is presented here only for completeness since it is computationally expensive to implement. Notice that the region of maximum likelihood for a continuous cost function corresponds to the conventional maximum likelihood estimate (MLE), which is usually a single point, and that the MLE point does not generally provide the position estimate with the minimum mean square error.

5	5	5	5	5	5	5	5	4	4	3	2	2	4	4
5	5	5	5	5	4	4	4	3	3	2	2	2	4	4
5	5	4	4	4	4	4	3	3	2	2	3	3	4	4
4	4	4	4	4	3	3	3	2	3	3	3	3	4	4
3	3	4	4	3	3	4	4	3	3	3	3	3	4	4
3	3	2	2	4	4	4	3	3	3	3	3	3	4	4
4	3	4	3	3	3	3	3	3	3	3	3	3	4	4
4	4	4	4	4	3	3	2	2	3	3	3	3	4	4
4	4	4	4	4	3	3	3	3	3	2	3	3	4	4
5	4	4	4	3	3	3	3	3	3	3	3	4	4	5
5	5	4	3	3	3	3	3	3	4	4	4	4	5	5
5	5	4	4	3	4	4	4	4	4	4	4	4	5	5
4	4	5	5	5	5	4	4	4	4	4	4	4	5	5
5	4	4	5	5	5	5	5	4	4	4	4	4	5	5
4	4	4	4	5	5	5	5	5	4	4	4	4	5	5

FIGURE 4.11 In the presence of noise, the area of maximum likelihood may consist of a collection of non-adjacent locales. An example is shown here, based on cell values observed in the final truth map of the position decoding algorithm applied to the Gaussian dot during the simulation studies. The numbers correspond to the relative cost of the cells (using the l_1 -norm). The region of minimal cost (maximal likelihood) is outlined and is treated as a single aggregate locale for the purpose of estimating the target position.

4.4.6 The tolerance parameter: backtracking

As an algorithm for solving a cost minimization problem, position decoding uses a *branch and bound* procedure [Lawle85]. The problem of finding the lowest cost region of target positions is partitioned into a set of such problems of each of the regions of equal cost in the main truth map $T^{(k)}$. Within each region, a lower bound on the final cost is given by the value stored in $T^{(k)}(n, m)$. This bound is used to select the region(s) which is further evaluated. The higher cost regions are discounted from further evaluation by the technique of partial evaluation.

In the absence of noise, only the least costly region need be evaluated in order for the branch and bound procedure to lead to the lowest cost region. This is expressed formally in Equation 4.3-2, which states that the globally lowest cost region is the intersection of the lowest cost regions for the individual pixels.

FIGURE 4.12 Backtracking is a search strategy for finding a path to the minimal cost solution. On the left, a single search path begins at the root (bottom) of a tree and follows the lowest cost branches. When the accumulated cost becomes too expensive to be optimal the search backtracks to the last viable branch point (marked with a star). An alternative, which position decoding emulates, is to search several paths in parallel, as shown on the right. At each level of the tree, several of the lowest cost paths are retained in the search repertoire, allowing instant backtracking to the next-best paths at all times. Using more parallel paths increases the chances of finding the optimal solution but requires more computation.

In the presence of noise, this path of lowest cost regions through the branch and bound procedure may not lead to the globally lowest cost. To ensure that the global minimum is found, *backtracking* must be used. Backtracking is discussed, for example, in [Wells71]. The term backtracking is most literally applied to a search strategy which follows a single path, usually making branch choices on the basis of minimal cost, and retracing the search path to the last viable alternative when the cost gets too high (see Figure 4.12). This process requires a mechanism for remembering which nodes have previously been tried and what their respective costs were. Backtracking can proceed until all possible solutions have been investigated, or until some satisfactory cost solution has been achieved.

Backtracking is accomplished in position decoding by considering several candidate solutions in parallel. When the current best selection becomes more costly than the previously-rated next best alternative, the next best alternative is available among the current candidates. This avoids retracing the path to the last viable node.

The strategy of parallel evaluation is theoretically approximate in that it is not assured of finding the optimal solution. It is possible that all of the evaluated candidates eventually exceed the true minimal cost, unless the parallel evaluation is extended to all possible alternatives.

Parallel evaluation is implemented in position decoding by retaining all cells of the truth map, which are within a tolerance C_t of the lowest cost cell, for further evaluation. This modification to the partial evaluation technique increases the computational effort of position decoding. In Section 4.6.4 the computational cost of the tolerance parameter is investigated.

Criteria for determining a suitable level for the tolerance threshold are not rigorously determined. Within the position decoding software (for listings of the program code, see Appendix C), there is a mechanism for dynamically increasing the tolerance, above a minimal level, based on the range of cost values in the truth map. Selection of the minimal level is chosen in an *ad hoc* manner based on the noise level. For the simulations studies, large tolerances are used because the associated computational costs are acceptable.

4.5 Implementing Position Decoding

A description of position decoding, as a practical algorithm, would not be complete without at least a cursory discussion of its implementation in real applications. This section presents heuristic discussion of some issues relevant to position decoding as a practical position estimation algorithm. The intent is not to be complete or rigorous, but rather to broach each subject in a manner which may provoke and initiate further consideration of implementation issues.

4.5.1 Target model parameter selection

In applications of high precision position estimation it may be necessary to establish parameters of the target model dynamically; that is, some details of the target model may need to be derived from the observed digital image prior to commencing position decoding. Temporal and spatial variations in illumination, for example, may be difficult to incorporate into a target model and so require that the grey level profile of the target be estimated dynamically.

Thorough analysis of techniques to determine model parameters is well beyond the scope of this study but the need for such pre-processing must be recognized. The use of position decoding with poor target model parameter estimation will compromise the benefits of high precision which the algorithm offers.

Theoretically, unknown target model parameters can be included as extra coordinates in an abstract target position space. Position decoding in more than two-dimensions, however, is computationally expensive and may be impractical, so preliminary estimation of target parameters may be essential.

When selecting targets suitable for precise position estimation, consideration should be given to preliminary parameter estimation. For example, grey-scale calibration targets and position estimation targets might be used in conjunction with each other. Calibration targets benefit from local constancy to provide statistical averaging while position estimation targets benefit from image variation to improve position information content. If a target possessing both features is used, active pixel selection (Section 4.5.3) can be applied to select the most relevant target portions for position decoding.

4.5.2 Target model format

The target model is external to the position decoding algorithm. The target model may be composed in many ways and modified without altering the main position

decoding software. Two methods were used during this study: functional specification and table-assisted computation.

The method of functional specification was used for Gaussian dot targets in the simulation studies. The target model is provided by an image function defined by explicit evaluation of the function

$$\mathcal{I}_{ij}(x, y) = \lfloor Ae^{-(x-i)^2-(y-j)^2} \rfloor. \quad (4.5-1)$$

Alternative, more efficient methods exist, such as tabulating the radii r_k of the contours of the Gaussian, with $r_k^2 = \log(A/k)$. The quadratic function $(x-i)^2 + (y-j)^2$ could then be compared against these radii to determine the appropriate image function value. Such a technique, referred to as table-assisted computation, eliminates unnecessary repeated computation of the exponential function.

The binary edge target model uses a table assisted computation in the simulation studies. Whenever the truth map is initialized or rescaled, a table of $\sin \theta_n$, $\cos \theta_n$ and b_m values are computed for the orientations θ_n and offset values b_m corresponding to the truth map indices (n, m) . The image function for pixel k (with indices (i_k, j_k)) and truth map cell (n, m) is then computed as

$$\mathcal{I}_{i_k j_k}(\theta_n, b_m) = j_k c(n) - i_k s(n) - b(m), \quad (4.5-2)$$

where $c(\cdot)$, $s(\cdot)$, and $b(\cdot)$ are the cosine, sine and b tables respectively.

Other methods of representing the target model are possible. An image function might be stored as a sampled raster which is interpolated to obtain evaluation at arbitrary points. The relationship between the density of the raster and the resultant estimation precision has not been evaluated, but this method allows the target to be specified in a very flexible manner. It is an appealing method for applications in which the target is described only in terms other digital images, as is the case for stereo image parallax estimation or target tracking applications.

FIGURE 4.13 A binary edge is represented by a list of pairs of pixels. Each pair consists of a dark and a light pixel, vertically adjacent to each other, which define the edge boundary in a vertical column. The pixels are ordered left to right, bottom to top, as indicated by the numbering.

4.5.3 Selecting active pixels

The input data for position decoding is an ordered list of pixel values and associated pixel indices. Position decoding does not require a two-dimensional array of pixel values. It is possible to select the most relevant pixels from an image for position estimation and disregard others. The pixels selected for position decoding are called *active pixels*. The choice of active pixels effects the precision, speed and robustness of position decoding.

A complete set of active pixels is defined for a binary edge, for example, by selecting vertically adjacent light and dark pixels in each column. Such a pixel list is used for the simulation studies of binary edges in Chapter 5 and is illustrated in Figure 4.13. It provides all of the available edge position information to position decoding.

A more complicated example is illustrated in Figure 4.14, where a cross shaped target is represented by a set of contours. If the position space for this target is a unit square then the pixel values of each of the shaded pixels, and no others, will

FIGURE 4.14 Shown here is a series of contours which represent a reseau-cross or a 4-sided star. The lightly shaded pixels will change value as the target centre moves within a unit square of its illustrated position. A pixel list which includes all 56 shaded pixels will contain all target position information. A smaller list consisting of the symmetric set of darkly shaded pixels will have less position information but may be less vulnerable to modelling errors.

vary with the target position. For this example, there are 56 shaded pixels and to be sure of obtaining all of the position information in a digital image of the cross, all 56 pixels must be used for position decoding. To reduce computational effort it may be necessary to make a careful choice of a few pixels rather than using them all. The choice of pixels for an active pixel list can be based on the robustness (sensitivity to noise and modeling errors) and position information (image gradient) of individual pixels.

Some pixels may be sensitive to modeling errors and should be omitted from the active pixel list. For example, uncertainty in the target model in the vicinity of the tips of the cross, perhaps due to limited knowledge of the sensor response pattern or unspecified rounding at the tips, may be grounds for excluding the pixels near the tips of the cross.

The central pixels have little intersection with the contours of the target and so they will provide less position information than other pixels. For computational efficiency, these can be eliminated from the active pixel list.

The effects of errors in the modeled target size or the overall intensity profile can be reduced by selecting active pixels symmetrically across the target. Using pixels on only one side of the target leaves position decoding vulnerable to substantial position estimation errors from poor modeling.

Symmetry of active pixels can be used to increase computation speed as well. Placing pixels in the active pixel list so that opposing pixels are sequential in the list can result in the feasible region in the truth map being reduced in size more quickly. Rapid reduction in the feasible region results in fewer truth map cell evaluations when partial evaluation is used (Section 4.3.6). This technique was used for the active pixel list of the Gaussian dot target in the simulation studies.

Based on the above heuristics, the darkly shaded pixels are a good choice for a computationally economical active pixel list. When all 56 pixels are used, the lineal bound on locale density for the target is nominally 76, as can be seen by applying the methods of Section 3.3.3 and Figure 3.6. Figure 4.15 is a sketch of the locale map for the set of eight darkly shaded pixels. When only these eight pixels are used, the lineal bound is nominally 18, as can be estimated by counting the horizontal density of locale boundaries in Figure 4.15. In this example, the computational savings ($56/8 = 7$) exceed the loss of precision ($76/18 \approx 4.2$) when a smaller set of carefully chosen pixels are used.

A pre-defined template of active pixels can be used to select pixels from a digital image. This allows the time consuming computations to estimate local image intensity gradients, and other characterizations of good active pixels, to be performed in advance of image acquisition and processing. The template is determined from the target model and is applied to the digital image based on the initial (approximate) estimate of the target position.

FIGURE 4.15 A locale map is generated from those contours of the cross shaped target which intersect the eight darkly shaded pixels shown in Figure 4.14. The position space of this locale map is one unit pixel square. The contours of the cross shaped target are approximated here with straight lines. The nominal horizontal locale density in this locale map is 18. If all 56 shaded pixels in Figure 4.14 were used to generate a locale map, then the nominal horizontal locale density would be 76 (not represented).

In short; too many active pixels lengthen the computation time and too few reduce the available position information. Pixels which are symmetrically opposed and in areas of high image gradient are desirable, while pixels in areas of poor target model definition can reduce tolerance to modeling errors.

4.5.4 Cost function selection

The cost function used for the simulation tests in this study is the l_1 -norm given in Equation 4.4-4. This function was selected because of its computational simplicity and because, in a noise free situation, it provides the optimal solution in a straightforward manner.

In real applications, it may be possible to improve on the l_1 -norm cost function. A cost function whose slope depends on the image intensity may be more effective since noise levels from CCD imaging sensors increase with the pixel intensity, due to *shot noise* in the sensor ([Jenki87], [Gagli76]). Reduced sensitivity to errors at higher image intensity can be achieved by reducing the slope of the cost function at larger image function values. Furthermore, reducing the sensitivity to errors as the

horizontal image intensity gradient increases may provide additional improvements since errors due to timing instability during digitizing increase with the horizontal image gradient [Dahle87].

Although these heuristics are based on the noise characteristics and the theory of maximum likelihood estimation, they do not lead to *exact* cost functions. In practise, cost functions such as

$$C_k(x, y; \hat{p}_k) = \left| \frac{\hat{p}_k}{\mathcal{I}_{i_k, j_k}(x, y)} - 1 \right| G(i_k, j_k), \quad (4.5-3)$$

derived from heuristic considerations, might be all that is available. In this contrived cost function, the first term corresponds to the l_1 -norm scaled by the intensity and G represents the reciprocal magnitude intensity gradient. Cost functions such as this may provide better estimation precision in physical applications than the simple l_1 -norm cost function, which was used in the simulation studies.

The main purpose of the cost function is to provide the best estimate based on some knowledge of the noise statistics. In the following section, knowledge of the noise distribution is applied with robust statistical estimation techniques [Bicke77], wherein outlying data values (lying on the tails of the distribution) are treated specially.

4.5.5 Pixel defects

It is not uncommon for CCD imaging sensors to have defective sensor sites which generate anomalous pixel values. The location of these defective sites can be determined in advance, during calibration of the image sensing array, and these specific pixels can be discounted during subsequent processing. Alternatively, the position decoding algorithm has the capability of detecting and excluding deviant pixel values as it evaluates the truth map. The ability to detect and adjust for deviant pixel values is made possible by the explicit use of a target model.

As the auxiliary truth map T' is evaluated, the lowest cost can be used as an indication of whether or not the pixel value is defective. If the lowest cost is higher

than, say, that associated with three standard deviations of the noise distribution, then the pixel can be excluded from the main truth map on the premise that it is defective.

Another technique drawn from the theory of robust estimation is to exclude the highest cost pixel, on a cell by cell basis, from the truth map evaluation. This can be accomplished by defining another truth map $T_{\max}$ which remembers the largest costs for each cell in the auxiliary truth map. Initialized to zero, it is adjusted after the auxiliary truth map T' has been evaluated for each pixel, using

$$T_{\max}(n, m) \leftarrow \begin{cases} T'(n, m) & \text{if } T'(n, m) > T_{\max}(n, m), \\ T_{\max}(n, m) & \text{otherwise.} \end{cases} \quad (4.5-4)$$

The main truth map is computed as usual but the region of minimal cost is based on the difference $(T - T_{\max})$ between the main and maximum truth maps.

Flexibility in the definition of the cost function and the image function permits such adaptations to be easily incorporated into position decoding, thereby allowing information about the noise and the target to be incorporated into the position estimate.

4.6 Evaluating Position Decoding

This section considers various performance related aspects of the position decoding algorithm. The discussion is kept brief and general since detailed simulation results are presented and discussed later in Chapter 5. The algorithm has been described, component by component, in the previous sections: here, position decoding is discussed as a complete algorithm, with regard to uniqueness, flexibility, stability, optimality and computational complexity.

4.6.1 Unique features of position decoding

There are many things which make position decoding unique:

- flexible cost function definition
- flexible target function definition
- abstract target position space
- theoretical optimality
- target locale size determination
- pixel-wise evaluation
- region swelling
- iterated resolution
- partial evaluation

Other algorithms may incorporate some of these characteristics but none have them all.

It might be argued that position decoding is basically the same as template matching, but the similarities are tenuous. Template matching is the simplest of all recognition, detection and positioning algorithms [Hall79]. If template matching is extended beyond its conventional definition, so as to apply a continuous template (such as the image function) to a discrete image, it can theoretically provide the optimal estimation of translational target position. Computer implementation of template matching to achieve such high precision subpixel precision, on the other hand, is not straightforward. Efforts have been made to make template matching efficient by sequential similarity detection ([Barne72] and [Munte81]) and hierarchical resolution techniques ([Rosen77], [Tanim81] and [Wong78]), but they address pixel, rather than subpixel, precision and are confined to position-wise evaluation. Unlike position decoding, these techniques cannot take advantage of region swelling to ensure successful identification of small target locales.

Search techniques, such as template matching, are done in the coordinate space of the digital image rather than a generalized target position space. Because of this, they have difficulty doing things like solving for the orientation and position of an edge simultaneously. Position decoding, on the other hand, can do this in a straightforward manner.

The Hough transform [Hough62] is an algorithm which has several similarities to the position decoding algorithm. Like position decoding, the Hough transform is computed pixel-wise, it uses an *accumulator array* which is similar to the truth map of position decoding, it is dependent upon a formal target model, and it requires prior identification of *feature pixels* which are similar to the active pixels described in Section 4.5.3. The Hough transform can be used for arbitrary target shapes and is capable of treating generalized position parameters [Balla81], [Merli75]. Recently, the potential precision of the Hough transform has been improved by a technique for iterative refinement of the accumulator array resolution [Illin87] and by analysis of the parameter estimation uncertainty due to quantization in image and parameter space [Veen81].

Parameter estimation with the Hough transform is based on the existence of a point $\mathbf{x}$ in parameter space which satisfies $f(\mathbf{x}, p_k) = 0$ for a continuous functions f and a set of feature pixels $\{p_k\}$. Position decoding, on the other hand, is based on a *region* of points which satisfy a discrete-valued functional relationship. The Hough transform is well known and has proven useful, particularly for edges, however it lacks the optimality and thorough quantization-error analysis available for position decoding.

Position decoding is uniquely appropriate as a companion algorithm for the theory of locales. Its approach to position estimation in quantized images is unique and theoretically rigorous. Its effectiveness as both a theoretical tool and a practical estimation technique brings theory and practice closer together in the field of digital image metrology.

4.6.2 Flexibility and stability

Position decoding is applicable to any target form, whether binary or grey-scale, with any sort of position parameters, provided that a suitable target model can be defined. The algorithm has the same form of implementation independent of the specific target shape or position space coordinate system. Large targets can be handled efficiently by selecting a (short) list of active pixels which keeps the computational complexity manageable.

The algorithm treats each pixel individually and provides many possibilities for tuning and refining the algorithm for specific applications. For example, pixels occurring in regions of maximum image gradient can be processed preferentially. If radiometric calibration of a CCD image sensor provides individual pixel sensitivity and bias then these can be accurately integrated into the algorithm without introducing the rounding errors which are inherent to image pre-compensation calculations. Various sorts of front-end intelligent pixel selection techniques can be employed when selecting the active pixel list, without impinging on the main position decoding processing sequence.

A particularly useful characteristic of the algorithm is its ability to handle *partially occluded* targets. The active pixel list allows front-end processing to exclude portions of the target which are occluded or unsuitable, thereby deducing a position estimate based on the best available information.

Experiments with the position decoding algorithm have been restricted to simulation situations. Practical demonstration of the algorithm using data obtained by a camera and physical target have not been performed, but the simulation results demonstrate that the algorithm is tolerant of modeling errors. The gradual degradation in position decoding performance with target modeling errors is reported in Section 5.3. Within the limited class of model errors considered, the position decoding algorithm is observed to be stable and robust.

4.6.3 Optimality

Position decoding is theoretically optimal in terms of mean square error in the absence of noise, as shown in Section 3.1. Position decoding has not been proven to be optimal when noise is present, but it closely resembles maximum likelihood estimation and performs well in comparison to other algorithms.

Theoretical optimality in the absence of noise is demonstrated with simulations in Section 5.5.4. It is shown that the estimation errors of position decoding for binary edges coincide with those predicted analytically by other researchers.

The comparative performance study in Section 5.2.3 shows that position decoding performs well in the presence of noise, although there is no definitive benchmark against which the possible sub-optimality of the algorithm in the presence of noise can be determined.

4.6.4 Computational complexity

Tests were done to investigate the relationship between the size of the truth map and the computational effort required for position decoding. A quadratic functional dependence was found between the width of the truth map N and the number of cell evaluations, as shown in Figure 4.16. The quadratic function has the form

$$\text{Cell evaluations} = 6.9(N - 4.8)^2 + 1406, \quad (4.6-1)$$

based on a Gaussian dot target with amplitude of $A=30$, no noise added, and a 3×3 patch of active pixels. The curve in the figure has a minimum at $N=5$, but such a small truth map will not clearly resolve the shape of locales and will have difficulty detecting small locales. For the simulation tests, a truth map size of at least $N=12$ was used.

The technique of partial evaluation with a tolerance parameter is used to reduce the computational effort. The tolerance parameter C_t determines what range of truth map values are retained for further evaluation as successive pixels are incorporated

FIGURE 4.16 The number of truth map cell evaluations is plotted as a function of the size of the truth map. The data is based on a 3×3 Gaussian dot target with no noise. The data is approximated by a quadratic function.

into the truth map. When $C_t=0$, only the most feasible region being retained. As C_t is made larger, more cells of the truth map are evaluated on every iteration. The Gaussian dot target was used to examine the effect of C_t on the amount of computation required for position decoding in the absence of noise. Figure 4.17 shows the number of truth map cell evaluations as a function of C_t . The Gaussian dot target was used with amplitude $A=30$, a truth map width of 15, and 9 active pixels as above. With only the most feasible region retained ($C_t=0$), the computational effort is about one third that when all feasible regions are retained ($C_t \rightarrow \infty$).

In the following chapter, the simulation studies apply the position decoding algorithm to noise-free image data as well as to noisy image data. A listing of the program code for the position decoding algorithm, as well as the associated modules defining the Gaussian dot and binary edge target models, are presented in Appendix C. Listings are also presented there for the diagnostic routines which display the evolution of the truth map as position decoding executes.

FIGURE 4.17 As the tolerance parameter C_t is increased, more truth map cell evaluations are required. The number of evaluated cells as a function of C_t is shown here for a 3×3 Gaussian dot target with an amplitude of 30, using a truth map of width 15. The number of cells evaluated (i.e., the computational cost) varies by a factor of about 3 for this example.

Chapter 5

SIMULATIONS

The position decoding algorithm provides a bridge between theory and experiment, making it possible to investigate optimal estimation by both analytical and simulation techniques. With simulated image data and the position decoding algorithm, the theoretical analysis of precision in noise-free digital images can be extended to noisy images so that the effects of noise on precision can be investigated in the context of the locales theory.

The simulations reported in this chapter are intended to support or illustrate the theory of locales, but there are various avenues of investigation suggested by the simulation data itself.

Some of the characteristics of the simulation results, particularly in regard to noise, are not yet fully explained and in these cases, heuristic or speculative discussions are presented. The formal theory is limited in regard to noise, as yet, but the simplicity and consistency of the data relationships is strong evidence that the general approach is an appropriate one for arriving at a better understanding of geometric precision in digital images.

This chapter begins with a description of the general framework of the simulation studies. Following this is a performance comparison between position decoding and three other position estimation algorithms. The performance comparisons use the Gaussian dot target and simulate noisy data as well as noise-free data. These simulations also demonstrate the effects of intensity resolution on position estimation precision.

The robustness of the position decoding algorithm to target modelling errors is demonstrated by introducing errors in the target amplitude.

The effects of spatial resolution on the performance of position decoding is characterized using a binary edge target. These simulations also demonstrate the application of position decoding to non-translational position parameters, namely the offset and orientation of an edge. A cursory discussion of the maximum estimation errors is presented and the dependence of precision on edge orientation is discussed. Optimality of position decoding for one-dimensional estimation is demonstrated by comparison with analytical results from the literature.

In the final section of this chapter, various aspects of noisy image data are investigated. The stability of position decoding in the presence of additive noise is demonstrated. The precision of position decoding is determined as a function of the noise level. It is shown that precision is essentially a linear function of the noise level and that the noise level in an image may be expressed as a noise-equivalent position parameter. The effects of noise on the anisotropic (orientation dependent) character of position estimation precision for binary edges is discussed. The results of this section, concerning noisy image data, are of a preliminary and heuristic nature.

For convenience, the numerous figures containing graphical data are collected together at the end of this chapter rather than interspersed with the text.

5.1 General Framework

5.1.1 Computational environment

A variety of computer resources were used for this project, including VAX-11, PDP-11, IBM PC/AT compatible, and Macintosh computers. The VAX-11 and PDP-11 computers were used for preliminary analysis and simulations. Since real-time performance was not an issue, an IBM-PC/AT compatible was used, once it became available, for most of the simulations, computations, and detailed analysis. The Macintosh computer was used for interactive data analysis and report preparation.

5.1.2 Target models

A target model is provided by the continuous entity function (Section 2.1.4) which is sampled and quantized to obtain image pixel values which are always greater than or equal to zero. The background pixels are presumed to be zero while the target pixel values range from zero to some maximum amplitude. Two target types are used in the simulations: a Gaussian dot (Section 2.2.5) and a binary edge (Section 2.3.4).

These two targets models have complementary characteristics. The position coordinates for the Gaussian dot are the Euclidean (x, y) -axis, while those of the binary edge are its orientation and offset (b, θ) . The Gaussian dot is symmetric, while the edge exhibits distinctly anisotropic (direction dependent) behavior. The Gaussian dot adheres to all of the simplifying assumptions made for the locale theory presented in Chapters 2 and 3, while the binary edge is an exception to many of them. The binary edge provides an opportunity to illustrate that the simplifying assumptions, used in the formal presentation of the locales theory, need not confine the application of the analytical techniques.

The dot is modelled with a fixed size and a variable number of quantization levels while the edge is modelled with a variable size and fixed binary quantization. The

size of the target is expressed in unit pixels, so changing the target size is equivalent to changing the spatial sampling resolution of the digitizing scheme. Similarly, the number of intensity levels represents the intensity resolution.

The smooth, circularly symmetric form of the Gaussian dot is suggestive of a small bright light source imaged through a typical lens. It is defined by the entity function

$$E(x, y) = Ae^{-(x^2+y^2)}. \quad (5.1-1)$$

which is spatially quantized by a 3×3 sampling window and intensity quantized by integer truncation. The position of the target is defined by the centre of the dot. The position space of the Gaussian dot is the set of coordinates (x_0, y_0) defined as

$$D_{\text{dot}} = \left\{ (x_0, y_0) : |x_0| \leq \frac{1}{2}, |y_0| \leq \frac{1}{2} \right\}. \quad (5.1-2)$$

When the target is positioned at (x_0, y_0) , the pixel values of its digital representation are given by

$$p_{i,j} = \lfloor E(i - x_0, j - y_0) \rfloor \quad \text{for } -1 \leq i \leq 1, -1 \leq j \leq 1. \quad (5.1-3)$$

The number of intensity levels is adjusted from 2 to 100 by varying the amplitude A of the Gaussian. Noise, when it is included, is uniformly distributed and is added to the Gaussian function prior to quantization. Zero-level clipping is performed to ensure that noise does not result in negative pixel values.

The offset and orientation of a binary edge are described in Figure 2.2, on page 48. The digital representation of a binary edge is given by the pixel values

$$p_{i,j} = U(b - i \sin \theta + j \cos \theta), \quad (5.1-4)$$

where $U(\cdot)$ is the unit step function.

The two-dimensional position space for the binary edge is the region in (θ, b) parameter space defined by

$$D_{\text{edge}} = \left\{ (\theta, b) : 0 \leq \theta \leq \frac{\pi}{4}, |b| \leq \frac{1}{2} \right\}. \quad (5.1-5)$$

The position space for the binary edge is restricted to one octant of the (θ, b) plane to take advantage of symmetry.

In regard to the realism of the target models, it should be appreciated that neither of the target models are able to account for fine details such as the area averaging effects of a typical photo-sensor. Such effects are minor and are beyond the scope of these simulations but they may need to be taken into account in very high-precision, practical implementations of position estimation.

Both the Gaussian dot and the binary edge are easily computed target functions. The number of quantization levels in the Gaussian dot is easily modified by varying its amplitude, with little impact on computational complexity of the position estimation algorithm. The Gaussian dot can be made larger by applying a scale factor in the exponent but the number of pixels required to describe it, and the associated computational complexity, increases as the square of the target size. For an edge, on the other hand, the number of pixels needed to describe it increases only linearly with its length. To keep computational complexity to a workable level, the edge is used in studies regarding target size and the Gaussian dot is used for studies involving variations in the number of quantization levels.

Various methods for describing an edge may be used, such as *object boundary coding* (OBC) or *grid intersection quantization* (GIQ) [Dorst84s]. For the simulations, a binary edge is presented to the position estimation algorithm as an ordered set of pairs of pixel indices. Each pair consists of two vertically adjacent pixels, (i, j) and $(i, j + 1)$, with one pixel value being 1 and the other 0, as illustrated in Figure 5.1. In this way, each pixel pair defines the edge boundary in a vertical column of the digital image of the binary edge.

For the Gaussian dot, which is restricted to a 3×3 sampling window, the pixels are placed in a list in the sequence shown in Figure 5.2. The ordering of the pixels is selected so as to provide good computational efficiency in the position decoding algorithm by reducing the size of the feasible region as quickly as possible.

FIGURE 5.1 A binary edge is represented by a list of pixel pairs. Each pair consists of a dark and a light pixel, vertically adjacent to each other, which define the edge boundary in a vertical column. The pixels are ordered left to right, bottom to top, as indicated by the numbering.

6	3	8
1	9	2
5	4	7

FIGURE 5.2 A Gaussian dot is represented by a 3x3 pixel patch. The pixels are presented to the position decoding algorithm in an ordered list, with the order as shown here. This ordering scheme places opposing pixels adjacent in the list so as to reduce the size of the feasible region quickly.

FIGURE 5.3 The locale pattern in (b, θ) space for an edge of length 5 is shown at the left (also, see Figure 2.14). The position space $D = \{(b, \theta) : |b| \leq \frac{1}{2}, 0 \leq \theta \leq \frac{\pi}{4}\}$, which is used for the simulations, is shown enlarged at the right (with right-slanted striping). The alternate position space, used in [Dorst84d], is also shown at the right (with left-slanted striping).

The locale pattern of the Gaussian dot has a more random appearance than that of the binary edge (Figure 5.3). The locale map of the binary edge has many coincident boundary intersections and greater symmetry. The Gaussian dot is a more “natural” target in this respect and is used as the principal example target for the majority of the analysis and simulations.

The binary edge has been studied extensively in the literature, perhaps in more detail than any other target form. Its popularity stems from its simplicity and relevance in everyday scenes. From an analytical point of view it characterizes the interaction between the target and pixel grid, which results in anisotropic (direction dependent) effects. Furthermore, it does not satisfy the conditions of translation invariance and sampling window sufficiency, which are presumed (for simplification) by the theory of locales developed in Chapters 2 and 3. Whereas the Gaussian dot is an uncomplicated target for demonstrating the theory of locales, the binary edge illustrates how the analysis can be extended beyond the simplifying assumptions of the theory.

5.1.3 Definition of parameters

There are six main parameters for the simulation experiments:

1. number of quantization levels,
2. target size,
3. noise amplitude,
4. truth map size,
5. tolerance,
6. grid size.

The number of *quantization levels* refers to the range of pixel values within the target. Since quantization is performed by integer truncation, the range of pixel values is equal to the greatest integer less than the amplitude A of the target. The *target size*, as applied to the binary edge, is the number of pixels horizontally along its length. Noise is additive and uniformly distributed in the interval $-N$ to N , where N is the *noise amplitude*.

The truth map is a data array used by the position decoding algorithm (Section 4.3.3). The *truth map size*, determined by the number of elements along a side of the square array, affects the computational complexity of the algorithm and the resolution of the shape of the selected locale. The typical size of the truth map for the simulation experiments is 25 cells. The evaluation of each cell of the truth map involves a call to the target definition subroutine (the entity function). These subroutine calls are the major computational burden of the position decoding algorithm and the number of such calls is the measure used of the computational complexity of position decoding. One factor which affects this is the *tolerance* value, which is the number of "next-best" cells which are kept track of when searching for the lowest-cost cell of the truth map. The tolerance parameter value is typically between 0 and 5, depending on the noise level. Later versions of the position decoding software had provisions for dynamic adjustment of this parameter.

The simulation experiments model the target at positions defined by a regular grid in the target position space. The target position grid is specified as

$$(x_{kl}, y_{kl}) = (\epsilon_x + k\Delta x - \frac{1}{2}, \epsilon_y + l\Delta y - \frac{1}{2}). \quad (5.1-6)$$

The indices k and l range from 0 to $M - 1$, where M is the *grid size*. The grid intervals, $(\Delta x, \Delta y)$ for the Gaussian dot and $(\Delta b, \Delta \theta)$ for binary edge, are equal to $1/M$ of the width of the position space in the respective x and y (or θ and b) directions. The offset (ϵ_x, ϵ_y) is used to avoid redundancy in the test points and is $(\Delta x/4, \Delta y/4)$ for the Gaussian dot and $(\Delta b/3, \Delta \theta/4)$ for the binary edge.

As discussed in the next section, error statistics are computed from the estimation errors for the target positioned at all of the points in the grid. The grid width M is typically 20 points on a side so that the number of target positions tested for each resultant data point is 400. Most of the performance comparison tests, for different estimation algorithms, use a grid width of 100 points. For investigation of the anisotropic behavior of the estimates, a grid width of 50 points is used.

5.1.4 Error estimation

Position estimates $(\hat{x}_{kl}, \hat{y}_{kl})$ are obtained from the position estimation algorithm for each point (x_{kl}, y_{kl}) on the target position grid. The maximum errors and the mean square errors MSE are calculated for each axis independently. For the Gaussian dot, only the x -axis errors are reported, with

$$E_{\max} = \max_{\substack{0 \leq k \leq N-1 \\ 0 \leq l \leq N-1}} |\hat{x}_{kl} - x_{kl}|, \quad (5.1-7)$$

and

$$\text{MSE}_x = \frac{1}{N} \sum_{\substack{0 \leq k \leq N-1 \\ 0 \leq l \leq N-1}} (\hat{x}_{kl} - x_{kl})^2. \quad (5.1-8)$$

For the binary edge, the b -axis and θ -axis errors are computed in the same way.

The mean square error MSE, the *root* mean square error RMSE and the *reciprocal root* mean square error RRMSE are used to refer to position estimation errors. The

RRMSE is displayed in many of the figures to emphasize the linear relationships in the data.

Units for the position (x, y) of the Gaussian dot and the offset b of the binary edge are determined by the unit pixel spacing. The orientation θ for the binary edge is expressed in radians, unless otherwise indicated.

If an a priori probability distribution of target positions is specified then the summation in Equation 5.1-8 should be weighted accordingly. This extra complication is unnecessary here since the position space for both the Gaussian dot and the binary edge are defined so that the a priori distribution of target positions is uniform.

5.1.5 Integrity checks

The integrity of the simulation results is essential since the results motivate and support the theoretical developments. As a case in point, the recurring linearity in the simulation results (*cf.* Figure 5.21 on page 176) provokes suspicion that coding or simulation idiosyncrasies are responsible rather than fundamental relationships. Extensive efforts were taken to discharge this suspicion by way of exhaustive diagnostics and cross checking of results. These extensive verification methods were incorporated as a matter of routine to ensure continued integrity.

Diagnostic Code. Exhaustive diagnostic printout code is liberally embedded within the simulation software modules. This code is conditionally executed, to a variable degree of verbosity, based on a global parameter. The code is not removed from the final execution module of the simulation software, so that debugging versions of the software are identical to the final execution versions. The global parameter can be modified either by recompilation or by invoking interactive debugging (CodeView®), which is permanently incorporated into the executable module.

Graphics Display. High resolution monochrome graphics with multiple levels of grey are used to display the evolving contents of the truth map. This diagnostic and demonstration tool is activated from the keyboard at any time during execution of the simulation. A colour version of the display was implemented but it provided no significant advantage over the monochrome display. This graphical diagnostic, in conjunction with printed output, provides a powerful debugging aid. Visual monitoring of the progress of the algorithm helped to resolve several subtle problems during software development.

Feasible Region Check. For each pixel in the list of active pixels submitted to the position decoding algorithm, a feasibility map of target positions is computed. A check is made to verify that the most feasible regions have a non-empty intersection. If the intersection is empty, the image distance contributed by each pixel is reported as a measure of the severity of the error. In the presence of noise, this check is expected to fail occasionally. Monitoring the frequency of occurrence of such failures, and their severity, provides some assurance that the algorithm is functioning correctly.

Locale Selection Check. The selected locale(s) used for the final position estimate should include the actual target position. Diagnostic code notifies when the actual target position does not fall within a rectangle enclosing the selected locales. Since noise may cause this to occur frequently, this diagnostic is disabled when noise is present.

Redundant Error Statistics. The (signed) average estimation error is computed to see that no obvious bias is introduced into the estimate. The bias is generally an order of magnitude smaller than the RMS estimation error. The symmetry of the Gaussian dot target is used to check results by comparing the x and y directional error statistics. Since the two statistics are essentially the same, only the x -direction is reported in the simulation results.

Traceable Modifications As the software modules were enhanced and modified, an audit trail of the modifications was maintained. Occasional re-processing of some of the data obtained with old versions of the software was done with newer versions as a test for consistency. The randomness inherent in the simulation generally precludes exact replication of results with different software versions but the tests were considered successful if the results were in agreement within a few percent.

5.1.6 Computational efficiency

Speed is not an essential issue in the simulations except in regard to completing the computations in a practical time, such as several days. The nominal execution time for each data point varies from 0.1 to 15 seconds on an IBM PC/AT compatible with floating point accelerator, running at 10 MHz. As an example of the execution times required for the simulations, the data on estimation errors for binary edges required 11 hours execution time for the noise-free portion (about 4 seconds per point) and about 200 hours for the portion with additive noise.

The computation times can be reduced by software optimization but, even if the reductions were significant, the effort required to achieve them is not justified for this study, particularly as it would require elimination of the integrity checks.

Most of the data for binary edges was computed subsequent to that of the Gaussian dot. Building upon the experiences with the Gaussian dot, the approach to programming the image function for the binary edge was modified so that look-up tables are used rather than trigonometric computations. This approach probably represents the best method of implementing an image function in a practical application of position decoding.

5.1.7 Repeatability

As was mentioned in the previous section in regard to integrity checks, repetitions of many simulations were performed to ensure consistency as the software and analysis techniques evolved. An equally important level of repeatability is achieved by presenting the software and analysis used to derive the results in sufficient detail that other researchers are able to reproduce the results within the limits of sampling and noise variations. Some results, such as the example of scattered locales reported in Section 2.2.4 are unlikely to reproduce exactly due to their sensitivity to minor computational differences, but results regarding trends and relationships in the data should be reproducible.

There are no special "tricks" used in the software (nor in the equipment), so repeatability of the essential results does not depend upon the software implemented here: any implementation of the position decoding algorithm will do equally well.

The emphasis in this study is on the development and demonstration of investigative techniques. The limited scope of the simulations, in terms of target forms and parameters, is acknowledged. There is an unlimited range of possible extensions to the simulation results and researchers who have interests in other targets and parameters will have no problem using position decoding to reconstruct the simulation environment to suit their purposes.

5.2 Performance Comparisons

Optimality of the position decoding algorithm is assured in theory when there is no noise but it is not known how superior this optimal performance is to the performance of other typical algorithms. To address this question, the position decoding algorithm is compared directly with several other position estimation algorithms which are capable of achieving subpixel precision. The relative performances are compared for intensity quantization ranging from binary to essentially continuous.

The theory of locales does not quantify the performance of position decoding in the presence of noise, so simulation tests are useful to determine how the algorithm will behave in the presence of noise. For this reason, the comparison tests were done using data with additive noise as well as noise-free data.

5.2.1 Targets and parameters

The target used for the performance comparison tests is the Gaussian dot, with an amplitude ranging from 2 to 100. The Gaussian dot is selected because it is easily processed by several algorithms, because its radial symmetry simplifies the performance measure and because it is a fairly realistic representation of a small physical dot convolved with an imaging system's response function. There are obvious limitations inherent in using a single target form and small window size but the restrictions were necessary since the computational complexity of the performance comparison expands dramatically with increased generality.

No particular advantage is afforded to the position decoding algorithm over the other algorithms by the use of the Gaussian dot as a target. In fact, whereas position decoding is a general algorithm, one of the other algorithms (exponential interpolation) is defined exclusively for the Gaussian dot form.

Position estimation precision is determined for each algorithm by calculating the positioning errors at 400 target positions (10,000 when noise is present) distributed uniformly on a raster within a unit pixel area, as specified in Equation 5.1-6. The grid is offset by $(\epsilon_x, \epsilon_y) = (0.0025, 0.0025)$. Only the x -direction estimation errors are reported since the y -direction results are essentially identical. The truth map size for position decoding is 12. The tolerance parameter is zero when there is no noise and 4 when there is noise.

When noise is added to the images, it is uniformly distributed on the interval $[-1, 1]$ and added to the entity function prior to quantization.

5.2.2 Description of the algorithms

Position decoding is compared with three other algorithms which were selected on the basis of their simplicity and the degree to which they make use of knowledge of the Gaussian form of the target.

The first is a *centre of mass* (COM) estimator, which determines the classical first moment of the intensity distribution. It does not depend on a model of the target and does not make use of any a priori information except that the target position is specified by its centre. The second is a *Fourier phase estimation* technique (FP), which requires a discrete model for the target. It is severely limited in its use of target information due to the discrete nature of its model. The third is an *exponential interpolation* algorithm (EI) which makes specific use of the exponential form of the Gaussian target. The EI estimate is based on an explicit solution to Equation 5.1-3 for (x_0, y_0) in terms of the pixel values p_{ij} . Finally, position decoding, which is the most complex of the algorithms, makes full use of a priori knowledge of the target shape by using a target model.

Centre of Mass Estimation (COM). For an entity function $E(x, y)$ representing an arbitrary target, the centre of mass $(\hat{x}, \hat{y})$ is given by the simple equations

$$\hat{x} = \frac{\sum i p_{ij}}{\sum p_{ij}} \quad \text{and} \quad \hat{y} = \frac{\sum j p_{ij}}{\sum p_{ij}}, \quad (5.2-1)$$

where the summations run over the indices i and j in the sampling window.

In the absence of noise, and with continuous intensity values, the centre of mass estimate can be explicitly determined analytically. With pixel values $\{p_{ij}\}$ given by Equation 5.1-3 and sampling done on a 3×3 window centred on the origin, the x -axis estimate as a function of the actual position (x_0, y_0) is given by

$$\hat{x} = \frac{\sinh(2x_0)}{(\cosh(2x_0) + \frac{1}{2}\epsilon)}. \quad (5.2-2)$$

The derivation of this result can be found in Appendix A.1. The y -axis estimate has a corresponding expression.

The COM estimate is skewed, as discussed in Section 3.1 and Figure 3.1. For continuous intensity values, the estimation error is given by

$$\delta x = (x_0 - \hat{x}) = x_0 - \frac{\sinh(2x_0)}{(\cosh(2x_0) + \frac{1}{2}e)}. \quad (5.2-3)$$

Symmetry of the estimation error ensures that the average error is zero but the RMS error in x is approximately 0.05. In principle, the estimation error can be reduced or eliminated by adding the value δx to the estimate but such a modification to the COM estimate is not pursued in this study.

The software listing of this algorithm is given in Appendix C.5.

Fourier Phase Estimation (FP). Since a linear phase shift in a Fourier transform corresponds to a translation in position [Oppen75], a target position estimate can be obtained by performing a linear regression on the phase shift in the Fourier transform of the observed image with respect to that of a reference image. If a target is represented by an entity function $E(x, y)$ then its Fourier transform is given by

$$\mathcal{F}(\xi, \eta) = \iint E(x, y) e^{-2\pi j(x\xi + y\eta)} dx dy. \quad (5.2-4)$$

The target, translated by $\vec{\Delta x} = (\Delta x, \Delta y)$, is represented by $E_{\vec{\Delta x}}(x, y) = E(x + \Delta x, y + \Delta y)$ and has the Fourier transform

$$\mathcal{F}_{\vec{\Delta x}}(\xi, \eta) = \mathcal{F}(\xi, \eta) e^{-2\pi j(\Delta x \xi + \Delta y \eta)}. \quad (5.2-5)$$

The quotient of the Fourier transforms is

$$Q(\xi, \eta) = \frac{\mathcal{F}(\xi, \eta)}{\mathcal{F}_{\vec{\Delta x}}(\xi, \eta)} \quad (5.2-6)$$

and it has unit amplitude with a phase given by

$$\arg(Q(\xi, \eta)) = -2\pi j(\Delta x \xi + \Delta y \eta). \quad (5.2-7)$$

In the non-continuous case, some assumptions on signal periodicity and band limiting must be made to establish a corresponding phase linearity. These theoretical

details are not elaborated on in this heuristic motivation for the Fourier phase estimation algorithm (see [Oppen75]). The discrete Fourier transform F of a digital image of the target is given by

$$F(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E(n - x_0, m - y_0) e^{-2\pi j(nk + ml)/N}, \quad (5.2-8)$$

where N is the width of the sampling window and $\vec{x}_0 = (x_0, y_0)$ is the position of the target. The corresponding quotient function for the digital image of a shifted target is $Q(k, l)$, and

$$\arg(Q(k, l)) = \frac{k\Delta x + l\Delta y}{2\pi N}. \quad (5.2-9)$$

An observed image, with discrete Fourier transform $\hat{F}(k, l)$, is treated as a translation of the model image to obtain the quotient function

$$\hat{Q}(k, l) = \frac{F(k, l)}{\hat{F}(k, l)}. \quad (5.2-10)$$

The quotient function is then modelled as having linear phase;

$$\arg(\hat{Q}(k, l)) = \alpha k + \beta l + \gamma. \quad (5.2-11)$$

The least squares estimate for the parameters α , β and γ is obtained by solving the normal matrix equation

$$\begin{bmatrix} \sum kQ(k, l) \\ \sum lQ(k, l) \\ \sum Q(k, l) \end{bmatrix} - \begin{bmatrix} \sum k^2 & \sum kl & \sum k \\ \sum kl & \sum l^2 & \sum l \\ \sum k & \sum l & \sum 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \mathbf{0}. \quad (5.2-12)$$

The summations are over the two indices k and l . Although these indices are conventionally taken to run between 0 and $(N - 1)$, if N is odd they can just as well run between $-(N - 1)/2$ and $+(N - 1)/2$, in which case the summations simplify to

$$\begin{bmatrix} \sum kQ(k, l) \\ \sum lQ(k, l) \\ \sum Q(k, l) \end{bmatrix} - \begin{bmatrix} \frac{(N^3 - N)}{12} & 0 & 0 \\ 0 & \frac{(N^3 - N)}{12} & 0 \\ 0 & 0 & N^2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \mathbf{0}. \quad (5.2-13)$$

This simple matrix equation is used to define the Fourier phase estimate of the position with

$$\hat{x} = \frac{\alpha}{2\pi/N} \quad \text{and} \quad \hat{y} = \frac{\beta}{2\pi/N}. \quad (5.2-14)$$

For the simulation studies, $N=5$. Software listings for the Fourier phase estimation algorithm can be found in Appendix C.6.

Exponential Interpolation (EI). An explicit solution to the simultaneous equations

$$p_{ij} = Ae^{-(i-x_0)^2 + (j-y_0)^2} \quad \text{for } -1 \leq i \leq 1, -1 \leq j \leq 1 \quad (5.2-15)$$

is given by

$$\hat{x} = \frac{1}{12} \log \frac{K_3(+1)}{K_3(-1)} \quad \text{with} \quad K_3(i) = \prod_{-1 \leq j \leq 1} p_{ij}. \quad (5.2-16)$$

A comparable definition is made for $\hat{y}$. The coefficient $1/12$ is determined by noting that without noise

$$K_3(i) = A^3 e^{-3(x_0^2 + y_0^2 + i^2) + 6ix_0 - 2}, \quad (5.2-17)$$

and hence

$$\frac{K_3(+1)}{K_3(-1)} = e^{12x_0}. \quad (5.2-18)$$

The subscript on K_3 identifies the window size. For completeness, the expression for larger (odd sized) windows is given as

$$\hat{x} = \frac{2}{N^3 - N} \sum_{i=1}^{\frac{N-1}{2}} \log \left(\frac{K_N(+i)}{K_N(-i)} \right), \quad \text{where} \quad K_N(i) = \prod_{|j| \leq \frac{N-1}{2}} p_{ij}. \quad (5.2-19)$$

The performance of this algorithm is much better when intensity quantization is done by integer rounding than when integer truncation is used. so $\frac{1}{2}$ is added to each pixel value before it is used to compute $\hat{x}$. thus;

$$K_3(i) = \prod_{-1 \leq j \leq 1} (p_{ij} + \frac{1}{2}). \quad (5.2-20)$$

Software listings for this algorithm can be found in Appendix C.7

5.2.3 Results of the comparison test

The position estimation precision of the four algorithms without noise are shown in Figures 5.6 and 5.7, starting on page 161. The lineal bound is seen to lie above all of the curves. The reciprocal root mean square error RRMSE for position decoding is consistently greater than that of the other algorithms. The performance curve for position decoding is linear, in dramatic contrast to that of the COM algorithm which is asymptotically bounded at about 20. (This indicates that the COM estimator cannot provide better than 0.05 pixel position estimation precision). The FP algorithm performs better than the COM algorithm. This is expected since the former incorporates more a priori information about the target shape. The EI algorithm makes use of more a priori information than either the COM or the FP algorithm and has correspondingly superior performance.

It is interesting that, with about a dozen quantization levels, the COM performance curve rises above its asymptotic limit. This is an example of intensity quantized images providing better position estimation precision than non-quantized images.

The addition of noise does not change the essential character of the performance curves, as can be seen in Figures 5.8 and 5.9 (pages 163 and 164) where the RRMSE for each of the four algorithms is shown as a function of the number of quantization levels A . The lineal bound and the RRMSE for position decoding without noise are included in Figure 5.8 for comparison.

In Figure 5.9 the curve for the COM estimate crosses that of the position decoding algorithm, which brings into question whether or not the latter is optimal in the presence of noise. In fact, it cannot be conclusively discerned whether or not the curves actually cross, due to the uncertainties inherent in the simulation data. This brings to the fore the important *caveat emptor* implicit in simulation studies: the data are subjugated by the conditions of the tests and should not be unduly generalized or applied literally. In this case, the truth map size for the position decoding algorithm was only 12 celis wide, in order to economize on computation. The coarse resolution

in the truth map generally increases the RMS error due to the poor estimate of the locale centroid. More detailed computations show that the COM and position decoding estimation errors are essentially indistinguishable for a Gaussian amplitude of about 9.

The theory determines that position decoding is optimal in the absence of noise but not necessarily in the presence of noise. Nevertheless, the simulations demonstrate that it has excellent (and linear) performance in the presence of noise. As for the COM estimate, it is very much to its credit that such a simple algorithm performs so well with 5 to 10 quantization levels, although it deteriorates dramatically at higher intensity resolution.

The slope of the lineal bound is about 12, which is considerably greater than the slope of the observed performance curve for position decoding, which is about 5 without noise. The divergence of these two curves is not surprising, since the lineal bound does not take into account the distribution of locale sizes and shapes.

5.3 Modelling Errors

The position decoding algorithm is dependent upon a good model for the target. In order to examine the stability of the algorithm under less than perfect target modelling, random amplitude errors were applied to the Gaussian dot target. Unlike the additive noise used in the other simulation tests, the same amplitude error is applied to all pixel values of the digital representation. The amplitude error is defined as a percentage of the correct (pre-quantization) pixel value. The target model used by the position decoding algorithm is not affected; the modelling error is induced by applying noise to the digital image.

This simulation test is not intended to emulate real-world modelling errors. The objective is to see if the performance of the algorithm degrades gracefully under moderate modelling errors. A steady and gradual decline in performance is desired rather than failure of the algorithm to converge on a position estimate.

The study of modelling errors uses target amplitude errors because they are easily quantifiable and are representative of the type of errors expected due to imprecise control of the lighting in a practical application.

The pixel values p_{ij} for a target position (x_0, y_0) are given by

$$p_{ij} = \lfloor A(1 + n)e^{-((x-i)^2 + (y-j)^2)} \rfloor. \quad (5.3-1)$$

The random error n is the same for all pixels, but changes with target position. The error is uniformly distributed in the interval $[-\alpha, \alpha]$, where α ranges from 0 to 50%. The target amplitude is fixed at $A = 10$ for these tests.

The results of the simulation tests are shown in Figure 5.10 (page 165) where the degradation in performance is seen to be gradual over the range of noise levels used. The detailed relationship between the severity of the modelling error and the degradation in estimation precision is not at issue here; the important thing is that a gradual decline is observed.

5.4 Effects of Target Size

The binary edge target is used to examine how precision varies with target size. The size of the binary edge is defined as the number of pixels K along its length.

The first objective is to determine if the performance curve for optimal position estimation using the position decoding algorithm has the same general shape as that of the lineal bound.

The second objective is to investigate how noise affects the performance curves and attempt to parametrize the curves in order to predict performance in the presence of noise.

The third objective is to investigate the anisotropic characteristics of the position estimation errors and make comparisons between the simulation data and analytical results reported in the literature for binary edges.

5.4.1 Target and parameters

The binary edge was chosen for these experiments because the computational complexity of position decoding increases only linearly with the length of the edge. Position estimation for edges has been extensively studied and there are published results to which the simulation data can be compared [Dorst84d]. Investigating binary edges provides an opportunity to demonstrate the application of the locales theory to non-standard locales since the edge target does not have a translation invariant image function (Section 2.1.4) or a sufficient sampling window (Section 2.2.1). Unlike the Gaussian dot, the two coordinates of its position space have different characteristics. Each coordinate provides its own insights into the locales approach to geometric precision. (See Sections 2.3.4 and 3.4.3.)

In considering the results of the simulation, particularly the maximum errors, it should be kept in mind that the definition of the target positioning grid omits the edge oriented with $\theta = 0$ since the grid is shifted by $(\epsilon_x, \epsilon_y) = (\frac{1}{3}\Delta b, \frac{1}{4}\Delta\theta)$ (Equation 5.1-6). Data showing the variation of precision with edge length is obtained using a 20×20 point grid. Data showing the dependence of precision on edge orientation is obtained using a 50×50 point grid. For all computations the size of the truth map is 25 cells square.

5.4.2 Results and discussion

The analysis developed in Section 3.4.3 for binary edges is useful for the discussion of these simulation tests. The locale pattern for an edge of length $K=5$ is shown in Figure 5.3 (page 134). The lineal bound on precision, in terms of the reciprocal root mean square error RRMSE, is given by $\sqrt{12V}$, where V is the total variation of the image function. Based on Equations 3.3-12, 3.4-9, and 3.4-12 and Figure 5.3, the lineal bound on RRMSE for the offset B_b and orientation B_θ of a binary edge of length K are given by

$$B_b \approx \sqrt{12}(1.12K) \approx 3.880K \quad (5.4-1)$$

and

$$B_\theta = \sqrt{12}(K-1)^2/\pi \approx (1.05(K-1))^2. \quad (5.4-2)$$

Note that the lineal bound on precision for the edge offset is linear while the lineal bound for orientation is quadratic.

The RRMSE In Offset b . The simulation results for the RRMSE in b are shown in Figure 5.11 (page 166). The performance curve is essentially linear but has a substantially different slope and intercept than that of the lineal bound. The slope of the data is 0.78, about one fifth that of the lineal bound.

According to the locales theory, the gap between the lineal bound and the observed RRMSE is a result of the distribution of locale sizes and shapes. The locale map for the binary edge has many coincident intersections due to colinearity of pixels in the imaging array, making it reasonable to expect a greater divergence between the RRMSE and the lineal bound than was observed for the Gaussian dot target (Figure 5.6 on page 161).

For the range of edge lengths considered in the simulation, the RRMSE is approximated by the linear relationship

$$\text{RRMSE}_b \approx 7.1 + 0.78K. \quad (5.4-3)$$

The non-zero intercept is apparently due to the fact that the locales along the boundary of the position space (at $b = \pm \frac{1}{2}$) are truncated. If the perimeter of the position space adds 2 "virtual" locale boundaries to the total variation then the adjusted lineal bound on the RRMSE (Equation 3.3-12) will increase by $2\sqrt{12}=7$, which is close to the observed intercept of 7.1.

The RRMSE in Orientation θ . The lineal bound on RRMSE for orientation estimation is a quadratic function of the edge length. When the estimation errors

obtained from the simulation are expressed as the *square root* of the RRMSE, as in Figure 5.12 (page 167), it is seen that the performance curve is linear. The observed estimation errors are approximated by the quadratic relationship

$$\begin{aligned} \text{RRMSE}_\theta &\approx 2.465 + 1.526K + 0.236K^2, \\ &\approx (0.49K + 1.56)^2. \end{aligned} \quad (5.4-4)$$

The ratio of the slope of the lineal bound to that of the observed RRMSE_θ is about $(1.05/0.49)^2 \approx 4.6$. For short edges ($K = 2, 3, 4$) the observed estimation errors are less than the lower bound provided by the lineal bound. This apparent anomaly arises from the definition of the position space used for the simulation, which truncates the large locales on its boundaries (at $\theta=0, \frac{\pi}{4}$). The observed data will lie below the lineal bound if it is adjusted upward by $2\sqrt{12}$ to account for the position space boundaries, as discussed above for the RRMSE in edge offset estimation.

The Maximum Errors. The results for the maximum estimation errors for the offset and orientation of the binary edge are shown in Figure 5.13 (page 168). The reciprocal maximum error is plotted as a function of the edge length. For edge offset the curve is essentially constant, fluctuating between 3 and 5. For edge orientation the curve increases faster than linearly with the length of the edge.

The values of the maximum errors are sensitive to details such the target position grid (Equation 5.1-6) and the size of the truth map. Nevertheless, the general character of these errors can be predicted from the largest locales in the locale maps.

Considering the locale map for a binary edge (Figure 5.3 on page 134), there are vertical columns of locales at $\theta=0$ and $\theta=\frac{\pi}{2}$. These locales are shaped like "eyes" and appear to be the largest locales. They do not diminish in height as the length of the edge increases and their width is $2 \arctan(\frac{1}{K})$. For the simulation studies, the position space is defined with $-\frac{1}{2} \leq b \leq \frac{1}{2}$ and $0 \leq \theta \leq \frac{\pi}{4}$, which truncates the two eyes at $\theta=0$, leaving them with a width of $\arctan(\frac{1}{K})$. The truncated eyes have a triangular shape and their centroids are approximately $(\frac{1}{3} \arctan(\frac{1}{K}), \pm \frac{1}{3})$. Within the locales

defined by these truncated eyes, the largest error in b is

$$\hat{\epsilon}_b = \frac{1}{3}.$$

which is in approximate agreement with the observed maximum error values.

The largest error in θ within the truncated locales is $\hat{\epsilon}_\theta = \frac{2}{3} \arctan(\frac{1}{K}) \approx \frac{2}{3K}$, so the predicted reciprocal maximum error in θ is

$$\hat{\epsilon}_\theta \approx \frac{3K}{2}.$$

This roughly approximates the experimental results shown in Figure 5.13.

Anisotropy of Errors. The precision with which a straight edge can be positioned depends greatly upon its actual orientation. An edge aligned either vertically or horizontally with the sampling grid will exhibit a position uncertainty equal to the spacing of the grid, but at other orientations the uncertainty is dramatically less. This anisotropic behavior has been studied in detail in the literature [Dorst84d], where the average error in edge offset estimation has been determined analytically as a function of the edge orientation. In this section the results from the literature are compared to the simulation results obtained for the position decoding algorithm.

The RMSE and the maximum errors in edge orientation and offset are plotted in Figures 5.14 and 5.15 (pages 169 and 170) as functions of the edge orientation. For each edge orientation, the estimation errors in θ and b are averaged with respect to the edge offset.

The maximum estimation errors are discontinuous functions of the edge orientation, although the small number of sample points does not allow this to be clearly visible in the figures. The discontinuities arise from the fact that the position estimates, given by the locale centroids, change discontinuously as a function of edge orientation. These discontinuities are not observed when edge offset is estimated with known edge orientation, as in [Dorst84d], rather than simultaneous estimation of θ and b .

Between points of discontinuity, the maximum estimation errors in θ are linear and always have the same magnitude slope. The maximum estimation errors in b are curved between points of discontinuity.

The RMS error curves for both θ and b are smooth functions of edge orientation except for a few cusps. The cusps for θ and b occur at the same orientations but those for θ are minima while those for b are maxima. The cusps occur at “eyes” in the locale pattern, where the orientation errors reach a minimum because of approximate symmetry about the vertical axis of the eyes and the offset errors reach a maximum because of the large vertical extent of the eyes.

A special simulation run of the position decoding algorithm was performed to obtain averaged magnitude error, computed on the basis of *known* edge orientation, in order to make a comparison with the analytically determined errors reported in [Dorst84d]. The simultaneous orientation and offset estimation capabilities of the algorithm were bypassed by using the exact orientation to obtain the offset estimate. The results[†] are shown in Figure 5.16 (page 171), where it can be seen that agreement between the simulation data and the analytical data is excellent. This confirms the optimality of the position decoding algorithm, at least for one parameter estimation.

5.5 Additive Noise

The formal theory of locales, as far as it has been developed, says little about position estimation in the presence of noise. It does, however, provide the position decoding algorithm which can be applied to noisy data. In this section, the algorithm is demonstrated with noisy data and used to investigate how position estimation precision varies with the noise level. Insofar as the position decoding algorithm provides the optimal position estimate, the performance curves obtained[†] from the simulations represent bounds on the attainable position estimation precision in the presence of noise. A distinctive property of the performance curves, particularly for the Gaussian

[†] Analytical data provided by L. Dorst, currently at Philips Labs, Briarcliff Manor, NY, USA.

dot target, is their linearity. The choice of axes for the graphs is made, in most cases, so that the linearity is clearly visible.

5.5.1 Targets and parameters

Both the Gaussian dot and the binary edge are investigated in this section. Noise is added to the entity functions of the target models prior to quantization. The noise n is uniformly distributed between $\pm N$, where N is the noise amplitude. Quantization is performed so as to ensure that negative pixel values do not occur. Uniformly distributed noise is used because it is easily computed and because its distribution does not have long tails such as those of a Gaussian probability distribution.

The noisy image of a Gaussian dot is obtained by adding the noise random variable n to the entity function,

$$I(x, y) = [E(x, y) + n]. \quad (5.5-1)$$

The image function is obtained by binary quantization of the edge amplitude profile $h(b)$ at a threshold t , as shown in Figure 5.4. The form of the profile $h(b)$ is irrelevant and need not be specified, unless there is noise present.

The addition of a noise value to the edge amplitude profile results in a vertical shift, as illustrated in Figure 5.5. The shifted edge profile $h(b) + n$ will cross the quantization threshold at a position which depends on the value of n and so the noise can be said to cause a shift in the edge position at a given pixel. For computational purposes, the noise is applied to each pixel of a binary edge as a uniformly distributed shift in the edge position. This is equivalent to the addition of uniformly distributed noise if the edge amplitude profile is linear in the vicinity of the threshold t .

For the simulation tests, the length K of the binary edge ranges from 2 to 25 pixels. Additive noise levels applied to the edge range from 0 to 0.5 pixel. The ratio of the Gaussian dot amplitude A to the noise amplitude N ranges between 1 and 30.

Target positions are selected from a regular grid in target position space, as in Equation 5.1-6. The width of the grid is 20 points for the Gaussian dot and the

FIGURE 5.4 A realistic edge would have a profile as shown here; smooth, approximately linear in its central region and extending across two or three pixels. Binary thresholding is done near the centre of the linear region of the profile. The position of the edge is (arbitrarily) defined by the point on the profile at which the threshold is reached.

FIGURE 5.5 The intensity profile of an edge is shown raised and lowered by additive noise. This shift in the profile moves the apparent position of the edge, which is defined by the point where the profile crosses the threshold t . For a given edge profile and noise distribution, there is a corresponding distribution of shifts in the edge position. For computational simplicity a noisy digital image of a binary edge is simulated by applying a random shift to the edge profile before computing each binary pixel value.

binary edges. For the studies of the anisotropy of the binary edge, a grid width of 50 points is used. The locale map width is 25 cells and the tolerance parameter value is set high enough that the truth map evaluation is essentially exhaustive.

5.5.2 Performance curves

Performance curves for the position decoding algorithm applied to the Gaussian dot are presented in Section 5.2.3, with and without noise (see Figures 5.6 to 5.9 starting on page 161). Those curves are linear functions of the target amplitude. Further data is presented here, in Figure 5.17 (page 172), showing the RMSE as a function of additive noise level for a variety of target amplitudes. These curves are also linear, indicating that the performance is a linear function of the noise level as well as the target amplitude.

Linearity of the performance curves for position decoding is in contrast to those of other algorithms, such as the centre of mass estimate (COM), as can be seen in Figure 5.18 (page 173). It is interesting that the COM curves are distinctly non-linear, unlike those of position decoding. This means that for a noise level of 1, say, decreasing the amplitude A of the target from 30 to 20 can actually result in improved estimation precision. The dips in the performance curves for the COM estimator indicate that raising the noise level can improve precision.

To address the question of whether or not the linear performance of position decoding is restricted to the Gaussian dot target, similar data was collected for a binary edge. Figures 5.19 and 5.20 (pages 174 and 175) show the RRMSE for estimation of the orientation and offset of a binary edge as a function of noise level. Various lengths of edges are considered.

In every case, the performance curves appear to be asymptotically linear but are convex upward at lower noise levels. This curvature indicates some immunity to noise, which may be associated with the tendency of the noise to randomize the locale pattern. (This is an heuristic, since the formal definition of locales does not

extend to noisy images.) By randomizing the locale pattern, the large estimation errors associated with large locales are reduced, offsetting the increase in estimation errors associated with the noise.

5.5.3 Linearity of performance

In this section, the predictability of the observed position estimation errors in the presence of noise is considered. The analyses here are tentative but encouraging; they demonstrate simple relationships between the observed data but the limited scope of this research precludes the development of further formalism or generalization.

It has been seen that position estimation precision for the Gaussian dot is a linear function of the target amplitude A (Figures 5.6 and 5.8) and an inverse function of the noise amplitude N (Figure 5.17). If the *noise-equivalent* amplitude is defined as $A' = A/(2N + 1)$ then it is found that precision is also a linear function of A' . This is shown by the scatter plot of the data collected for position estimation errors for the noisy Gaussian dot in Figure 5.21 (page 176).

For the binary edge, the corresponding noise-equivalent edge lengths are defined separately for edge orientation K'_θ and edge offset K'_b . The linearity of performance as a function of these parameters is shown in Figures 5.22 and 5.23 (pages 177 and 178).

For edge orientation, the noise-equivalent length K'_θ is derived from the performance curves in Figures 5.12 and 5.19, as discussed in Appendix A.6.1, to obtain

$$K'_\theta = \frac{(K - 1)^2 - 13}{0.56K(N + 3.2) - 4.0} \quad (5.5-2)$$

For edge offset, the noise-equivalent length is derived from the slopes of the performance curves in Figures 5.11 and 5.20, as discussed in Appendix A.6.2, to obtain

$$K'_b = \frac{K + 9.1}{0.22NK + 2.0N + 1.3} \quad (5.5-3)$$

It may be feasible to apply these relationships to define noise, or *geometric fidelity*.

in an imaging system in terms of the ratio of observed and theoretical performances. The observed position estimation precision for the position decoding algorithm applied to a dot shaped target imaged by the system, for example, could be compared to the theoretical precision predicted by the lineal bound. The result would be a geometric image quality measure which has its basis in objective theory rather than subjective assessment. Rigorous examination of this issue is beyond the scope of this thesis but it deserves mention because such techniques are needed in the field of digital image metrology, which is lacking a unifying framework of standards and calibration techniques.

5.5.4 Anisotropic performance

The anisotropy of the RMSE for edge orientation and offset estimation, for varying levels of noise, is shown in Figures 5.24 and 5.25 respectively (pages 179 and 180). The data was obtained with an edge of length 5. The characteristic anisotropic structure in the error curves are obscured at a noise amplitude of about 0.3 for orientation estimation and about 0.2 for offset estimation. Referring to Figures 5.19 and 5.20 (pages 174 and 175), these noise levels correspond roughly to where the errors begin to vary linearly with the noise level. Heuristically, this suggests that noise first randomizes the directional behavior of the estimates (by randomizing the locale pattern) then, at higher levels, it diminishes the average precision in a linear manner.

In the case of edge orientation estimation errors (Figure 5.24), the detailed structure of the noise-free curve is randomized as noise is added but a gradual variation in precision with edge orientation still persists, with a decreasing trend toward $\theta = 0^\circ$ and $\theta = 45^\circ$. This trend appears to be related to the strong symmetry in the locale pattern at these orientations.

5.6 Summary

In this chapter the position decoding algorithm has been demonstrated and used to establish the optimal position estimation precision for the Gaussian dot and the binary edge targets. Optimality of the algorithm has been tested in the noise-free situation by comparing its performance with analytical results obtained by other researchers for binary edges. The stability (convergence) of the algorithm in the presence of additive noise and target modelling errors has been demonstrated.

The position decoding algorithm has been shown to have performance superior to several other algorithms, both with and without additive noise in the data. Position decoding has significantly superior performance when the data has high intensity-quantization resolution.

The optimal position estimation precision has been determined as a function of the number of quantization levels and the target size for the dot and edge targets, respectively, using the position decoding algorithm. The lineal bound has been compared to the performance of position decoding in the absence of noise, as summarized below.

Target	Lineal Bound	Position Decoding	Gap (Ratio of Slopes)
Gaussian Dot	$12.2(A - 1)$	$5A$	2.5
Edge Orientation	$(1.05(K - 1))^2$	$(0.49(K + 3.25))^2$	4.6
Edge Offset	$3.88K$	$7.1 + 0.78K$	5.0

The significantly different gap observed for edge offset may be due to the characteristic "eyes" in the locale patterns of the binary edges.

Insofar as the algorithm is near-optimal in the presence of noise, its performance curves provide estimates of the optimal position estimation precision for noisy data.

The RRMSE in the presence of noise has been shown to bear a linear relationship to the noise-equivalent amplitude A' and noise-equivalent lengths K'_b , K'_θ for the Gaussian dot and the binary edge, respectively, as defined in the table below.

Target	Noise-Equivalent Target Parameter
Gaussian Dot	$A' = \frac{A}{2N + 1}$
Edge Orientation	$K'_\theta = \frac{(K - 1)^2 - 13}{0.56K(N + 3.2) - 4.0}$
Edge Offset	$K'_b = \frac{K + 9.1}{0.22NK + 2.0N + 1.3}$

The anisotropic character of the estimation errors for the binary edge was investigated. It was shown that the binary edge exhibited some error-immunity at low noise levels and that the position estimation errors become essentially isotropic with the addition of noise.

FIGURE 5.6 Position estimation errors from several algorithms are compared for the Gaussian dot target. The reciprocal root mean square error in the x -direction ($RRMSE_x$) is shown as a function of the amplitude (A) of the Gaussian dot. The lineal bound provides an upper bound to the attainable precision. The optimal position estimate, as determined by the performance of position decoding, is beneath the lineal bound and above the other estimates. The other estimates are exponential interpolation (EI), Fourier phase estimation (FP), and centre of mass calculation (COM). The slope of the lineal bound is about 12 while the slope of the position decoding curve is about 5.

FIGURE 5.7 The performance curves of Figure 5.6 are shown here in greater detail for lower target amplitudes.

FIGURE 5.8 As in figure 5.6, the position estimation errors of four algorithms are compared for the Gaussian dot target. In this case, noise is added to the target images. The noise is uniformly distributed between ± 1 and is added to the signal before quantization.

FIGURE 5.9 This figure shows the performance curves (with noise) of Figure 5.8 in greater detail at lower target amplitudes

FIGURE 5.10 Target modeling errors, for the Gaussian dot in the position decoding algorithm, are simulated by rescaling the image intensity prior to quantization. Increasing errors of up to 50% result in a steady decline in performance but the algorithm continues to converge on a position estimate without failure

FIGURE 5.11 The reciprocal RMSE in offset estimation is shown as a function of binary edge length. The curve is essentially linear but has a different slope and offset than the lineal bound

FIGURE 5.12 The reciprocal fourth root of the mean square error in orientation estimation is shown as a function of binary edge length. The simulation data curve is essentially linear but has a different slope and offset than the lineal bound.

FIGURE 5.13 The reciprocal maximum estimation errors in offset and orientation are shown as a function of binary edge length

FIGURE 5.14 The maximum error and RMSE in orientation estimation are plotted as a function of edge orientation for a binary edge of length 5.

FIGURE 5.15 The maximum error and RMSE in offset estimation is plotted as a function of edge orientation for a binary edge of length 5

FIGURE 5.16 The analytically determined optimal estimation errors are compared to the position decoding estimation errors. The magnitude of the average errors in edge position estimation are plotted as a function of the edge orientation. Edge position is measured as the vertical distance from the origin. Edge orientation is measured as $\tan(\theta)$.

Analytical data courtesy L. Dorst

FIGURE 5.17 The RMSE in position estimation (in the x -direction) as a function of the noise amplitude, for position decoding applied to the Gaussian dot target, is shown for various target amplitudes (A). In each case, the curve is essentially linear.

FIGURE 5.18 The RMSE in position estimation (x -direction) for the centre of mass (COM) estimate, applied to the Gaussian dot target, as a function of noise amplitude for various target amplitudes (A). Unlike the position decoding estimate (Figure 5.17), the COM shows no linear trends.

FIGURE 5.19 The RMSE for edge orientation estimation, using position decoding, is shown as a function of noise amplitude for edge lengths $K = 5, 7, 10$ and 15 . Notice the asymptotic linearity. At low noise levels, particularly for short edge lengths, the noise does not degrade performance significantly.

FIGURE 5.20 The RMSE for edge offset estimation, using position decoding, is shown as a function of noise amplitude for edge lengths $K = 5, 7, 10$ and 15 . Just as for the orientation estimation errors (Figure 5.19) performance is asymptotically linear and low noise level do not degrade performance significantly for short edges.

FIGURE 5.21 The RRMSE in position estimation for the Gaussian dot, using position decoding, is linear with target amplitude (Figures 5.6 and 5.8) and inversely linear with noise amplitude (figure 5.17). The RRMSE is also essentially linear with the ratio $A' = A/(2N + 1)$, called the noise equivalent target amplitude. The slope of the data is approximately 3.8.

3

OF/DE

3

Microl

FIGURE 5.22 The RRMS estimation errors in edge orientation are plotted as a function of the noise-equivalent edge length K'_θ . The noise-equivalent edge length is derived in Appendix A.6.1

FIGURE 5.23 The RRMS estimation errors in edge offset are plotted as a function of the noise-equivalent edge length K'_b . The noise-equivalent edge length is derived in Appendix A.6.2

FIGURE 5.24 As noise is added to a binary edge of length 5, the characteristic anisotropic distribution of orientation estimation errors is randomized. At a noise level of about 0.30 (dashed curve) the anisotropic pattern is essentially obscured, although a gradual variation with edge orientation persists. The drooping of the error curve at $\theta=0$ and $\theta=45^\circ$, which becomes even more distinctive at a noise level of 0.50, may be due to the strong symmetry in the locale pattern at these orientations.

FIGURE 5.25 As noise is added to a binary edge of length 5, the characteristic anisotropic distribution of offset estimation errors are randomized. At a noise level of about 0.20 (dashed curve) the anisotropic pattern is essentially obscured and the errors are approximately independent of edge orientation

Chapter 6

SUMMARY AND CONCLUSIONS

6.1 Summary

This thesis has addressed the problem of high precision position estimation in digital image metrology. The objectives of the research, which are fundamental to the discipline of digital image metrology, are as follows.

1. Determine the optimal position estimation precision for arbitrary target forms and arbitrary digitizing schemes.
2. Find a simple, practical method for computing a bound on the optimal precision obtainable from a digital image with arbitrary target form and digitizing scheme.
3. Design an algorithm which achieves the optimal precision.
4. Implement a variation of the theoretically optimal algorithm which is computationally practical and suitable for use with noisy data.
5. Establish a unifying theoretical framework for the analysis of geometric precision in digital images based on the concept of locales.

Each of these objectives is met by this thesis research. Objective (1) is satisfied by Equation 3.2-4, which expresses the optimal precision in terms of the locale shape, size and density parameters. The lineal bound on precision (Equation 3.3-12), which is based on the lineal density of locales, achieves objective (2). The position decoding algorithm, described in Chapter 4, and its implementation for the simulation studies, reported in Chapter 5, address objectives (3) and (4). The application of the locales theory to targets as diverse as the Gaussian dot and binary edges demonstrates the unifying framework provided by the theory, as desired by objective (5).

The formalism for locales, which is developed in Chapter 2, is both general and realistic. It presumes a target model which incorporates the characteristics of the imaging systems and specifically relies on the realism of the target model to resolve issues such as locale connectedness and convexity.

The main concepts of the formalism are: the target *position space* (Section 2.1.1), the *image function* (Section 2.1.5), the *sampling window* (Section 2.2.1), the *image space* (Section 2.2.2), the *locales* (Section 2.2.3) and the *locale map* (Section 2.3).

Chapter 3 addresses the question of optimal position estimation. A simple expression for the *optimal mean square error* MSE_0 in terms of the shape, size and density parameters of the locale map, S , D , and N_a respectively is provided by Equation 3.2-4

$$MSE_0 = \frac{S + D}{N_a}.$$

It is remarkable that the optimal position estimation precision, which depends upon the complex and variable pattern of locales, can be expressed in such a simple form. Each of the three parameters describes a natural and distinctly different characteristic of the locale pattern.

This expression for MSE_0 is elegantly simple but it is computationally difficult to evaluate. The *lineal bound* (Section 3.3.4), on the other hand, is very simple to compute, as illustrated by several examples in Section 3.4. The tightness of the lineal bound to the optimal precision is examined in the simulation studies of Chapter 5. It is found that the lineal bound maintains a constant ratio with the optimal precision

(Sections 5.2.3 and 5.4.2). This systematic relationship suggests that it may be possible to improve the lineal bound by applying a factor related to the nominal value of $S+D$ (Appendix B.3).

The lineal bound is based on the *total variation* V (Section 3.3.2) of the image function and can be computed graphically (Figure 3.6) or analytically (Equation 3.3-10). In either case, it provides the following bound on the optimal mean square error in the x -direction, (Equation 3.3-12)

$$\text{MSE}_x \geq \frac{1}{12V^2}.$$

The *position decoding* algorithm (Chapter 4) was devised to achieve the optimal position estimation precision in the absence of noise. Using a target model, the algorithm determines the *feasible regions* (Section 4.3.1) in target position space which can give rise to the observed pixel values in the digital image. These feasible regions are intersected to determine the target locale ℓ . The centroid $(\bar{x}_\ell, \bar{y}_\ell)$ of the locale ℓ is computed to obtain the minimal mean square error position estimate. Thus, position decoding attains the optimal position estimation precision of Equation 3.2-4. The size of the locale, as determined by the position decoding algorithm, indicates the position estimation uncertainty in the estimate due to quantization and sampling.

The position decoding algorithm employs a parallel *branch and bound* search strategy (Section 4.4.6) with a maximum-likelihood *cost function* (Sections 4.4.2 and 4.5.4) to achieve robustness in the presence of noise. Further robustness and computational efficiency are achieved by careful selection and ordering of the *active pixels* used by the algorithm (Section 4.5.3).

The computational complexity of the algorithm is greatly reduced by using *iterated resolution* and partial evaluation in the truth map (Section 4.3.5 and 4.3.6). To ensure that small locales are not missed, the position decoding algorithm performs *region dilation* (Section 4.3.4) prior to intersecting the feasible regions.

Simulations were conducted and are reported in Chapter 5. The gap between the lineal bound and the optimal estimation precision was determined for the Gaussian

dot (Section 5.2.3) and the binary edge targets (Section 5.4.2) using the position decoding algorithm. Optimality of the position decoding algorithm in the absence of noise is confirmed by comparing its performance to the predicted optimal offset estimation errors for a binary edge, as determined analytically by other researchers (Section 5.4.2).

The estimation precision of the position decoding algorithm, with and without noise, is compared to that of three other position estimation algorithms (Section 5.2) and position decoding is shown to have superior performance. The precision of the position decoding algorithm is shown to vary linearly with the amplitude of the noise. This *linearity* is observed at all target amplitudes for the Gaussian dot and (with higher noise levels) at all target lengths for the binary edge.

The stability of the position decoding algorithm in the presence of target *modelling errors* is tested by applying an amplitude scale-factor error (multiplicative noise) to the target model (Section 5.3). It is shown that the precision of the algorithm degrades gradually as the error magnitude is increased and the algorithm always succeeds to converge on a position estimate.

The addition of low amplitude noise to the binary edge is shown to diminish the *anisotropic* character of the offset and orientation estimation errors (Section 5.5.4), while having little effect on the average mean square error (Section 5.5.2). This is heuristically attributed to a *randomizing* of the locale pattern and suggests avenues for further research to define “noisy locales” and a modified lineal bound.

6.2 Future Research

A modified lineal bound. The lineal bound differs from the optimal precision by a factor of about 2.5 for the Gaussian dot and 5 for the binary edge. This gap is significant but, because it is systematic, it is possible to adjust the lineal bound by these factors to obtain a tighter bound on precision. Future research should investigate the class of targets to which such a modified lineal bound can be applied.

An alternate approach to the lineal bound is discussed in Appendix B.3. In this approach, the size and shape term ($S+D$) is presumed constant and the modified lineal bound is obtained from the ratio of the area-density to the lineal density of locales. Further research should investigate the variability of the term ($S + D$), particularly in the presence of noise. It is conjectured that the term is relatively constant.

Overall versus directional precision A better understanding of the relationship between the lineal density of locales and the area-density of locales is needed. This relationship has been investigated for radial symmetric targets such as the Gaussian dot and a simple expression has been found to relate them (Equation A.3-6) but relationships should be determined for a broader range of target forms.

The area density of locales is related to the overall precision while the lineal density is related to the directional precision. In the case of the binary edge, the anisotropic variations in precision for estimation of edge orientation and offset seem to counter-balance each other. In Figures 5.12 and 5.13, orientation precision increases where offset precision decreases and vice versa. This suggests that, although the overall precision may be relatively constant, the apportioning of available precision to the two position parameters changes significantly as the edge orientation changes. The overall precision may have a simpler relationship to spatial resolution, intensity resolution and noise than do the individual position parameters. Further investigation of the relationships between area-density and lineal density of locales may help clarify these issues and lead to a simple formulation of the optimal precision for individual position parameters.

Higher dimensional position spaces There are many applications involving targets with more than two position parameters, such the three-parameter position space of a circle with unknown position (x, y) and radius r . The simple method of computing the lineal bound may be applied to such targets but the tightness of the bound in higher dimensional spaces is yet undetermined. This issue requires further investigation.

Further research is needed to gain an understanding of the effects of systematic modelling errors on geometric precision. For targets with several position parameters, high precision may not be needed in all parameters. In such cases, some parameters may be roughly approximated and the others estimated to greater precision. Approximating target parameters in this way corresponds to committing systematic target modelling errors.

Noisy locales Further research is needed for the formulation of noise within the locale theory. A fruitful approach may be to formulate noise-induced errors in terms of aggregate locales. As discussed in Appendix B.5, errors due to noise can be expressed as an error-volume in image space. Future investigations should seek to determine the density of locales within these error-volumes in image space. If this is determined, then the (noise-free) locales theory can be applied to obtain estimates for the size of the resultant aggregate locales in position space.

The formulation of noise in terms of aggregate locales leads to the determination of the overall precision, rather than the directional precision. The relationships between area-density of locales and lineal density of locales, as mentioned above, will then be essential to obtain estimates of precision for individual position parameters.

Scattered locales Sections 2.4.4 and 4.4.4 show that aggregate locales may be scattered but the nature of this scattering has not been thoroughly investigated. Future research should investigate the frequency of occurrence of locale scattering and the "radius" of scattering to determine the extent of the effect on precision.

Randomized locale patterns In the presence of noise it may be possible to consider the locale pattern as being randomized in some way. Loosely speaking, when noise is present each "noisy" locale may be replaced by an aggregate locale, or a locale cluster. This aggregation will have an averaging effect on the shape and size parameters (S and D) of noisy locales. This suggests that it may be possible to predict the value of $(S + D)$ by considering a randomly generated locale pattern. Such

a pattern might be constructed, for example, by overlaying randomly placed line or arc segments. Statistical analysis of the shape and size parameters for such random locale patterns may lead to a natural definition of effective values for S and D in the presence of noise. These values can then be applied as in Equation 3.2-4 to obtain an estimate of the optimal precision in the presence of noise.

Implementing position decoding The position decoding algorithm is dependent upon an effective target model. In practical applications, the determination of an appropriate target model may be the primary impediment to effective implementation of the algorithm. The straight edge is a simple and tractable target for testing the position decoding algorithm with real-world data. Further research toward implementing position decoding in practical applications should emphasize edge targets. Starting with an accurately determined edge response function for the imaging system, a realistic target model for a physically straight edge can be obtained. With such a target model, the effectiveness of position decoding in the presence of real-world image data vagaries can be studied.

The computational efficiency of the program code used for the simulation studies can be greatly improved upon for practical implementation but, where speed is very important, computational parallelism may be required. One interesting approach to parallel implementation of position decoding is the utilization of neural network principles to enable the algorithm to “learn” the target model and perform position estimation rapidly.

Videometry Using precise algorithms such as position decoding may be futile if the underlying factors such as illumination, vibration control and system calibration are not at a commensurate level of accuracy. A major part of the measurement process is thorough characterization and quantification of the geometric properties of the image acquisition system. For this reason, future research into techniques for high precision position estimation in practical applications must involve videometry issues such as those mentioned in Section 1.3 and the references therein.

Performance benchmarks and geometric fidelity The position decoding algorithm is developed and demonstrated here primarily to support and illustrate the locales theory; nevertheless, the algorithm may be tremendously useful as a fundamental benchmark for position estimation. Further studies should be conducted to evaluate the universality of the algorithm and feasibility of using it as a benchmark algorithm.

The theoretical optimality of the position decoding algorithm may be used as a basis for a measure of geometric image quality. By applying the algorithm to a given image and comparing the observed estimation precision to the predicted optimal precision of the algorithm, a measure of geometric noise, or geometric image quality, can be obtained for the image. This idea might be applied to image transformations as well, by comparing the geometric quality of the input image to that of the output image. An objective determination of geometric image quality which is universally applicable and easily determinable would be very useful and perhaps it can be provided by the position decoding algorithm and the lineal bound.

6.3 Conclusions

The locales theory provides a rigorous, general and unifying theoretical framework for the analysis of position estimation precision in noise-free digital images. This theory is simple, elegant and fundamental to digital image metrology.

The optimal position estimator is given by the centroid of the target locale. The optimal mean square error obtained with this estimator applies to all shapes and sizes of targets, to all spatial and intensity quantization resolutions, and to generalized position parameters such as edge orientation.

Position decoding achieves the optimal position estimation precision in the absence of noise. This algorithm has been implemented, and its demonstrated robustness in the presence of noise invites its use in practical applications.

The lineal bound provides an easily computed bound on the optimal precision, and is a practical indicator of potential precision in position estimation problems.

The locales theory provides new analytical tools and a new position estimation algorithm. It provides new understanding of geometric precision in digital images and opens new avenues for future research.

Appendix A

DERIVATIONS AND PROOFS

A.1 COM Estimation Errors

In this appendix, the centre of mass (COM) position estimate for a spatially sampled image of a Gaussian dot target is considered. A closed-form expression is derived for the position estimate as a function of the actual target position.

Spatial sampling gives the continuous pixel values p_{ij} with

$$p_{ij} = A e^{-a(x-i)^2 - b(y-j)^2},$$

where A , a and b are positive real numbers, (x, y) is the position of the target centre, and (i, j) are the pixel indices from a three by three sampling window

$$\mathbf{W} = \{ (i, j) : i \in \{-1, 0, +1\}, j \in \{-1, 0, +1\} \}.$$

These pixel values are not quantized; they are real numbers. The centre of mass (COM) position estimate for the target is $(\hat{x}, \hat{y})$, with

$$\hat{x} = \frac{\sum i p_{ij}}{\sum p_{ij}} \quad \text{and} \quad \hat{y} = \frac{\sum j p_{ij}}{\sum p_{ij}},$$

where the summations are over the indices i and j in the sampling window. The

position estimate will be computed as a function of the target position.

$$\begin{aligned}
 \hat{x} &= \sum_{i=-1}^1 \sum_{j=-1}^1 i e^{-a(x-i)^2 - b(y-j)^2} \bigg/ \sum_{i=-1}^1 \sum_{j=-1}^1 e^{-a(x-i)^2 - b(y-j)^2} , \\
 &= \sum_{i=-1}^1 i e^{-a(x-i)^2} \bigg/ \sum_{i=-1}^1 e^{-a(x-i)^2} , \\
 &= \sum_{i=-1}^1 i e^{2axi - ai^2} \bigg/ \sum_{i=-1}^1 e^{2axi - ai^2} , \\
 &= \left[-e^{-2ax-a} + e^{2ax-a} \right] \bigg/ \left[e^{-2ax-a} + e^0 + e^{2ax-a} \right] , \\
 &= \frac{\sinh(2ax)}{\cosh(2ax) + \frac{1}{2}e^a} .
 \end{aligned}$$

Similarly, for the y -axis,

$$\hat{y} = \frac{\sinh(2by)}{\cosh(2by) + \frac{1}{2}e^b} . \quad (\text{A.1-1})$$

A.2 Periodicity of Locales

In this appendix it is shown that a translational target model (Section 2.1.4) with a sufficient sampling window (Section 2.2.1) has a locale pattern which is periodic with a period equal to the spacing in the pixel raster.

To prove periodicity of the locale pattern it must be shown that, for any two points $\vec{x}_1 = (x_1, y_1)$ and $\vec{x}_2 = (x_2, y_2)$ in the target position space $\mathbf{D}$, and for any integer translations $\vec{d} = (i_0, j_0)$ with $\vec{x}_1 + \vec{d}$ and $\vec{x}_2 + \vec{d}$ in $\mathbf{D}$,

$$\vec{x}_1 \equiv \vec{x}_2 \iff \vec{x}_1 + \vec{d} \equiv \vec{x}_2 + \vec{d} . \quad (\text{A.2-1})$$

Recall that the equivalence relation between points in $\mathbf{D}$ is defined by (Equation 2.2-3)

$$\vec{x}_1 \equiv \vec{x}_2 \iff \mathcal{I}_0(x_1 - i_k, y_1 - j_k) = \mathcal{I}_0(x_2 - i_k, y_2 - j_k) \quad \forall (i_k, j_k) \in \mathbf{W} , \quad (\text{A.2-2})$$

presuming a translational target model.

For a sufficient sampling window W ,

$$\mathcal{I}_0(x_1 - i, y_1 - j) = 0 = \mathcal{I}_0(x_2 - i, y_2 - j) \quad (i, j) \notin W.$$

Consequently, Equation A.2-2 can be written without restricting the indices (i_k, j_k) ,

$$\begin{aligned} \vec{x}_1 \equiv \vec{x}_2 &\iff \mathcal{I}_0(x_1 - i, y_1 - j) = \mathcal{I}_0(x_2 - i, y_2 - j) \quad \forall (i, j), \\ &\iff \mathcal{I}_0(x_1 - i_0 + i, y_1 - j_0 + j) = \mathcal{I}_0(x_2 - i_0 + i, y_2 - j_0 + j) \quad \forall (i, j), \\ &\iff \vec{x}_1 + \vec{d} \equiv \vec{x}_2 + \vec{d}, \end{aligned} \quad (\text{A.2-3})$$

proving periodicity of the locale pattern.

A.3 Area Density of Locales

This appendix begins with a discussion of the density of locales in image space (rather than in position space) to show that the locale density there depends upon the average number of edges, or locale neighbours, in position space. The discussion elaborates on the claim made in Section 3.3.1 that the area bound B_2 can be refined by a factor of $n/2K$, where K is the dimensionality of image space and n is the average number of locale neighbours.

In Section A.3.2 it is proven that the average number of edges for a locale in position space is four.

In Section A.3.3, the area-density of locales is expressed in terms of the lineal density and the overall length of the contours of the target model. To illustrate the results it is shown that a target which is circularly symmetric, such as the Gaussian dot has a locale area-density N_2 which is given in terms of the lineal density N_1 as $N_2 = \pi N_1^2 / 4$.

A.3.1 Locale density in image space

Image space is a K -dimensional vector space with integer coordinate values. The image space for a digitizing scheme which has K pixels and a set Q of possible

quantization values is denoted $\mathbf{Q}^K$. The set of all locales $\mathcal{L}$ is a subset of the image space. neighbours in $\mathbf{Q}^K$ are pairs of points separated by an image distance of one. Any point in the interior of the image space has $2K$ neighbours, corresponding to adding ± 1 to any one of its K pixel values. Assume that locales all lie in the interior of $\mathbf{Q}^K$. (This technical detail can be satisfied artificially by including two fictitious values $(+\infty$ and $-\infty)$ to the quantization set $\mathbf{Q}$.)

Let n_ℓ be the number of locales which neighbour on the locale ℓ . The average number of locale neighbours for locales is

$$n = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} n_\ell.$$

Similarly, let n_q be the number of locales which neighbour on an arbitrary non-locale point $q \in \mathbf{Q}^K$. The average number of locales adjacent to non-locales is

$$\bar{n} = \frac{1}{|\mathbf{Q}^K| - |\mathcal{L}|} \sum_{q \notin \mathcal{L}} n_q.$$

The number of neighbours of locales can be summed in two ways: either as, $2K |\mathcal{L}|$ or as $\bar{n}(|\mathbf{Q}^K| - |\mathcal{L}|) + n |\mathcal{L}|$. By assuming that locales are homogenously distributed in image space, so that $\bar{n}=n$, it follows that $2K |\mathcal{L}| = n |\mathbf{Q}^K|$; that is,

$$\frac{|\mathcal{L}|}{|\mathbf{Q}^K|} = \frac{n}{2K}. \quad (\text{A.3-1})$$

Intuitively, the assumption that $\bar{n}=n$ is not realistic. In a noisy image, one does not expect to be able to modify a single pixel value to obtain a noise-free rendition of the image. This means that there is no locale neighbouring on the (noisy) image, hence the number of locales neighbouring on an arbitrary point in image space is often zero and should be much less than one on the average. If it is presumed that $\bar{n} \ll n$ then

$$\bar{n} |\mathbf{Q}^K| + n |\mathcal{L}| \approx 2K |\mathcal{L}|, \quad (\text{A.3-2})$$

and so

$$\frac{|\mathcal{L}|}{|\mathbf{Q}^K|} \approx \frac{\bar{n}}{2K - n} \ll \frac{n}{2K}. \quad (\text{A.3-3})$$

In this case, the average locale density throughout image space is much less than $n/(2K)$, which is consistent with the bound B_2 on the number of locales in Section 3.3.1 being excessively large, even after refinement by the factor $n/(2K)$.

In the next section, it will be shown that, based on locale adjacency in position space, it is reasonable to assume that $n=4$.

A.3.2 Locale neighbours in position space

The locale map is constructed by overlaying smooth contour segments from the target image function. If it is assumed that no more than two contour segments intersect at a given point, then every vertex will belong to exactly four locales (*i.e.*, every vertex has four adjoining edges). The number of vertices on the boundary of a locale always equals the number of edges and every edge belongs to two locales, so it follows that the total number of edges E is twice the total number of vertices V ; $E = 2V$.

At each vertex of the locale map, the sum of the angles (in radians) formed by the tangents to the intersecting contours is 2π . For all the vertices in the locale map, the sum of all angles is $2\pi V$, or πE . Within each locale ℓ , with e_ℓ edges, the sum of the internal angles is $\pi(e_\ell - 2)$. If L is the total number of locales in the locale map then the sum of all of the internal angles is $\pi(2E - 2L)$. Equating the sum of all internal angles to the sum of angles at vertices gives $\pi(2E - 2L) = \pi E$; that is, $E = 2L$.

If the average number of edges per locale is $\bar{e}$, then $E = \bar{e}L/2$, since each edge belongs to two locales. Combining these relations, $E = 2L$ and $E = \bar{e}L/2$, gives

$$\bar{e} = 4. \quad (\text{A.3-4})$$

The assumption that all vertices belong to exactly four locales is not unreasonable for a "random" target. If the contours which make up the locale map are considered to be randomly generated (in some way), then the probability of three or more contour intersecting at the same point is zero. This means that, although targets (such as straight edges) can be contrived with contour patterns which do provide more than

four edges per vertex, the probability of a randomly selected target having such a vertex is zero.

A.3.3 Area-density of locales in position space

Let C be the total length of the contours comprising the target model and let N_l and N_a denote the lineal density and area-density respectively of locales in the locale map of the target. Let E denote the total number of edges in the locale map.

Assume that the lineal density is isotropic and that the locale map is generated by a complete sampling window. The locale map within a unit square can be constructed by cutting the target contours into short segments and overlaying them appropriately in the unit square. With this in mind, notice that the number of edges in the locale map is equal to the total length of the contours times the lineal density of locales;

$$E = N_l C$$

The average number of edges per locale is $\bar{e} = 2E/N_a$, so

$$N_a = \frac{2N_l C}{\bar{e}}. \quad (\text{A.3-5})$$

From the previous section, $\bar{e} = 4$. Now consider, as an example, a circularly symmetric target which is represented by a set of concentric circular contours with radii $\{r_k : 1 \leq k \leq K-1\}$ for which $C = 2\pi \sum r_k$. It was shown by Equation 3.4-4 that the lineal density of locales for such targets is given by $N_l = 4 \sum r_k$; so, for this class of targets the area-bound on locale density can be expressed in terms of the lineal density as

$$N_2 = \frac{\pi}{4} N_1^2. \quad (\text{A.3-6})$$

A.4 The Shape Parameter

For an M -dimensional position space, the *shape coefficient* s_ℓ of a locale ℓ is defined in terms of the second central moment $m_\ell^{(2)}$ and the locale size $|\chi_\ell|$ as

$$s_\ell = \frac{m_\ell^{(2)}}{|\chi_\ell|^\gamma} \quad \text{where } \gamma = \frac{M+2}{M}. \quad (\text{A.4-1})$$

It will be shown that this coefficient is independent of the size of the locale. Note that vectors (in this section only) are denoted by lowercase boldface type, such as $\mathbf{x}$.

Consider two regions ℓ and ℓ' in position space. The regions are said to have the same shape if they are identical except for translation and scaling: that is, ℓ' has the same shape as ℓ when

$$\ell' = \{ \mathbf{x} \in \mathbf{D} : \alpha^{-1}(\mathbf{x} - \mathbf{x}_0) \in \ell \} \quad (\text{A.4-2})$$

for some constant scalar α and some constant vector $\mathbf{x}_0$.

The sizes of the two locales are related by (cf. Equation 3.1-3)

$$\begin{aligned} |\chi_{\ell'}| &= \int \cdots \int_{\mathbf{x} \in \ell'} dx_1 \cdots dx_M, \\ &= \int \cdots \int_{\mathbf{x} \in \ell} \alpha^M dx_1 \cdots dx_M, \\ &= \alpha^M |\chi_\ell|. \end{aligned} \quad (\text{A.4-3})$$

The centroids $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$, for the locales ℓ and ℓ' respectively, are related by

$$\begin{aligned} \bar{\mathbf{x}}' &= |\chi_{\ell'}|^{-1} \int \cdots \int_{\mathbf{x} \in \ell'} \mathbf{x} d\mathbf{x}, \\ &= \alpha^{-M} |\chi_\ell|^{-1} \int \cdots \int_{\mathbf{x} \in \ell} (\alpha \mathbf{x} + \mathbf{x}_0) \alpha^M d\mathbf{x}, \\ &= \alpha \bar{\mathbf{x}} + \mathbf{x}_0. \end{aligned} \quad (\text{A.4-4})$$

Similarly, the second central moments are related by

$$\begin{aligned}
m_{\ell'}^{(2)} &= \int \cdots \int_{\mathbf{x} \in \ell'} \sum_{i=1}^M (x_i - \bar{x}_i')^2 d\mathbf{x} , \\
&= \int \cdots \int_{\mathbf{x} \in \ell'} \sum_{i=1}^M (\alpha x_i - \alpha \bar{x}_i')^2 \alpha^M d\mathbf{x} , \\
&= \alpha^{M+2} m_{\ell}^{(2)} .
\end{aligned} \tag{A.4-5}$$

With $\gamma = (M + 2)/M$, and using Equation A.4-3,

$$m_{\ell'}^{(2)} = \frac{|\lambda_{\ell'}|^\gamma}{|\lambda_{\ell}|^\gamma} m_{\ell}^{(2)} , \tag{A.4-6}$$

so that the shape coefficients

$$s_{\ell} = \frac{m_{\ell}^{(2)}}{|\lambda_{\ell}|^\gamma} \quad \text{and} \quad s_{\ell'} = \frac{m_{\ell'}^{(2)}}{|\lambda_{\ell'}|^\gamma} \tag{A.4-7}$$

are equal. Since ℓ and ℓ' are arbitrary locales with the same shape, it follows that the shape coefficient is independent of locale size.

A.5 The Optimal MSE and the Parameters S , D and N_a

Equation 3.2-4 expresses the mean square error (MSE) of the optimal position estimator in terms of the three parameters S , D and N_a . This equation is proven and generalized in this section. The three parameters are defined in Section 3.2.1 for two-dimensional position space. They are generalized here to M -dimensional position space here, as follows.

$$\frac{1}{S} = \frac{1}{N} \sum_{\ell \in \mathcal{C}} s_{\ell}^{-M/2} , \tag{A.5-1}$$

$$D = \frac{N_a}{A} \sum_{\ell \in \mathcal{C}} s_{\ell}^{M/2} \varepsilon_{\ell}^2 \quad \text{with} \quad \varepsilon_{\ell} = |\lambda_{\ell}| - \left(\frac{S}{N_a s_{\ell}^{M/2}} \right) , \tag{A.5-2}$$

and

$$N_a = \frac{N}{A} . \tag{A.5-3}$$

As in the two-dimensional case, N is the number of locales and A is the total area (or volume) of the position space. The shape distribution parameter D is a weighted variance of ϵ_ℓ which, as will be shown in a subsequent section, is the deviation of the locale sizes from the minimizing condition $\epsilon_\ell = 0$. Notice that S and D are dimensionless coefficients. Notice also that

$$\sum_{\ell \in \mathcal{L}} \epsilon_\ell = A - \frac{S}{N_a} \frac{N}{S} = 0. \quad (\text{A.5-4})$$

The mean square error MSE_0 for the optimal position estimator in M -dimensional space is (using Equations 3.1-6 and A.4-1)

$$\begin{aligned} \text{MSE}_0 &= \frac{1}{A} \sum_{\ell \in \mathcal{L}} m_\ell^{(2)}, \\ &= \frac{1}{A} \sum_{\ell \in \mathcal{L}} s_\ell |\chi_\ell|^\gamma. \end{aligned} \quad (\text{A.5-5})$$

The next section will use the Lagrange-multiplier technique to evaluate the locale sizes, for given locale shapes, which gives the minimum possible value for this optimal MSE. This set of locale sizes is then used in a Taylor-series expansion to obtain an expression for MSE_0 in terms of the parameters S , D and N_a . For the two-dimensional case, the Taylor expansion is exact.

A.5.1 Lagrange minimization

The optimal MSE, as a function of the locale sizes, has the form $T(\mathbf{x}) = \frac{1}{A} \sum_{i=1}^N s_i x_i^\gamma$, where $x_i = |\chi_i|$ for some indexing scheme on the set of locales. Using the method of Lagrange multipliers, this function of locale sizes can be minimized under the constraint that $\sum_{i=1}^N x_i = A$ by solving

$$\nabla \left(T(\hat{\mathbf{x}}) - \lambda \sum_{i=1}^N x_i \right) = \mathbf{0}. \quad (\text{A.5-6})$$

The solution requires that

$$\hat{x}_i = \left(\frac{\lambda}{\gamma s_i} \right)^{1/(\gamma-1)}. \quad (\text{A.5-7})$$

Using $\sum_{i=1}^N x_i = A$ to solve for the multiplier λ , substituting S into the equation, and using $1/(\gamma - 1) = M/2$ gives

$$\hat{x}_i = \frac{S}{N_a s_i^{M/2}}. \quad (\text{A.5-8})$$

The notation $|\hat{\chi}_\ell| = \hat{x}_i$ (where locale ℓ is associated with index i) is used for the locale size which solves the Lagrange minimization problem above. For $M=2$,

$$z_\ell = |\chi_\ell| - |\hat{\chi}_\ell|. \quad (\text{A.5-9})$$

Furthermore,

$$\begin{aligned} \text{MSE}_0 &= \frac{1}{A} \sum_{\ell \in \mathcal{L}} s_\ell |\chi_\ell|^\gamma, \\ &\geq \frac{1}{A} \sum_{\ell \in \mathcal{L}} s_\ell |\hat{\chi}_\ell|^\gamma, \\ &= \left(\frac{S}{N_a} \right)^{\gamma-1} \end{aligned} \quad (\text{A.5-10})$$

provides a lower bound on MSE_0 which is unattainable unless the locale sizes and shapes for the target happen to satisfy the minimizing condition $z_\ell = 0$ for all ℓ .

A.5.2 Taylor expansion

The form $T(\mathbf{x}) = (1/A) \sum_{i=1}^N s_i x_i^\gamma$ contains no cross-terms and so it has the simplified Taylor expansion about $\tilde{\mathbf{x}}$ given by

$$T(\tilde{\mathbf{x}} + \mathbf{z}) = \sum_{k=0}^{\infty} \sum_{i=1}^N \frac{z_i^k}{k!} \frac{\partial^k}{\partial x_i^k} T(\tilde{\mathbf{x}}). \quad (\text{A.5-11})$$

The first Taylor term is $T(\tilde{\mathbf{x}})$, which was shown above to be $(S/N_a)^{\gamma-1}$. The second term ($k=1$) is always zero, since

$$\begin{aligned} \frac{\gamma}{A} \sum_{i=1}^N z_i s_i \tilde{x}_i^{\gamma-1} &= \frac{\gamma}{A} \sum_{i=1}^N z_i s_i \left(\frac{S}{N_a s_i^{M/2}} \right)^{\gamma-1}, \\ &= \gamma \left(\frac{S}{N_a} \right)^{\gamma-2} \sum_{i=1}^N z_i, \end{aligned} \quad (\text{A.5-12})$$

and $\sum z_i = 0$ as shown earlier.

The third Taylor term is evaluated as follows.

$$\begin{aligned}
 \frac{\gamma(\gamma-1)}{2A} \sum_{i=1}^N z_i^2 s_i \tilde{x}_i^{\gamma-2} &= \frac{\gamma(\gamma-1)}{2A} \sum_{i=1}^N z_i^2 s_i \left(\frac{S}{N_a s_i^{M/2}} \right)^{\gamma-2}, \\
 &= \frac{\gamma(\gamma-1)}{2A} \left(\frac{S}{N_a} \right)^{\gamma-2} \sum_{i=1}^N s_i^{M/2} z_i^2, \\
 &= \frac{\gamma(\gamma-1)}{2A} \left(\frac{S}{N_a} \right)^{\gamma-2} \frac{D}{N_a}.
 \end{aligned} \tag{A.5-13}$$

Combining the first three terms of the Taylor expansion gives

$$\text{MSE}_0 \approx \frac{2S + \gamma(\gamma-1)D}{2N_a} \left(\frac{S}{N_a} \right)^{(\gamma-2)}. \tag{A.5-14}$$

When $M = 2$ (i.e., $\gamma = 2$), the function $T(\mathbf{x})$ is quadratic and all Taylor terms beyond the first three are zero. In this case, Equation 3.2-4 is obtained exactly;

$$\text{MSE}_0 = \frac{S + D}{N_a} \quad \text{when } M = 2. \tag{A.5-15}$$

A.6 Noise-Equivalent Target Parameters

This appendix discusses the noise-equivalent parameters A' , K'_θ and K'_b , which are presented in Section 5.5.3. For a given parameter A and noise N , the noise-equivalent parameter A' is defined such that the position estimation precision with the parameter value A' without noise is equal (or proportional) to the position estimation precision with the parameter value A and noise N . The main reasons for considering noise-equivalent parameters are (i) to explore the possibility of accommodating noisy data within the locales formalism by expressing the noise as a modification to the intensity or spatial resolution of the target, and (ii) to find a simple method for predicting the position estimation precision of the position decoding algorithm in the presence of noise.

The discussions in this section illustrate that simple expressions can be found for noise-equivalent parameters however the analysis does not, on its own merits, lead to conclusions about formalizing geometric precision in the presence of noise. It does provide analytic expressions for predicting performance for the specific target examples in question. Further research, beyond the scope of this thesis, is required in this area before substantive general conclusions can be made.

Noise-equivalent Amplitude for the Gaussian Dot For the Gaussian dot, the most direct approach to obtaining a noise-equivalent amplitude A' works well. In the absence of noise, the intensity resolution is equal to the number of quantization levels A , but when additive noise is present, the intensity resolution is A divided by the noise N . Plotting the RRMSE for the Gaussian dot as a function of A/N gives a good linear fit, however an even better fit is obtain by using $A/(N + 1)$, as displayed in Figure 5.21. (Note that the range of noise values in the simulation is actually $2N$.)

Strictly speaking, $A' = A/(2N + 1)$ is not a noise-equivalent amplitude since the slope of the curve in Figure 5.21 differs from that of the noise-free performance curve of Figure 5.4. The ratio of the slopes is about 3.7 : 5.0. This fact is not important here since the main issue is the existence of an effective noise-equivalent amplitude rather than the evaluation of its exact value.

The linearity of the RRMSE as a function of A' for the Gaussian dot suggests that noise can be treated conceptually as an amalgamation of locales in the locale map. If a noise level of N quantization levels is treated within the locales theory as though, on average, N^2 locales were amalgomated into one, then the theoretical performance for the Gaussian dot match the observed performance in the presence of noise. This interpretation of noise within the locales formalism is consistent with the assumption of a planar topology for the locales in image space, as discussed in Appendix B.5.

Incorporating noise into the locales formalism in this way, however, may not work for other target and position parameters. To explore this possibility (in a tentative and preliminary way), noise-equivalent edge lengths are sought for the binary edge.

The idea is to see if a reasonably linear relationship can be obtained over the range of results observed during the simulations, and to examine the functional form of the resultant noise-equivalent parameters.

Since the amount of data to be fitted is limited and the objectives are tentative, *ad hoc* linearizations are done. The algebraic details are presented in the subsequent subsections.

The fit obtained to linear performance (Figures 5.22 and 5.23) is quite good but the form of the expressions for the noise-equivalent edge lengths does not follow the paradigm of the Gaussian dot nor does it suggest an elegant alternative.

It appears that additive noise cannot be treated as a simple reduction in locale density by locale amalgamation for the binary edge, as it can for Gaussian dot, since the expressions for the noise-equivalent parameters are more complicated and have mixed terms in K and N . The underlying principles are not yet elucidated but the fact that essential linearity exists in all the performance curves suggests that a fundamental approach to treating noise for general target forms and position parameters may indeed exist.

Noise-equivalent length for edge orientation The RMSE in edge orientation estimation is modelled as a linear function of the noise amplitude N , with coefficients a_k and b_k which depend on the edge length K , as

$$\text{RMSE}_\theta = a_k + b_k N. \quad (\text{A.6-1})$$

Figure A.1 shows the offset a_k and slope b_k coefficients for the observed RMSE in orientation estimation for several edge lengths. Figure A.1 is obtained from Figure 5.19, using only the linear portion of each curve. The values of a_k and b_k are plotted as a function of K in Figure A.3. The graph suggests that a_k and b_k both vary roughly as $1/K$. As shown in the figure, the coefficients fit well to the relationships

$$\begin{aligned}\frac{1}{a_k} &= 0.32 + 5.6K, \\ \frac{1}{b_k} &= -4.0 + 1.8K.\end{aligned}\tag{A.6-2}$$

Substituting the above expressions for a_k and b_k into Equation A.6-1, and using $K > 2$ and $N < 1$ to eliminate insignificant terms, gives

$$\text{RMSE}_\theta \approx \frac{0.56K(N + 3.2) - 4.0}{(K - 1)^2 - 13} \stackrel{\text{def}}{=} \frac{1}{K'_\theta}.\tag{A.6-3}$$

Figure 5.22 shows the simulation data (with non-zero noise amplitude) plotted as a function of K'_θ .

Noise-equivalent length for edge offset Figure A.2 shows the offset and slope coefficients for the RMSE in edge offset estimation as a function of noise amplitude. The curves were obtained from the linear portions of Figure 5.20. The slopes of the curves are approximated by 0.22, so that

$$\text{RMSE}_b \approx a_k + 0.22N.\tag{A.6-4}$$

The offset coefficients a_k are determined from the noise-free performance curves illustrated in Figure 5.9,

$$\frac{1}{a_k} = 7.1 + 0.78K.\tag{A.6-5}$$

Combining these two equations gives

$$\text{RMSE}_b \approx \frac{0.22NK + 2.0N + 1.3}{K + 9.1} \stackrel{\text{def}}{=} \frac{1}{K'_b}.\tag{A.6-6}$$

Figure 5.23 shows the simulation data (with non-zero noise amplitude) plotted as a function of K'_b .

FIGURE A.1 The RRMSE for binary edge orientation estimation: the slope and intercept are calculated from the linear portions of Figure 5.19.

FIGURE A.2 The RRMSE for binary edge offset estimation: the slope and intercept are calculated from the linear portions of Figure 5.20.

FIGURE A.3 The noise-equivalent edge length for orientation estimation (K'_θ) is modelled as a linear function $K'_\theta = a_k + b_k N$. The coefficients a_k and b_k are estimated from Figure A.1. In the top graph, they resemble an inverse function of K . In the lower graphs, a linear fit is made for the curve $1/a_k$ versus K and for the curve $1/b_k$ versus K .

Appendix B

DISCUSSIONS

B.1 Example Target Model Transformations

In this appendix, each transformation and intermediary function in the sequence (see Section 2.1)

$$f_{\bar{x}_0}(x, y) \xrightarrow{\phi_1} \mathcal{O}_{\bar{x}_0}(x, y) \xrightarrow{\phi_2} E_{i,j}(x_0, y_0) \xrightarrow{\phi_3} \mathcal{I}_{i,j}(x_0, y_0) \xrightarrow{\Xi} \mathbf{I}_{\bar{x}_0} \quad (\text{B.1-1})$$

is discussed. An example functional form is presented to illustrate the nature of each transformation. The notation and terminology associated with these functions and transformations are used throughout the locales formalism and it is intended that this appendix will help clarify their meaning and usage.

It is emphasized that the locale formalism does not depend on any specific formulation of these transformation and that the formulations presented here are only examples.

B.1.1 Optical blurring and distortion

The symbol Φ_1 is used to denote the transformation from the brightness function f to the optical image function $\mathcal{O}$,

$$\Phi_1 : f_{\bar{x}_0}(x, y) \mapsto \mathcal{O}_{\bar{x}_0}(x, y). \quad (\text{B.1-2})$$

The vector $\bar{x}_0$ is the target position. The coordinates (x, y) refer to position in the object plane when they are used as variables of the brightness function f and they refer to position in the image plane when used as variables of the optical image function $\mathcal{O}$.

The transformation Φ_1 incorporates the optical characteristics of the imaging system, which are typically described in terms of a point spread function (Figure 2.3), and distortion (Figure 2.4). The point spread function, depicted in Figure 2.3, is applied as a convolution to the target brightness function. Distortions (Figure 2.4) tend to vary gradually over the field of view of the imaging system and it is realistic to assume linearity or local constancy.

An optical image of the target might be expressed in the form

$$\mathcal{O}_{\bar{x}_0}(\rho x, \rho y) = \int_{x'} \int_{y'} S(x - x', y - y') f_{\bar{x}_0}(x', y') dx' dy'. \quad (\text{B.1-3})$$

The radial distortion ρ is expressed as a polynomial [Slama80, Eq. 4.22]

$$\rho = \alpha_0 + \alpha_1 r + \alpha_2 r^3 + \cdots + \alpha_n r^n, \quad r^2 = x_0^2 + y_0^2. \quad (\text{B.1-4})$$

with all terms anti-symmetric, except the first term. The point spread function S can be expressed as an Airy distribution [Hecht79, Eq. 10.56] or, approximated with a Gaussian distribution

$$S(x, y) = a e^{-(x^2 + y^2)/(2\sigma^2)}. \quad (\text{B.1-5})$$

If the radial distortion function is approximated locally by a constant, then the optical image function $\mathcal{O}$ will be translation invariant whenever the brightness function is, giving

$$\mathcal{O}_{\bar{x}_0}(x, y) = \mathcal{O}_0(x - \rho x_0, y - \rho y_0). \quad (\text{B.1-6})$$

B.1.2 Photosensitivity and area averaging

The symbol Φ_2 is used to denote the transformation from the optical image function $\mathcal{O}$ to the entity function E ,

$$\Phi_2 : \mathcal{O}_{\vec{x}_0}(x, y) \mapsto E_{ij}(\vec{x}_0). \quad (\text{B.1-7})$$

The entity function is parametrized by the sensor indices (i, j) and is a continuous function of the continuous target position $\vec{x}_0$. The role of parameter and variable have been interchanged by the transformation from $\mathcal{O}$ to E .

Following [Chamb78], the entity function can be obtained from the optical image function in a form such as

$$E_{ij}(x_0, y_0) = \int_{x'} \int_{y'} A(\alpha_x i - x', \alpha_y j - y') T(x', y') \mathcal{O}_{\vec{x}_0}(x', y') dx' dy', \quad (\text{B.1-8})$$

where T is the optical transmittance function of the array surface, A is the aperture response function (including lateral diffusion) and $\vec{x}_0 = (x_0, y_0)$ is the target position. The coefficients α_x and α_y define the spacing of the sensors on the array.

The above formulation of the entity function is not generally translation invariant but since the transmittance function is periodic on a regular array,

$$T(x' + \alpha_x i, y' + \alpha_y j) = T(x', y') \quad \forall(i, j), \quad (\text{B.1-9})$$

E will in fact be translation invariant whenever $\mathcal{O}$ is. From Equations B.1-6, B.1-8 and B.1-9 it follows that

$$E_{ij}(x_0, y_0) = E_0(x_0 - \frac{\alpha_x}{\rho} i, y_0 - \frac{\alpha_y}{\rho} j), \quad (\text{B.1-10})$$

where ρ is the local radial distortion coefficient.

B.1.3 Filtering, resampling and quantization

The symbol Φ_3 is used to denote the transformation from the entity function E to the image function $\mathcal{I}$,

$$\Phi_3 : E_{ij}(x_0, y_0) \mapsto \mathcal{I}_{ij}(x_0, y_0).$$

The image function is parametrized by indices (i, j) , just as the entity function is, but if resampling is performed then the indexing sets may be different. The indices of E are for sensor elements while the indices of $\mathcal{I}$ are for image pixels. The image function has discrete values as a result of (intensity) quantization.

The sensor signals, represented by the entity function values, are read from the imaging array one at a time to form a time varying (video) waveform $V(t)$. The signal $E_{ij}(x_0, y_0)$ of sensor s_{ij} is read from the sensor array at time $t_{ij}^s = \lambda_1 i + \lambda_2 j$, producing an impulse response $h(t - t_{ij}^s)$. The impulse response usually resembles that of a low-pass filter and is presumed to be the same for all sensors. The resulting video signal $V(t)$ is expressed as the sum

$$V(t) = \sum_{ij} h(t - t_{ij}^s) E_{ij}(x_0, y_0).$$

This video signal is sampled at $(t_{ij}^p) = \lambda_3 i + \lambda_4 j$. When the two sets $\{t_{ij}^s\}$ and $\{t_{ij}^p\}$ differ, the video signal is said to be *resampled*. The new filtered, resampled and quantized digital representation of the target is expressed by the *image function*

$$\begin{aligned} \mathcal{I}_{ij}(x_0, y_0) &= \lfloor V(t_{ij}^p) \rfloor \\ &= \lfloor \sum_{kl} h(t_{ij}^p - t_{kl}^s) E_{kl}(x_0, y_0) \rfloor \end{aligned}$$

where $\lfloor \cdot \rfloor$ denotes quantization.

If there is no resampling, so that the sampling sets $\{t_{ij}^s\}$ and $\{t_{ij}^p\}$ are the same, then translation invariance of the image function can be shown as follows. Using translation invariance of the entity function (Equation B.1-10 with $\rho=1$),

$$\begin{aligned} \mathcal{I}_{ij}(x_0, y_0) &= \lfloor \sum_{kl} h(\lambda_1(i - k) + \lambda_2(j - l)) E_{kl}(x_0, y_0) \rfloor, \\ &= \lfloor \sum_{kl} h(\lambda_1(k) + \lambda_2(l)) E_{(i-k)(j-l)}(x_0, y_0) \rfloor, \\ &= \lfloor \sum_{kl} h(t_{kl}^s) E_0(x_0 - \alpha_x(i - k), y_0 - \alpha_y(j - l)) \rfloor, \\ &= \mathcal{I}_0(x_0 - \alpha_x i, y_0 - \alpha_y j). \end{aligned}$$

B.1.4 Digital storage

The symbol Ξ is used to denote the transformation from the image function $\mathcal{I}$ to the digital image $\mathbf{I}$,

$$\Xi : \mathcal{I}(x_0, y_0) \mapsto \mathbf{I}_{x_0}. \quad (\text{B.1-11})$$

This transformation is a notational formality. It transforms the functional form of the image function into an order set of pixel values by sampling them at a point (x_0, y_0) . This transformation explicitly selects the pixels of interest, namely those within the sampling window $\mathbf{W}$ (Section 2.2.1). The digital image is associated with a point in K -dimensional image space, where K is the number of samples in the sampling window.

B.2 Feasible Region Dilation

The resolution of the truth map may be quite low relative to the size of the target locales, until several truth map rescaling iterations have taken place (Section 4.3.5). Consequently, feasible region dilation must be performed so that small locales are not missed by the truth map evaluation (Section 4.3.4) .

Feasible region dilation is the same as conventional region dilation, as described in [Haral87]. In the position decoding algorithm, feasible region dilation is accomplished by applying a 3×3 maximizing filter to the *negative* of the auxiliary truth map T' (Section 4.3.3). For implementation with noisy data, T' is multi-valued, with the value in a cell representing the cost associated with the cell. The multi-valued T' consists of a sequence of nested regions of diminishing cost. The feasible region dilation algorithm enlarges each of the regions to form a new set of nested regions in T' .

The 3×3 maximizing filter is implemented *in-place* by using three temporary row buffers S_a , S_b and S_c for the partially processed data. Care is taken to make the algorithm efficient because it is executed many times during position decoding. An

individual row R_k of the truth map is processed column-wise by the filter to obtain three new rows R_{k-1} , S_a , and S_c , as follows. Let (j) indicate the column index for each row vector, then for $1 \leq j \leq M$

$$\begin{aligned} S_a(j) &\leftarrow \max_{|j-j'| \leq 1} \{ R_k(j') \} \\ S_c(j) &\leftarrow \max \{ S_c(j), S_b(j) \} \\ R_{k-1}(j) &\leftarrow \max \{ S_c(j), S_a(j) \}. \end{aligned} \quad (\text{B.2-1})$$

Once all column indices have been processed ($1 \leq j \leq M$), the three indices (a, b, c) for the temporary rows are rotated to (c, a, b) , the row index k is incremented by one and the above process repeats. The output R_{k-1} of the algorithm always lags one row behind the input R_k in the truth map, so that there is no (recursive) interaction in the data.

The algorithm treats boundary values, with indices outside of the legal range ($1 \leq k \leq N$ and $1 \leq j \leq M$), as $-\infty$ values. No more than four data comparisons are needed for each output value with this algorithm and, since it is done in-place, it uses a minimum amount of data storage.

B.3 The Modified Lineal Bound

The lineal bound on (directional) precision is based on the lineal density of locales and is expressed in terms of the total variation as $1/(12V^2)$ (Equation 3.3-12). It does not take into account the distribution of locale shapes and sizes, as prescribed by Equation 3.2-4 and so it is not surprising that there is a significant gap between the lineal bound and the optimal precision. For the Gaussian dot, this gap is a factor of 2.5 (Section 5.6). The purpose of a *modified lineal bound* is to reduce this gap and obtain a better estimate of the optimal (directional) precision in the form K_0/V^2 for some constant K_0 . Since the modified lineal bound is based on the total variation, it will have the same ease of computation as the lineal bound does.

An empirically determined modified lineal bound can be obtained by comparing

the position estimation errors observed in the simulation studies with the total variation provided by the locales theory for the Gaussian dot target. The mean square error in the x -direction obtained by simulation for the Gaussian dot is

$$\text{MSE}_x \approx \frac{1}{25A^2}, \quad (\text{B.3-1})$$

(see Figure 5.17). The total variation for the Gaussian dot is

$$V \approx \frac{5}{\sqrt{2}}(A - 1), \quad \text{for } \sigma^2 = \frac{1}{2}, \quad (\text{B.3-2})$$

(see Equation 3.4-4.) Combining the above relationships gives

$$\text{MSE}_x \approx \frac{1}{25(\frac{\sqrt{2}}{5}V + 1)^2} \approx \frac{1}{2V^2} \quad \text{for } A \gg 1. \quad (\text{B.3-3})$$

This expression is empirical rather than analytical and, unlike the lineal bound, it is an approximation to the optimal (directional) precision rather than a bound on it.

Another approach to the modified lineal bound is to estimate a value for the term $(S + D)$ in the expression for the optimal (overall) mean square error (Equation 3.2-4)

$$\text{MSE}_0 = \frac{S + D}{N_a} \quad (\text{B.3-4})$$

and to use the lineal density of locales N_l to determine a value for the area-density N_a . For the case of the Gaussian dot, the following three relationships are used to do this. First, following Appendix A.3.3 and using the fact that the total variation V approximates the lineal density of locales N_l ,

$$N_a = \frac{\pi}{4}N_l^2 \approx \frac{25\pi}{8}(A - 1)^2. \quad (\text{B.3-5})$$

Second, if the estimation errors are equally apportioned among the target position parameters, as they are for the x -direction and y -direction position parameters of the Gaussian dot, then the directional mean square error is half the overall mean square error,

$$\text{MSE}_x = \frac{1}{2}\text{MSE}_0 = \frac{S + D}{2N_a}. \quad (\text{B.3-6})$$

Finally, the observed values of the MSE_x for the Gaussian dot are provided by Equation B.3-1 above. Combining these three relationships gives

$$(S + D) \approx \frac{\pi(A - 1)^2}{4A^2} \approx \frac{\pi}{4}. \quad (\text{B.3-7})$$

Presuming that the value of $(S + D)$ is constant for a broad range of target shapes, a more general modified lineal bound is proposed.

$$MSE_x \approx \frac{\pi K_0}{8V^2}, \quad \text{with } K_0 = \frac{N_l^2}{N_a}. \quad (\text{B.3-8})$$

The constant K_0 can be determined from the contour pattern of the target, as is shown for the Gaussian dot in Appendix A.3.3.

It is emphasized that the modified lineal bound discussed in this section is an *estimate* of the optimal precision, rather than a bound on it. The extent to which the modified lineal bound can be applied to targets other than the Gaussian dot is left as a subject for further research.

B.4 Empty Resolution

Consider the problem of determining suitable values for spatial and intensity quantization resolution when designing an image acquisition system. Assume that a unit area array sensor has K^2 sensors arranged in a regular square raster. Assume that the maximum output from each sensor is 1.0 and that the noise level is $n = N_0 K$ for some constant N_0 . This condition on the noise level is consistent with the properties of shot noise in a solid state sensor. Further assume that uniform intensity quantization is performed with quantization intervals $e < n$, so that the number of quantization levels spanned by the noise is $N = n/e > 1$.

A target which spans the full range of intensity quantization levels has amplitude $A = 1/e$. The total variation for a Gaussian dot is approximately $V = 5\sigma A$ (Equation 3.4-4). The "radius" σ is proportional to K . Based on the simulation results

for the Gaussian dot in the presence of noise, the root mean square error RMSE in position estimation varies inversely with the noise level N .

Using these assumptions and facts, RMSE is expressed as

$$\text{RMSE} \propto \frac{V}{N} \propto \frac{\sigma A}{n/e} \propto \frac{K/e}{n/e} \propto \frac{1}{N_0}, \quad (\text{B.4-1})$$

indicating that, when $n > e$, neither spatial nor intensity resolution have any effect on position estimation precision. Although this may seem surprising, it results from the simple fact that increased spatial resolution is offset by an increased noise level, and increased intensity resolution amplifies the noise as much as the signal.

Empirical studies of the effects of aperture size on position estimation precision have verified that improving the spatial resolution beyond some level does not yield a significant improvement in precision [Uuruh82].

This being the case, the data volume needed to store the image without loss of geometric precision is minimized as follows. The spatial resolution should be reduced as much as possible without interfering with effective detection and recognition of the target. At the same time, the intensity resolution should be reduced so that $n/e \approx 1$. Whenever $n/e > 1$, so that precision is essentially unaffected by changes in resolution, the system can be said to have *empty resolution*.

B.5 Noise Tolerance and Locale Topology

In this appendix, the relationship between image noise and positional noise is discussed. Image noise is defined in terms of errors in the pixel values of the digital representation of a target. The image noise level determines an error-volume V_e in image space defined by the distribution of pixel error values associated with the noise. The size of V_e can be expressed as the number of image points within it.

Target position estimation errors determine an error-area A_e in target position space. The area of A_e can be expressed by the number of locales within it.

The problem is to relate the noise level to the size of A_e . The size of V_e is easily related to the noise level. For example, uniformly distributed additive noise, which is identically and independently distributed in the interval $[-N/2, +N/2]$ for each pixel, will result in an error-volume in image space Q^K of size N^K . This error-volume is a K -dimensional cube centred on the origin.

Each position space locale in the error-area A_e has a corresponding image space locale in V_e , so the size of A_e is related to the density in *image space* of locales within V_e . In Appendix A.3.1 the density of locales in image space is discussed. It is shown that the average number of locale neighbours determines a bound on the density (Equation [\[eq-A.3-1\]](#)). Using this to estimate the number of locales in V_e , the following inequality is obtained:

$$|A_e| \leq \frac{n}{2K} |V_e| = \frac{nN^K}{2K}, \quad (B.5-1)$$

where n is the number of locale neighbours (nominally, $n=4$) and $|\cdot|$ indicates the size of the error-area A_e and error-volume V_e respectively.

The above inequality permits that the size of the error-area A_e can grow as the K^{th} power of the noise amplitude N . This is contrary to the results of the simulation tests, where the root mean square directional estimation errors RMSE ϵ_r for the Gaussian dot increase linearly with noise amplitude. Linearly increasing estimation errors imply that the size of A_e increases quadratically; that is,

$$|A_e| \propto N^2. \quad (B.5-2)$$

This brings into question the topology of locales in image space, rather than merely their density. It has been argued in the summary of Section 2.4.3 that locale adjacency in position space and image space correspond. This means that the planar topology of locales in position space must be preserved in image space.

Although the locale topology in image space may be assumed planar, it may still be convoluted. Consider a sheet of paper, which is a plane in 3-dimensional space, crumpled tightly into a ball. The area of paper within a small cube inside this ball depends on the *volume* of the cube. By analogy, if the planar topology of locales

is extremely convoluted in image space then the number of locales within the error-volume V_e can grow as the K^{th} power of the noise level N .

On the other hand, assume that the locales in image space form a simple hyperplane (i.e., they are form a "flat" subset of image space $\mathbf{Q}^K$, without convolutions.). The number of locales intersecting the error-volume V_e will then be proportional to N^2 . Since this is more consistent with the simulation observations, it is concluded that the set of locales in image space are arranged more like a "flat" sheet than a "crumpled" one.

In summary, the locales formalism provides analytical methods for determining the density of locales in position space. If analytic methods are developed for determining the density of locales in image space, then a straight-forward specification of a noise distribution in image space can be readily interpreted as a positional-noise distribution in position space. This is one possible approach to accommodating noise within the formalism of the locales theory.

Appendix C

SOFTWARE LISTINGS

C.1 The Position Decoding Modules

```
/******  
*  
* COPYRIGHT (C) 1989 BY D.I. HAVELOCK  
*  
* Position Decoding.          2.11          D.Havelock  
*  
* REVISION HISTORY  
* v0.1  iterative increase in resolution (3X mag.) implemented.  
*       Program arguments used to establish object location.  
* v0.2  Catch small locales which might fall between truth map cells  
*       by using best value to define 'true'.  
*       Provide robustness for signal errors.  
* v0.32 catch floating point errors and shift debug levels.  
* v0.33 Provide random additive noise to the signal  
* v0.34 Track several candidate feasible regions to accommodate  
*       errors in the signal.  
* v1.00 Restructure so that DECODE is a callable utility in a separate  
*       file with processing parameters adjusted through external  
*       variables  
*  
* v1.10 Rescale TMap allowing it to grow outside of unit square.  
*       Return Error code for no locale found at all.  
*       Count levels of rescaling using XXX3.  
*       Adjust tolerance internally to account for gap in TMap due  
*       to its size by watching TMap_Worst_Dist.  
*       Allow 32767 range in TMap rather than 255.  
*       Provide screen display of locale estimation, with display  
*       toggled on and off by hitting the keyboard.  
*       Eliminate computation of error bounds.  
* v1.11 Use the tolerance level to accept N-best feasible levels in  
*       final selection of region in truth map at each resolution,  
*       rather than just taking the best.  
*  
* v2.00
```

```

* ===== Modify calling sequence to ENTITY FUNCTION to allow for non-
* translation invariant entity functions. Pixel index and TMap
* indices are now used as calling parameters. Sense of
* position estimate reversed.
* Provision for Entity_Function_Init to allow definition of
* lookup table on indices.
* External definition of position space coordinate ranges is
* made possible with the use of scale and offset initial value
* variables. Default setting are invoked (for centered unit
* square) by leaving TMap_Scale_... set to zero.
* Remove equilateral constraint on TMap scale factors.
* Video display routines are collected and removed to a separate
* source file. ('DCV_1.00.C').
* Provide character string to identify this S/R set: Decode_Ident
* v2.01 Convert debug code to run-time from Compile-time.
* Eliminate TMap rescaling for display from S/R Decode.
* Turn Video off at every start of Decode: Need keystroke to
* see the video show on every estimation point.
* v2.10 Adaptation for inclusion of Noise.
* A fraction of the range of values in the truth map are retained
* for noise tolerance. The fraction is TMap_Small, which
* is nominally 0.3.
* This is useful only for targets with large contrast.
* At least TMap_Tol extra levels are retained from index to index.
* The number of retained levels is reduced from TMap_Tol+1 until
* either only one level remains, or rescaling is effected.
* v2.11 Significant altering in the method of handling small TMaps. The
* feasible region for each pixel index is modified before it is
* incorporated into the truth map. It is modified by convolving it
* with a 3 by 3 'minimum' function. This has the effect of ensuring
* that feasibility region boundaries enclose all of the feasible
* region. It is important that it be applied at each index, rather
* than on the resultant map to minimize the possibility of losing
* locale 'between the cracks' of a small map.
* To accomplish this, the function "Swell" is added and an
* additional truth map for the image distances associated with each
* pixel index is defined.
* This modification may lead to obsolescence of existing
* "backtracking" and error tolerance techniques.
* */

/* *****
*
* EXTERNALS:
*
* Decode_Ident Character string identifying version of this S/R set.
* Entity_Function The model function for the signal. It returns an
* unsigned byte corresponding to the model function (entity
* function) for a specified pixel evaluated at the specified
* coordinate.
* Entity_Function_Init Definition of entity function in terms of indices
* allows (requires) mapping of indices to reflect changing
* scale and offsets in the truth map.
* Decode_Signal An array of unsigned pixel values representing the signal
* or target image.
* Decode_Index An array of integers indexing the pixels of the signal
* vector in the order in which they are to be used for position
* decoding.
* Decode_Length The number of pixels indexed by Decode_Index.
* Position_Est Real values 2-dimensional position estimate

```

```

***
*   TMap      A truth map (signed integer values) corresponding to the
*              probability of the signal being at the position associated
*              with the entries.
*   XMap      A feasibility map for individual pixels. Similar to TMap
*   TMap_Size  Defined expression for the number of cells along each side
*              of the truth map; the size of the truth map is the square of
*              this number. NOTE: the size should be greater than
*              (2./(1-TMap_Small)) to avoid endless rescale cycling.
*   TMap_Ovflo One more than the largest TMap value.
*   TMap_False Value assigned to non-feasible cells of the truth map.
*              Set to 0.
*   TMap_True  Value, or level above which the cell in the truth map is
*              considered as feasible.
*   TMap_Best  Highest value in truth map for current Decode_Index
*   TMap_Worst_Dist; Largest image distance encountered in TMap Scan.
*   TMap_Tol   Difference tolerated between the values of TMap_True
*              (lower) and TMap_Best (higher).
*   TMap_Rescaled; Flag value to indicate whether rescaling of the truth
*              map is needed.
*   TMap_Mass  Number of 'true' entries in truth table
*   TMap_Scale_X_Init; Initial Position space coordinate ranges...
*   TMap_Scale_Y_Init; Initial Position space coordinate ranges...
*   TMap_Offset_X_Init; Initial Position space coordinate ranges...
*   TMap_Offset_Y_Init; Initial Position space coordinate ranges...
*   TMap_Scale_X; TMap_Scale_Y The scale factor and the offsets used \
*   TMap_Offset_X; TMap_Offset_Y to derive position space coordinates \
*                               from the index of the truth map.
*   TMap_Max_J; TMap_Max_K; The range of non-zero values in TMap.
*   TMap_Min_J; TMap_Min_K;
*   TMap_Small; Fraction of width of TMap spanned by locale below which
*              rescaling is performed. Also determines fraction of range of
*              values appearing in map to be used as a 'gap' parameter to
*              ensure selection of index-wide best feasible regions.
*   TMap_Scale_Min; Smallest scale to which the truth map is rescaled,
*              effectively limiting the minimum size of resolvable locales.
*   XXX1,XXX2; Counters of calls to Entity_Function for each signal index,
*              and overall, respectively.
*   XXX3      Counter for number of TMap rescales performed.
*
** */

/* *****
*   DEBUG levels:
*   -1 no debugging output
*   0 test with no debugging internal to Decode.
*   1 Final output; position, locale map and position estimates printed.
*   2 Trace of truth map, and counting of cell tests
*   3 Full trace.
** */

#include <stdio.h>
#include <malloc.h>
#include <math.h>
#include <graph.h>
#include <conio.h>

```

```

/* *****
*
*  EXTERNAL DECLARATIONS AND DEFINITIONS.
*
** */

/* externals provided by Decode to the user */

char  *Decode_Ident="DC2.11";
int    Decode (void);
float  Position_Est[2] = {(float)0., (float)0.};
int    DEBUG=-1;

/* externals to be provided by the user to Decode */

extern int      Entity_Function_Init(void);
extern unsigned char  Entity_Function (int index, int tj, int tk);
extern unsigned char  Decode_Signal[];      /* size = Decode_Length */
extern int    Decode_Index[] [2];          /* size = Decode_Length */
extern int      Decode_Length;

/* 'Visible' external definitions which provide access to the      */
/* configuration of DECODE and to its status information            */

int      **TMap = NULL; /* Dynamically set to TMap[TMap_Size][TMap_Size] */
int      **XMap = NULL; /* As for TMap */
int      TMap_Size = 21;
int      TMap_Ovflo = 32767;
int      TMap_True = 23767;
int      TMap_False = 0;
int      TMap_Best = 0;
int      TMap_Worst_Dist = 32767;
int      TMap_Tol = 0;
int      TMap_Rescaled = 1;
int      TMap_Mass = 0;
int      TMap_Min_J, TMap_Max_J;
int      TMap_Min_K, TMap_Max_K;
long int  XXX1, XXX2, XXX3;
float     TMap_Small=(float)0.3;
float     TMap_Scale_Min = (float) .001;
float     TMap_Scale_X_Init = (float) 0.00; /* 0 => use unit square TMap */
float     TMap_Scale_Y_Init = (float) 0.00;
float     TMap_Offset_X_Init = -(float) 0.5;
float     TMap_Offset_Y_Init = -(float) 0.5;
float     TMap_Scale_X, TMap_Scale_Y;
float     TMap_Offset_X, TMap_Offset_Y;

extern void  Turn_Video_Off (void);
extern void  Turn_Video_On (void);
extern void  Video_Init (void);
extern void  Toggle_Video (void);
extern void  Show_Video (void);
extern int   Video_On;

/* 'Invisible' externals provide internal operating variables      */

static void  Display_TMap (int threshold);
static void  Display_XMap (int threshold);
static size_t Alloc_TMap (void);
static void  Reset_TMap (int set);

```

```

static int      Init_Decode (void);
static int      Rescale_TMap (int threshold);
static int      Scan_TMap (int index);
static int      Locale_Est (int threshold);
static void      Swell(int **);

/* *****
*   Alloc_TMap                                D.Havelock          29.1.88
*
*   Dynamically allocate space for, and vector, the storage for the truth
*   map array. The non-existence of memory allocation for TMap is detected
*   by a NULL value for the array pointer. If existing and correct
*   allocation is detected then no action is done. If an incorrect allocation
*   exists, then it deallocated and a new one made.
*
*   RETURNS
*   -1    no memory available for allocation
*   0     successful allocation made
*   n>0   insufficient memory available, only n bytes left.
*
** */
size_t Alloc_TMap()
{
    extern      **TMap, **XMap;
    static int  *tmap_storage = NULL;
    int         vsize=0, asize=0, i;

    if (TMap != NULL) vsize =(int)(_msize(TMap)/sizeof(int *));
    if (tmap_storage != NULL) asize =(int)(_msize(tmap_storage)/sizeof(int));

    /* check first for defective array storage allocations */
    if ( (TMap[0] != tmap_storage) || (vsize*vsize != asize/2) ||\
        (XMap != NULL && XMap[0] != &tmap_storage[TMap_Size*TMap_Size]) ) {
        printf(\
"\n*** WARNING <Alloc_TMap>: Unexpected allocation will not be freed\n");
        if (tmap_storage != NULL) free(tmap_storage);
        tmap_storage = NULL;
    }

    /* then check if different array size is requested */
    else if (tmap_storage != NULL && vsize != TMap_Size) {
        free(TMap);
        TMap = NULL;
        free(XMap);
        XMap = NULL;
        free(tmap_storage);
        tmap_storage = NULL;
    }

    /* now allocate space and map the vectors if need be */
    if (TMap == NULL) {
        tmap_storage = (int *) malloc((size_t) \
            (TMap_Size*TMap_Size*2) * sizeof(int));
        if (tmap_storage == NULL) {
            i=(int)_memavl();
            return((size_t)(i==0? -1: i));
        }
        TMap = (int **) malloc((size_t)TMap_Size*sizeof(int *));
        XMap = (int **) malloc((size_t)TMap_Size*sizeof(int *));
        if (TMap == NULL || XMap == NULL) {

```

```

        free(tmap_storage);
        tmap_storage = NULL;
        free(TMap);
        TMap = NULL;
        free(XMap);
        XMap = NULL;
        i=(int)_memavl();
        return((size_t)(i==0? -1: i));
    }

if (DEBUG >= 3) {
    printf("tmap allocated at %p\n",(int *far*)tmap_storage);
    printf(" vectoring pointers for TMap...   and XMap...\n");
}

    for (i=0; i<TMap_Size; i++) {
        TMap[i] = &tmap_storage[TMap_Size*i];
        XMap[i] = &tmap_storage[TMap_Size*(i+TMap_Size)];
if (DEBUG >= 3) {
    printf("    %p    %p\n", (int *far*)TMap[i], (int *far*)XMap[i]);
}
    }

    return(0);
}

/* *****
*   Init_Decode                      D.Havelock          07.11.88
*
*   Set the scale and offset so that the positions being sampled will
*   span the FULL width of the initial interval. Set all of the truth
*   map entries to true.
*   Calls Entity_Function_Init to establish any transformations on the
*   truth map indices, since rescaling alters the meaning of them.
*
*   RETURNS:
*       0 => A.O.K.
*      -1 => Fatal error, request exit.
** */

int Init_Decode()
{
    size_t mem_shortage;

    if ((mem_shortage=Alloc_TMap())!=0) {
        printf("\n *** ERROR <Init_Decode>: array alloc. failure\n");
        printf(" only %u bytes available\n",mem_shortage);
        return (-1);
    }
    if (TMap_Scale_X_Init==(float)0. || TMap_Scale_Y_Init==(float)0) {
        TMap_Scale_X = (float)1./(TMap_Size-1);
        TMap_Scale_Y = (float)1./(TMap_Size-1);
    }
    else {
        TMap_Scale_X = TMap_Scale_X_Init;
        TMap_Scale_Y = TMap_Scale_Y_Init;
    }
    TMap_Offset_X = TMap_Offset_X_Init;
    TMap_Offset_Y = TMap_Offset_Y_Init;
    TMap_Min_J = TMap_Min_K = 0;

```

```

TMap_Max_J = TMap_Max_K = TMap_Size-1;
TMap_Best = TMap_False;
TMap_True = TMap_Ovflo-1;
TMap_Rescaled = 1;
XXX1 = XXX2 = XXX3 = 0;
Reset_TMap(TMap_True);

/* Call to user supplied Entity function initialization */
if (Entity_Function_Init()) return(-1);

if (DEBUG >=2) {
{
int i;
printf ("Indexed signal array: \n");
for (i=0; i<Decode_Length; i++) printf ("%i, ",Decode_Signal[i]);
printf ("\n");
printf ("Index for pixels: \n");
for (i=0; i<Decode_Length; i++) {
printf ("(%i,%i), ",Decode_Index[i][0],Decode_Index[i][1]);
}
printf ("\n");
}
}

return(0);
}

/* *****
* Display_TMap D.Havelock 27.1.88
*
* Provide a simple tabular listing of the truth map with all values
* below the specified threshold being set to zero. Other entries are
* displayed as their distance from their largest possible value up to
* a maximum of 9.
*
** */

void Display_TMap(threshold)

int threshold;
{
extern int **TMap;
int tj, tk, i;

for (tj=0; tj<TMap_Size; tj++ ) {
for (tk=0; tk<TMap_Size; tk++ ) {
(TMap[tj][tk] > threshold) ? \
printf ("%i ",(i=TMap_Ovflo-TMap[tj][tk]-1) > 9 ? 9 : i) : \
printf (". ");
}
printf ("\n") ;
}
printf ("\n") ;
return;
}

/* *****
* Reset_TMap D.Havelock 12.11.87
*

```



```

* Initialize the truth map array to a specified value.
** */
void Reset_TMap (set)
    int      set;
{
    extern int **TMap;
    int      tj, tk;
    tj=tk=0;
    for (tj=tk=0; (tj<TMap_Size); (tk = ++tk%TMap_Size)!=0 ? tk : tj++) {
        TMap[tj][tk] = set;
    }
    TMap_Best = set;
    TMap_Worst_Dist = 0;
    return;
}

/* *****
* Swell                                D. Havelock          15.11.88
*
* Replace each entry with the local maximum in a 3 by 3 window. It does
* this by 'Maxing' three consecutive elements of each row, then pushing
* the resultant row onto a circular buffer of three rows. The three
* rows on the buffer are Maxed and output back to the original array.
* Effects at the edges of array (top, bottom, etc.) are handled by
* separate and explicit instructions outside of the inner loop.
* This technique accomplishes the maximum of 9 elements in 4 compares.
*
** */

void Swell(x)
    int      **x;
{
    extern int  TMap_Size, TMap_Ovflo;
    int      a, b, c, temp;
    int      lowest, end, tj, tk;
    static int **r = NULL;

    if (r==NULL) {
        r = (int **) malloc((size_t)3*sizeof(int *));
        if (r== NULL) {
            printf(" Error in <Swell> unable to get any memory.");
        }
        for (temp=0; temp<3; temp++){
            r[temp] = (int *) malloc((size_t)TMap_Size*sizeof(int));
            if (r[temp]==NULL) printf(" Error<Swell> unable to allocate mem.");
        }
    }
    a=0;
    b=1;
    c=2;
    lowest = -(TMap_Ovflo-1);
    end = TMap_Size-1;

    /* ** Set up recorded rows b and c, tending to end points separately ** */

    r[b][0]   = (x[0][0]   > x[0][1]) ? x[0][0]   : x[0][1];
    r[b][end] = (x[0][end] > x[0][end-1]) ? x[0][end] : x[0][end-1];
    r[c][0]   = lowest;
    r[c][end] = lowest;
    for (tk=1; tk<end; tk++){

```

```

    r[b][tk] = (x[0][tk-1] > x[0][tk]) ? x[0][tk-1] : x[0][tk];
    r[b][tk] = (x[0][tk+1] > r[b][tk]) ? x[0][tk+1] : r[b][tk];
    r[c][tk] = lowest;
}

/* Perform 3 by 3 maximization on central region and left right edges */
for (tj=1; tj<TMap_Size; tj++) {
    r[a][0] = (x[tj][0] > x[tj][1]) ? x[tj][0] : x[tj][1];
    r[c][0] = (r[c][0] > r[b][0]) ? r[c][0] : r[b][0];
    x[tj-1][0] = (r[c][0] > r[a][0]) ? r[c][0] : r[a][0];
    r[a][end] = (x[tj][end] > x[tj][end-1]) ? x[tj][end] : x[tj][end-1];
    r[c][end] = (r[c][end] > r[b][end]) ? r[c][end] : r[b][end];
    x[tj-1][end] = (r[c][end] > r[a][end]) ? r[c][end] : r[a][end];
    for (tk=1; tk<end; tk++) {
        r[a][tk] = (x[tj][tk-1] > x[tj][tk]) ? x[tj][tk-1] : x[tj][tk];
        r[a][tk] = (x[tj][tk+1] > r[a][tk]) ? x[tj][tk+1] : r[a][tk];
        r[c][tk] = (r[c][tk] > r[b][tk]) ? r[c][tk] : r[b][tk];
        x[tj-1][tk] = (r[c][tk] > r[a][tk]) ? r[c][tk] : r[a][tk];
    }
    temp=c; /* rotate circular buffer */
    c=b;
    b=a;
    a=temp;
}

/* Compute maximum for last row, tending to ends first */
x[end][0] = (r[a][0] > r[b][0]) ? r[a][0] : r[b][0];
x[end][end] = (r[a][end] > r[b][end]) ? r[a][end] : r[b][end];
for (tk=1; tk<end; tk++) {
    x[end][tk] = (r[c][tk] > r[b][tk]) ? r[c][tk] : r[b][tk];
}
}

/* *****
* Display_XMap                                D.Havelock          15.11.88
*
* Provide a simple tabular listing of the truth map with all values
* below the specified threshold being set to zero. Other entries are
* displayed as their distance from their largest possible value up to
* a maximum of 9.
*
** */

void Display_XMap(threshold)

    int    threshold;
{
    extern int  **XMap;
    int    tj, tk, i;

    for (tj=0; tj<TMap_Size; tj++) {
        for (tk=0; tk<TMap_Size; tk++) {
            (XMap[tj][tk] > threshold) ? \
                printf ("%i ", (i=TMap_Ovfilo-XMap[tj][tk]-1) > 9 ? 9 : i) : \
                printf (". ");
        }
        printf ("\n");
    }
}

```

```

    printf ("\n") ;
    return;
}

/* *****
*Rescale_TMap                                D.Havelock          07.11.88
*
* If the representation of the feasible region is small in the truth map,
* then the map is rescaled. Note that the original shape of the locale is
* lost, and the TMap is reset to TMap_True. Possible performance gains
* may be made by preserving the shape, swelling it appropriately of course.
* Rescaling the truth map requires that the entity function be
* reinitialized. Consequently, if rescaling is done, Entity_Function_Init
* is called.
*
* RETURN
* 0 => no rescaling done
* 1 => rescaled
* -1 => fatal error (from Entity_Function_Init)
*
** */

int Rescale_TMap(threshold)
    int threshold;
{
    extern int **TMap;
    int tj, tk;
    float small;
    /*
    * Establish the range of true cell indicies
    */
    TMap_Max_J=-1; TMap_Max_K=-1; TMap_Min_J=TMap_Size; TMap_Min_K=TMap_Size;
    for (tj=tk=0; (tj<TMap_Size); (tk = ++tk%TMap_Size) !=0 ? tk : tj++) {
        if (TMap[tj][tk] >= threshold) {
            TMap_Max_J = (tj>TMap_Max_J ? tj : TMap_Max_J);
            TMap_Min_J = (tj<TMap_Min_J ? tj : TMap_Min_J);
            TMap_Max_K = (tk>TMap_Max_K ? tk : TMap_Max_K);
            TMap_Min_K = (tk<TMap_Min_K ? tk : TMap_Min_K);
        }
    }
    /*
    * Check to see if rescaling to increase resolution in the TMap is necessary.
    */
    if ((TMap_Scale_X <= TMap_Scale_Min) || (TMap_Scale_Y <= TMap_Scale_Min)){
        TMap_Rescaled=0;
        return(0);
    }
    /*
    * Do not rescale if the range is not small in both directions
    */
    small = TMap_Small*(float)TMap_Size;
    if ( (float)(TMap_Max_J-TMap_Min_J) > small ||\
        (float)(TMap_Max_K-TMap_Min_K) > small ) {
        TMap_Rescaled=0;
        return(0);
    }

    /* When expanding the resolution, consideration is given to the fact that
    * the truth cells represent the feasibility at the CENTER of each cell. In
    * order not to exclude possible locale area, the feasible area is swelled

```

```

* somewhat.
*/
    TMap_Min_J -= 1;
    TMap_Max_J += 1;
    TMap_Min_K -= 1;
    TMap_Max_K += 1;
/*
*   note that if small>TMap_Size-2, we could rescale to the original !!!
*/
    TMap_Offset_X += TMap_Min_J * TMap_Scale_X;
    TMap_Scale_X *= (float)(TMap_Max_J - TMap_Min_J) / (float)(TMap_Size - 1);
    TMap_Offset_Y += TMap_Min_K * TMap_Scale_Y;
    TMap_Scale_Y *= (float)(TMap_Max_K - TMap_Min_K) / (float)(TMap_Size - 1);
/*
*   Reset map contents and levels
*/
    TMap_True = TMap_Ovfl-1;
    Reset_TMap(TMap_True);
    TMap_Rescaled = 1;
    ++XXX3;          /* Count number of re-scalings */

if (DEBUG >= 3) {
    printf("TMap rescaled as %f, %f, %f\n",\
        TMap_Scale_X, TMap_Offset_X, TMap_Offset_Y);
}

/*
*   Reinitialize the entity function
*/
    if (Entity_Function_Init()) return(-1);

    return(1);
}

/* *****
*   Scan_TMap          3.0          D.Havelock          15.11.88
*
*   For each element of the truth map, include the component of image
*   distance for the current pixel index by appropriately adjusting
*   (reducing) the value of the truth entry.
*   Only TMap entries which are sufficiently high are adjusted, all others
*   are reset to 'false'. By considering several of the top levels as 'true',
*   parallel treatment of candidate locales is accomplished (some degree of
*   back-tracking is effected).
*   The best and worst values in the truth map are re-established.
*
*   The position space coordinates for cell (tj,tk) of the truth map
*   are given by
*       tx = TMap_Scale_X*tj + TMap_Offset_X;
*       ty = TMap_Scale_Y*tk + TMap_Offset_Y;
*   The coordinate values must be determined by the Entity function. Only
*   the cell indices are passed so that the function may optimize
*   computational effort (by use of lookup table for eg.)
*
*   Individual feasibility regions are computed in "XMap". This map is
*   conditioned by expanding low-cost regions (function 'Swell').
*
*   RETURN:
*       0   => a.o.k.
*      -1   => TMap_True too low to proceed; it conflicts with TMap_False.

```

```

*
** */

int Scan_TMap(index)
    int    index;
{
    extern int  **TMap, **XMap;
    int        tj, tk, pj, pk;

    /* Pixel index being processed is (pj,pk) */
    pj = Decode_Index[index][0];
    pk = Decode_Index[index][1];

    /* Establish threshold for re-evaluating truth map entries */
    /* and reset the best entry value */

    if (TMap_True <= TMap_False) {
        TMap_True = TMap_False - 1;
        return(-1);
    }
    TMap_Best = TMap_False;
    TMap_Worst_Dist = 0;

    if (DEBUG >= 2) {
        printf ("Scanning index %i for pixel (%i,%i)\n",\
            index, pj, pk);
    }

    /* Select truth map cells currently above truth threshold */

    XXX1 = 0;          /* reset count of calls to entity function */
    for (tj=0; tj<TMap_Size; tj++) {
        for (tk=0; tk<TMap_Size; tk++) {
            if (TMap[tj][tk] >= TMap_True) {
                ++XXX1;      /* count entity calls for this index */
                ++XXX2;      /* count entity calls for all index */
            }
        }
    }

    if (DEBUG >= 3) {
        printf ("index %i=(%2i, %2i) cell: %2i, %2i,    s=%i, f=%i\n",\
            index, pj, pk, tj, tk,
            (int) Decode_Signal[index], \
            (int) Entity_Function(index, tj, tk));
    }

    /* Compute new truth (mis-match) component based on distance from */
    /* the entity function. */

    XMap[tj][tk] = -abs((int)(Decode_Signal[index] - \
        Entity_Function(index, tj, tk)));

    }
    else {
        TMap[tj][tk] = TMap_False;
        XMap[tj][tk] = -(TMap_Ovflo-1);
    }
}

/* Adjust the individual Feasibility region */
Swell (XMap);

```

```

/* Update the truth map with latest feasibility region */
for (tj=0; tj<TMap_Size; tj++) {
  for (tk=0; tk<TMap_Size; tk++) {
    if (XMap[tj][tk] > -(TMap_Ovflo-1)) {
      if ( (TMap[tj][tk] += XMap[tj][tk] ) > TMap_False) {
        TMap_Worst_Dist = (-XMap[tj][tk] > TMap_Worst_Dist ? \
                          -XMap[tj][tk] : TMap_Worst_Dist);
        TMap_Best = (TMap[tj][tk] > TMap_Best ? \
                    TMap[tj][tk] : TMap_Best);
      }
      else { TMap[tj][tk] = TMap_False; }
    }
  }
}

/* Has the cumulative image distance increased from last index to */
/* this one... */
if (TMap_Best < TMap_True + TMap_Tol) {
  if (TMap_Best <= TMap_False + TMap_Tol) { /* distance limit */
    printf(" *** WARNING *** \
           Active pixel %i = (%i,%i) is out of whack!\n",\
           index, pj, pk);
    printf("future versions may selectively ignore such pixels\n");
    return(0); /* Just warn; Next call may cause (-1) return. */
  }
}

/* ***if (DEBUG >= 1) {*** */
/* Notify of non-zero image distance: caused by normal noisy signal */
printf(" Pixel %i = (%i,%i) has bad fit...\n",\
       index, pj, pk);
printf("Distance to Entity Function: %i\n", TMap_True-TMap_Best);
/* ***}*** */
}
return(0);
}

/* *****
* Locale_Est          2.0          D.Havelock          07.11.88
*
* Estimate the centroid of the locale, based on its shape in the
* truth map.
* RETURN:
* 0 => a.o.k.
* -1 => No estimate is possible (TMap_Mass = 0).
*
** */

int Locale_Est(threshold)
int threshold;
{
  extern int **TMap;
  int tj, tk;
  int i; /* this may later be replaced by PDF */

  Position_Est[0]=Position_Est[1] = (float)0.0;
  TMap_Mass=0;
  for (tj=tk=0; (tj<TMap_Size); (tk = ++tk%TMap_Size)!=0 ? tk : tj++) {
    if (TMap[tj][tk] >= threshold) i=1;
    else i=0;
    Position_Est[0] += (float)(tj*i);
    Position_Est[1] += (float)(tk*i);
    TMap_Mass += i;
  }
}

```

```

    }
    if (TMap_Mass <= 0) {
        Position_Est[0] = (float)0;
        Position_Est[1] = (float)0;
        return (-1);
    }
    else {
        Position_Est[0] += TMap_Scale_X/(float)TMap_Mass;
        Position_Est[0] += TMap_Offset_X;
        Position_Est[1] += TMap_Scale_Y/(float)TMap_Mass;
        Position_Est[1] += TMap_Offset_Y;

        if (DEBUG >= 1) {
            printf ("est. and absolute certainty error (a.c.e.):\\
                \\n (%5f, %5f)\\n (%5f, %5f)\\n",\\
                Position_Est[0], Position_Est[1], \\
                (float)(TMap_Max_J-TMap_Min_J+1)*TMap_Scale_X,\\
                (float)(TMap_Max_K-TMap_Min_K+1)*TMap_Scale_Y);

            printf ("scale and offset:\\n %f, %f\\n %f, %f\\n",\\
                TMap_Scale_X, TMap_Scale_Y, TMap_Offset_X, TMap_Offset_Y) ;
        }
    }

    return(0);
}

/*****
*   Decode                                     D.Havelock       27.04.87
*
*   This routine determines the mean and variance of position for a given
*   signal image, based on a modelling function. The signal function is given
*   as an arbitrarily ordered vector of pixels values, with an associated index
*   relating the values to pixel image array coordinates.
*   RETURN:
*       0   => a.o.k.
*      -1   => Initialization failure.
*      -2   => Scan failure due to poor 'pixel fit'.
*      -3   => Estimate failure due to all-false TMap.
*
** */

int Decode ()
{
    int    index, gap;

    /* Establish run-time initialization for DECODE                                     */

    if (Init_Decode() ) return(-1);
    Video_Init();
    gap = 1;

    /* Iteratively increase the resolution of the truth map, by                         */
    /* rescaling until the locale is well resolved.                                   */

    while (TMap_Rescaled) {
        if (kbhit()) {
            getch();
            Toggle_Video();
        }
        gap = (int) (gap*TMap_Small);          /* Reduce gap to reflect rescaling */
    }
}

```

```

    TMap_Worst_Dist = 1;
    if (Video_On) {
        Toggle_Video();
        Toggle_Video();
    }

/* Index through the active pixels in the signal, further
/* restricting range of the locale is monitored to determine if
/* rescaling to a higher resolution is needed.
*/

    for (index=0; (index<Decode_Length); ++index) {
        if (gap<TMap_Worst_Dist) gap = TMap_Worst_Dist;
        TMap_True = TMap_Best - TMap_Tol - (int)(gap*TMap_Small);
        Show_Video();
        if (Scan_TMap(index)) {
            Turn_Video_Off();
            return (-2);
        }
    }

if (DEBUG >= 2) {
    printf ("cells evaluated; this pass, cumm., rescalings: \
%li, %li, %li\n",XXX1,XXX2,XXX3);
    printf ("TMap with scale and offset %f, (%f, %f)\n",\
        TMap_Scale_X, TMap_Offset_X, TMap_Offset_Y) ;
    Display_TMap(0);
}

    }

    TMap_True=TMap_Best;
    Show_Video();
    {
        int tol=TMap_Tol;
        do {
            TMap_Rescaled = Rescale_TMap(TMap_Best-tol); /* NOTE WELL */
        } while (--tol>=0 &&
            !TMap_Rescaled);
    } /* END BLOCK */

}

if (DEBUG >= 1) {
    printf ("\nFINAL TMap has scale and offset %f, (%f, %f)\n",\
        TMap_Scale_X, TMap_Offset_X, TMap_Offset_Y) ;
    printf ("Total cell tests: %li, with %li rescales.\n",XXX2,XXX3);
    if (DEBUG >= 2) {
        Display_TMap(0);
    }
}

if (TMap_Best == TMap_False) {
    Turn_Video_Off();
    return (-3);
}

if (Locale_Est(TMap_Best)) {
    Turn_Video_Off();
    return (-3);
}

if ( Video_On ) {
    Toggle_Video();
    Toggle_Video();
    Show_Video();
    Show_Video();
}

```



```

    if (Rohit()) {
        getch();
        Toggle_Video();
    }
    return (0);
}

```

C.2 The Truth Map Display Modules

```

/*****
* Video display routines for position decoding
*
* REVISION HISTORY
* v1.01 Initialize with video off. Remove full-screen rescaling.
*
** */

#include <stdio.h>
#include <graph.h>

/* Visible externals for interaction with the Video subroutines */

char *Decode_Video_Ident="DCV1.01";
void Turn_Video_Off (void);
void Turn_Video_On (void);
void Video_Init (void);
void Show_Video (void);
void Toggle_Video (void);
int Video_On = -1;

/* Externals used from Decode S/R package */

extern int **TMap, TMap_Size, TMap_True, TMap_Best;
extern float TMap_Scale_X, TMap_Scale_Y;
extern float TMap_Offset_X, TMap_Offset_Y;

/* 'Invisible' externals provide internal operating variables */

static short V_Mode = _HRESBW;
static short V_Color_Max = 1;
static double V_Scale_X = 640.;
static double V_Scale_Y = 200.;

/*****
* Show_Video D.Havelock 27.04.88
*
* Display non-false cells of the truth map as patches on the screen
* using configuration parameters established by Turn_Video_On.
** */
void Show_Video()
{
    double sx,sy,ex,ey;
    short tlx,tly,brx,bry,x,y,ssx,ssy;
    int i,j,tint;

```

```

/* ** for debug only ***
*   float TMap_Scale_X= (float)1./(TMap_Size-1);
*   float TMap_Scale_Y= (float)1./(TMap_Size-1);
*   float TMap_Offset_X=(float)-.5;
*   float TMap_Offset_Y=(float)-.5;
** */

    if (!Video_On) return;

    sx = (double)TMap_Scale_X * V_Scale_X;
    sy = (double)TMap_Scale_Y * V_Scale_Y;
    ssx = (short)sx;
    ssy = (short)sy;
/*
    The screen goes from object positions (x,y)=(-.5-ex , -5.-ey) to
    (x,y) = (+.5+ex , +.5+ey) where ex=ey=1./(2*(TMap_Size-1)) is the overlap
    of the truth map over the unit pixel at the lowest resolution.
    The following determines the location on the screen of the center of
    the top left (tl) truth cell.
*/
    ex = ey = 1./(2*(TMap_Size-1));
    x = (short) (((double)TMap_Offset_X + 0.5 + ex) * V_Scale_X);
    y = (short) (((double)TMap_Offset_Y + 0.5 + ey) * V_Scale_Y);

    for (j=0; j<TMap_Size; j++) {
        for (i=0; i<TMap_Size; i++) {
            if (TMap[i][j]>=TMap_True) {
                tint = V_Color_Max - (TMap_Best - TMap[i][j]);
            }
            else tint = 0;      /* clear all non-true cells */

            tlx = x + (short)(i*ssx-ssx/2.);
            tly = y + (short)(j*ssy-ssy/2.);
            brx = tlx + ssx;
            bry = tly + ssy;
            _setcolor( (short)( tint>0? tint: 0) );
            _rectangle(_GFILLINTERIOR,tlx,tly,brx,bry);
        }
        _rectangle(_GBORDER,x-(short)(sx/2.-1),y-(short)(sy/2.-1),\
            x+(short)(TMap_Size*ssx-ssx/2.+1),y+(short)(TMap_Size*ssy-ssy/2.+1) );
    }
    return;
}
/*****
*   Toggle_Video                                D.Havelock                27.04.87
*
*   Turn Video on or off.
** */
void Toggle_Video()
{
    if (Video_On) Turn_Video_Off();
    else Turn_Video_On();
    return;
}
/*****
*   Turn_Video_On                                D.Havelock                27.04.87
*
*   Turn Video on.
** */
void Turn_Video_On()
{

```

```

    if(Video_On) return;
    _setvideomode(V_Mode);
    _setcolor(0);
    _clearscreen(_GCLEARSCREEN);
    _setcolor (V_Color_Max);

    Video_On = (!1==1);
    return;
}
/*****
*   Turn_Video_Off                                D.Havelock          27.04.87
*
*   Turn Video off.
** */
void Turn_Video_Off()
{
    if(!Video_On) return;
    _setvideomode(_DEFAULTMODE);
    Video_On = (!1==1);
    return;
}
/*****
*   Video_Init                                    D.Havelock          27.04.87
*
*   Setup static parameters for displaying the locale region.
** */
void Video_Init()
{
    static int  videoset=0;
    if (videoset != 0) return;
    videoset = 1;
    V_Mode = _HRES16COLOR;
    V_Color_Max = 15;
    V_Scale_X = 640.*(1.-2./TMap_Size); /* should be 1/size, really */
    V_Scale_Y = 200.*(1.-2./TMap_Size);
    Video_On = (!1==1);                /* ensure initial state is 'off'*/
    return;
}

```

C.3 Test Software for the Gaussian Dot Target

```

/* DCTEST versions                                D.Havelock
*
* Driving routine for position decoding. It includes the definition of the
* entity function.
* Written with DC1_00.c, the position decoding package.
* THE TARGET MODEL CALLING SEQUENCE OF DC1_00 DIFFERS FROM THAT OF DC2_00.
*
* 1.00a was written to test the effect of various noise levels on the
* precision of position decoding for the standard Gaussian entity function.
*
** */

#include <math.h>
#include <stdio.h>

```

```

#include <stdlib.h>

#define ACTIVE_PIXELS 9
#define DEBUG 3
#ifdef (DEBUG)
#else
    #define DEBUG -1
#endif

/* *****
 *
 *   EXTERNALS
 *
 ** */

/* externals specific to Position Decoding Package */

int          Decode(void);
unsigned char Entity_Function(double x, double y);

unsigned char Decode_Signal[ACTIVE_PIXELS];
int          Decode_Index[ACTIVE_PIXELS][2]={
    -1,0,  1,0,  0,1,  0,-1,
    -1,-1, -1,1,  1,-1,  1,1,  0,0 };
int          Decode_Length = ACTIVE_PIXELS;
extern float Position_Est[2];
extern float Position_Est_Err[2];
extern int   TMap_Tol;      /* tolerance for each pixel fit, so that */
                           /* the n-best locales are monitored.    */

/* externals specific to this file */

void          main(int , char **, char **);
static void    Test_Decode(float *, float *, float *, float *, double);
static double  Noise(double);
static unsigned char Local_Entity(double x, double y, double z);

static float   Amp, Amp_Min, Amp_Max, Amp_Inc;
static int     Mesh_Size, Tolerance;

/* *****
 *   Calling Arguments for MAIN:
 *   argv[0] points to task name in invocation
 *   argv[1] points to "start amplitude for entity function"
 *   argv[2] points to "amplitude maximum"
 *   argv[3] points to "amplitude increment"
 *   argv[4] points to "start amplitude of noise"
 *   argv[5] points to "noise maximum"
 *   argv[6] points to "noise increment"
 *   argv[7] points to "Number of iterations along X and Y"
 *   argv[8] points to "Tolerance in the truth Map"
 ** */

void main(argc, argv, envp)
    int argc;
    char **argv;
    char **envp;
{
    extern float   Amp, Amp_Min, Amp_Max, Amp_Inc;
    extern int     Mesh_Size, Tolerance;

```

```

float    err_x, err_y, rms_x, rms_y;
double   nse, nse_min, nse_max, nse_inc;

/*
 *               print out the argument list for this program
 */
printf("    *** Position Decoding Test a *** \n");
printf("        Using DC Version 1.00      \n");
printf("                                \n");

envp=envp;
if (argc < 8) {
    printf("Input Amp, Amp_Max, Amp_Inc,\n \
noise, noise_max, noise_inc,\n \
Mesh_Size, Tolerance:\n");
    scanf("%f, %f, %f, %lf, %lf, %lf, %i, %i",\
        &Amp_Min, &Amp_Max, &Amp_Inc, \
        &nse_min, &nse_max, &nse_inc, \
        &Mesh_Size, &Tolerance);
}
else {
    Amp_Min = (float)atof(argv[1]);
    Amp_Max = (float)atof(argv[2]);
    Amp_Inc = (float)atof(argv[3]);
    nse_min = atof(argv[4]);
    nse_max = atof(argv[5]);
    nse_inc = atof(argv[6]);
    Mesh_Size = atoi(argv[7]);
    Tolerance = atoi(argv[8]);
}

#ifdef DEBUG > 2
    printf("\nAmp_Min: %f, Amp_Max: %f, Amp_Inc: %f\n \
nse_min: %f, nse_max: %f, nse_inc: %f\n \
Mesh_Size: %i, Tolerance: %i\n",\
        Amp_Min, Amp_Max, Amp_Inc,\
        nse_min, nse_max, nse_inc, Mesh_Size, Tolerance);
#endif

/*
 * configure position decoding package through external constants
 */
TMap_Tol = Tolerance;

/*
 * call driving routine to test the position decoding software.
 */
printf ("Amp, Mesh_Size, Error (x), Error (y),    Rms (x), Rms (y)\n");
printf ("-----\n");
for ( nse = nse_min; nse <= nse_max; nse += nse_inc) {
    for ( Amp = Amp_Min; Amp <= Amp_Max; Amp += Amp_Inc) {
        Test_Decode (&err_x, &err_y, &rms_x, &rms_y, nse);
        printf ("%f , %f , %f, %f,    %f, %f\n", Amp, (float)nse,\
            err_x, err_y, rms_x, rms_y);
    }
}

    exit(0);
}

/* *****
 *   Noise                1.0                D.Havelock                12.11.87
 *
 * generate signed noise of specified amplitude
 ** */

```

```

double Noise(z)
    double z;

{
    extern int rand();
    double temp;

    temp = z*((double)rand()/16384.-1);
    return( temp );
}

/* *****
*   Entity_Function           1.0           D.Havelock           12.11.87
*
*   For a given coordinate value, an integer representing the sampled value
*   of the model of the image of the signal is returned.
** */

unsigned char Entity_Function(x, y)
    double x,y;
{
    double pixel,temp;
    extern float Amp;

    temp = -((x*x)+(y*y));
    pixel = Amp*exp(temp);
    return ((unsigned char)pixel);
}

/* *****
*   Local_Entity             1.0           D.Havelock           12.11.87
*
*   For a given coordinate value, an integer representing the sampled value
*   of the model of the image of the signal with added noise, at the level
*   indicated by z is returned.
** */

unsigned char Local_Entity(x, y, z)
    double x,y,z;
{
    double pixel,temp,temp2;
    extern float Amp;

    temp = -((x*x)+(y*y));
    temp2 = Noise(z);
    pixel = Amp*exp(temp) + temp2;
    pixel = pixel>0. ? pixel : 0.; /* signal should not be negative */
    pixel = pixel<255. ? pixel : 255.; /* signal should not exceed byte */
    return ((unsigned char)pixel);
}

/* *****
*   Test_Decode              1.0           D.Havelock           Date
*
*   Driving routine to test the position decoding algorithm.
** */

void Test_Decode(err_x, err_y, rms_x, rms_y, z)
    float *err_x, *err_y, *rms_x, *rms_y;

```

```

double      z;
{

extern unsigned char Decode_Signal [];
extern int      Decode_Index [] [2], Decode_Length;
extern float    Amp;
extern int      Mesh_Size;

extern float    Position_Est[2];
extern float    Position_Est_Err[2];
extern int      index;
extern float    x, y, at_x, at_y, mesh_step;

mesh_step = (float)1./Mesh_Size;
*err_x = *err_y = (float)0.0;
*rms_x = *rms_y = (float)0.0;
for (at_y = -(float)0.5+(mesh_step/(float)4.); \
     at_y < (float)0.5; at_y += mesh_step) {
    for (at_x = -(float)0.5+(mesh_step/4.);
         at_x < (float)0.5; at_x += mesh_step) {
        for (index=0; index<Decode_Length; index++) {
            x = (float)Decode_Index[index][0] + at_x;
            y = (float)Decode_Index[index][1] + at_y;
            Decode_Signal[index] = (Local_Entity ((double)x,(double)y,z));
        }
        if (Decode ()) {
            printf("Decode failed... error values will be biased.\n");
        }
        *err_x += (at_x - Position_Est[0]);
        *err_y += (at_y - Position_Est[1]);
        *rms_x += (at_x - Position_Est[0])*(at_x - Position_Est[0]);
        *rms_y += (at_y - Position_Est[1])*(at_y - Position_Est[1]);
    }
    *rms_x = (float)sqrt ((double) (*rms_x)) / (float)Mesh_Size;
    *rms_y = (float)sqrt ((double) (*rms_y)) / (float)Mesh_Size;
    *err_x /= (float)Mesh_Size;
    *err_y /= (float)Mesh_Size;
}

```

C.4 Test Software for the Binary Edge Target

```

/* EDGES.C version 1.21      D.Havelock
*
* Driving routine for position decoding. It includes the definition of the
* entity function and entity function initialization routine.
* Written with DC2_01.c and DCV1_00.c, the position decoding package.
*
* Modification History:
*
* V1.01 Debug conditionals converted from compile-time to run-time
*       Rescaling of TMap for display eliminated from Decode.
* V1.02 12 Nov. 1988
*       Print out input parameters always.
* V1.10 Include Noise; Input parameter Noise_b, representing the translational

```

```

*      noise in the offset "b" of the edge. This form of noise is intended
*      for use with binary edges. Grey-level edges should have noise in
*      amplitude (in addition to this?).
*      Default Histo.Max implemented if entered value is <=0.
*      Corrected averaging of call count to provide number of calls per point.
* V1.11 Count, rather than report, the locale selection errors.
* V1.20 Special edition to examine average amplitude errors.
* V1.21 Fix erroneous duplicate reporting of x-Magnitude errors so that
*      y-mag. errors are reported too.
* V1.22 Change target position space to correspond to that of Dorst & Druin,
*      namely,  $0 < b < 1/\cos(\theta)$ .
* V1.23 Increase target position grid by a factor of 10 in the offset
*      direction to get very good estimates of the average magnitude errors.
* v1.30 Constrain estimate to finding offset with known orientation in order to
*      be compatible with the results of Dorst & Duin (Spirograph theory).
*
** */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <fcntl.h>

#define ACTIVE_PIXELS 100
#define EDGE_LEVELS 1
#define TMAP_TABLES_SIZE 100
#define HISTO_SIZE 1000

/* *****
*
* EXTERNALS
*
** */

/* externals specific to Position Decoding Package */

int          Decode (void);
void         Turn_Video_Off (void);
int          Entity_Function_Init (void);
unsigned char Entity_Function (int index, int tj, int tk);

unsigned char Decode_Signal[ACTIVE_PIXELS];
int          Decode_Index[ACTIVE_PIXELS][2];
int          Decode_Length = ACTIVE_PIXELS;

extern char  *Decode_Ident;
extern char  *Decode_Video_Ident;
extern float Position_Est[2];
extern int   TMap_Tol;          /* tolerance for each pixel fit, so that */
                                /* the n-best locales are monitored. */
extern int   TMap_Size;        /* Number of cells across a side
of */

                                /* the truth map */
extern long int  XXX2;         /* total of calls to entity function. */

extern float  TMap_Offset_X;
extern float  TMap_Offset_Y;
extern float  TMap_Scale_X;
extern float  TMap_Scale_Y;
extern int    TMap_Min_J;
extern int    TMap_Max_J;

```



```

extern int      TMap_Min_K;
extern int      TMap_Max_K;

extern int      DEBUG;

/* externals specific to this file */

static double   Edge_Level[EDGE_LEVELS]= {0.0};
static double   c[TMAP_TABLES_SIZE];
static double   s[TMAP_TABLES_SIZE];
static double   b[TMAP_TABLES_SIZE];
static double   Noise_b;
static int      Mesh_Size, Tolerance, Map_Size;
static double   Total_Calls;
static int      Histo[HISTO_SIZE], Histo_Size=100;
static int      Select_Errors;
static float    Histo_Max;
static double   Err_Max_x, Err_Rms_x, Err_Bias_x;
static double   Err_Max_y, Err_Rms_y, Err_Bias_y;
static double   Avg1_Max_x, Avg1_Rms_x, Avg1_Bias_x;
static double   Avg1_Max_y, Avg1_Rms_y, Avg1_Bias_y;
static double   At_X, At_Y;
char            *Edges_Ident="edges v1.23";

void            main (int , char **, char **);
static int      Get_Params (int argc, char **argv, char **envp);
static void     Test_Decode (void);
static void     Report(int);
static double   Noise (double);
int             Target (double x, double y, double z);
static void     Print_Histo (double);

/* *****
 * Calling Arguments for MAIN:
 *   argv[0] points to task name in invocation
 *   argv[1] points to "Number of iterations along X and Y"
 *   argv[2] points to "Tolerance in the truth Map"
 *   argv[3] points to "Truth-Map Size"
 *   argv[4] points to "Target size"
 *   argv[5] points to "Number of cells in the histogram"
 *   argv[6] points to "Range of histogram"
 ** */

void main(argc, argv, envp)
    int argc;
    char **argv;
    char **envp;
{
    extern int      Tolerance;
    extern int      Map_Size;
    extern double   Total_Calls;

    time_t ltime;

/*
 * Identity this run.
 */
    time(&ltime);
    printf( "This run performed %s\n", ctime(&ltime));
    printf("\n");

```

```

printf("    *** Position Decoding Test  %s          *** \n",\
      Edges_Ident);
printf("    ***   - Using DC Version %s -          *** \n",\
      Decode_Ident);
printf("    ***           and Video S/R Version %s -    *** \n",\
      Decode_Video_Ident);
printf("\n");

/*
 * Get processing parameters.
 */
if (Get_Params(argc, argv, envp)) exit(-1);
/*
 * Configure position decoding package with provided parameters.
 */
  TMap_Tol = Tolerance;
  TMap_Size = Map_Size;
/*
 * call driving routine to test the position decoding software.
 */
if (DEBUG > 1) {
  printf ("    ...<main>: ready to call Test_Decode\n");
}

  At_Y = At_Y;          /* Just 'cause 'taint used nowhere */
  Total_Calls=0.;
  Test_Decode ();
  exit(0);
}

/* *****
 * Report                                D. Havelock                09.11.88
 *
 * Format output to report statistics of the run. Parameter 'type' indicates
 * report function to be performed.
 *
 * 0 => banner header
 * 1 => intermediary printout (average in y only)
 * 10 => summary for scanned mesh
 * 20 => summary for group of scans
 * 100=> Histogram.
 * 900=> final statistics.
 *
 ** */
void Report(type)
  int type;
{
  extern int Mesh_Size;
  extern int Histo[], Histo_Size;
  extern float Histo_Max;
  extern double Err_Max_x, Err_Rms_x, Err_Bias_x;
  extern double Err_Max_y, Err_Rms_y, Err_Bias_y;
  extern double Total_Calls;
  extern double At_x, At_Y;

  if (DEBUG > 1) {
    printf ("Entered <Report(%i)>\n",type);
  }
  /*
 * Type 0
 */

```

```

        if (type==0) {
            printf (\
"Max.(x), Rms.(x),  MAG.(x), Max.(y), Rms.(y), MAG.(y), Theta/pi4, calls\n");
            printf (\
"-----\n");
        }
/*
* Type 1
*/
    if (type==1) {
        printf ("%6.4f , %6.4f , %6.4f | %6.4f , %6.4f , %6.4f | %.3f, %6.0f\n", \
                Err_Max_x, Err_Rms_x, Err_Bias_x, \
                Err_Max_y, Err_Rms_y, Err_Bias_y, \
                At_X, Total_Calls);
    }
/*
* Type 10 or 20
*/
    if (type==10 || type==20) {
        printf (\
"-----\n");
        printf ("%6.4f , %6.4f , %6.4f | %6.4f , %6.4f , %6.4f\n", \
                Err_Max_x, Err_Rms_x, Err_Bias_x, \
                Err_Max_y, Err_Rms_y, Err_Bias_y );
        printf ("Locale selection errors: %i", Select_Errors);
    }
/*
* Type 100
*/
    if (type==100) {
        Print_Histo(Err_Rms_x);
    }
/*
* Type 900 not implemented
*/
    return;
}
/* *****
* Get_Params                                D. Havelock                09.11.88
*
* The main routine may be invoked with parameters on the command line,
* alternatively, the parameters may be provided interactively following
* a prompt.
*
* Calling Arguments for MAIN:
* Calling Arguments for MAIN:
*     argv[0] points to task name in invocation
*     argv[1] points to "Number of iterations along X and Y"
*     argv[2] points to "Tolerance in the truth Map"
*             argv[3] points to "Truth-Map Size"
*     argv[4] points to "Target size"
*     argv[5] points to "Noise"
*             argv[6] points to "Number of cells in the histogram"
*             argv[7] points to "Range of histogram"
** */

int Get_Params (argc, argv, envp)
    int  argc;
    char **argv;
    char **envp;
{

```

```

extern int      Mesh_Size, Tolerance;
extern int      Map_Size, Decode_Length;
extern double   Noise_b;
extern int      Histo[], Histo_Size;
extern float    Histo_Max;

/*
 *      If needed, print out the argument list, else get params from
 *      com.line.
 */
envp=envp; /* Jus 'cause t'aint used */

if (argc < 7) {
    printf("\nInput the following:\n
        \n Mesh_Size, Tolerance, Map_Size, Target_Width, Noise,\n
        \n No. of Cells in Histogram, Histogram Range\n");
    scanf("%i, %i, %i, %i, %f, %i, %f",\
        &Mesh_Size, &Tolerance, &Map_Size, &Decode_Length, &Noise_b,\
        &Histo_Size, &Histo_Max);
}
else {
    Mesh_Size = atoi(argv[1]);
    Tolerance = atoi(argv[2]);
    Map_Size = atoi(argv[3]);
    Decode_Length = atoi(argv[4]);
    Noise_b = atof(argv[5]);
    Histo_Size = atoi(argv[6]);
    Histo_Max = (float) atof(argv[7]);
}

/*
 *      Condition the parameters.
 */
if ((Decode_Length != 2) > ACTIVE_PIXELS) {
    printf ("Target Width exceeds %i\n",ACTIVE_PIXELS/2);
    printf ("          reset to max.\n");
    Decode_Length = (ACTIVE_PIXELS/2)*2;
}

if (Histo_Size > HISTO_SIZE) {
    printf("Adjusting Histo_Size to %i (the max.)\n",HISTO_SIZE);
    Histo_Size = HISTO_SIZE;
}

if (Histo_Max <= (float) 0.) {
    Histo_Max = (float)(4./Decode_Length); /* Default Histo. range. */
}

printf(" \
    \n Mesh Size:      %i, \
    \n Tolerance:      %i, \
    \n Truth Map Size: %i, \
    \n Target Size:    %i, \
    \n Noise Level:     %f, \
    \n Histo_Size:      %i, \
    \n Histo_Max:       %f \n", \
    Mesh_Size, Tolerance, Map_Size, Decode_Length/2, Noise_b, \
    Histo_Size, Histo_Max);

return(0);
}
/* *****
 *      Print Histo
 *
 *      D.Havelock
 *      12.04.88
 */

```

```

*
* Print the histogram array to the printer and to a file.
** */

void Print_Histo(rms_histo)
double      rms_histo;
{
    extern int  Histo[];
    extern int  Histo_Size;
    static FILE *fphisto=NULL;
    int        index1, index2;
    time_t ltime;

    time(&ltime);
    if (fphisto== NULL) {
        if ((fphisto=fopen("dcteste.dat","a+"))==NULL) \
            printf(" *** Error opening histogram data file ***\n");
        else \
            printf(" ...Histogram data file 'dcteste.dat' opened %s\n",\
                ctime(&ltime));
    }
    printf("low bound bin (%f): %i\n",-Histo_Max,Histo[0]);
    if (fphisto!=NULL) {
        fprintf(fphisto, "This histo appended %s\n",ctime(&ltime));
        fprintf(fphisto, " Mesh Size: %i,\n
            \n Tolerance: %i, Truth Map Size: %i, rms_histo %f, \n
            \n Histo_Size: %i, Histo_Max: %f \n", \
            Mesh_Size, Tolerance, TMap_Size, rms_histo, \
            Histo_Size, Histo_Max);
        fprintf(fphisto, "low bound bin(%f): %i\n",\
            -Histo_Max, Histo[0]);
    }
    for (index1=1 ; index1<Histo_Size ; index1 += 10) {
        for (index2=0 ; index2<10 ; index2++) {
            printf("%6i ",Histo[index1+index2]);
            if(fphisto!=NULL) \
                fprintf(fphisto,"%6i ",Histo[index1+index2]);
        }
        printf("\n");
        fprintf(fphisto, "\n");
    }

    printf("high bound bin: %i\n",Histo[Histo_Size]);
    if (fphisto!=NULL) {
        fprintf(fphisto, "high bound bin: %i\n",Histo[Histo_Size]);
        fclose(fphisto);
        fphisto=NULL;
    }

    return;
}

/* *****
*   Noise                               D.Havelock          12.11.87
*
*   Generate signed noise of specified amplitude; -z <= noise < z.
*
** */

double Noise(z)
double z;

```

```

{
    extern int rand();
    double temp;

    if (z == 0.) return(0.);
    temp = z*((double)rand()/16384.-1.);
    return( temp );
}
/* *****
*   Entity_Function          Edges: temp3   D.Havelock   16.12.87
*
*   The Cos, Sin and b values are defined in lookup tables by ..._Init.
*   Dist is negative on the dark side of the edge and pos. on the bright.
*
*   V1.0temp3   To compute estimate in offset based on presumed knowledge
*   of the edge orientation, all inappropriate columns have been flagged
*   with c[i]=-1 except the one corresponding to the actual edge
*   (orientation. All other columns return an E_F value of 99.
*
** */
unsigned char Entity_Function(index, tj, tk)
    int      index, tj, tk;
{
    extern double   Edge_Level[];
    extern int      Decode_Index[][2];
    extern double   c[];
    extern double   s[];
    extern double   b[];
    double         dist;
    int            m;

    if (c[tj] == -2.) {
        m=99;
    }
    else {
        dist = Decode_Index[index][1]*c[tj] - Decode_Index[index][0]*s[tj] -
b[tk];
        for (m=0; m < EDGE_LEVELS; m++) if (dist < Edge_Level[m]) break;
    }
    if (DEBUG > 2) {
        printf ("f(%3i,%3i,%3i): %.4f   %i\n",index,tj,tk,dist,m);
    }

    return ((unsigned char)m);
}

/* *****
*   Entity_Function_Init      Edges: temp3   D.Havelock   16.12.87
*
*   The sine and cosine table for the theta values of the truth map are
*   tabulated for fast processing. The values of b are also tabulated.
*   The edge function "h(x)" is effected by the table of values Edge_Levels.
*   The K'th entry in Edge_Levels gives the orthogonal distance, from the line
*   position, where the quantized edge response changes from K to (K+1).
*
*   V1.0temp3   Only the entry corresponding to the actual theta is defined.
*
** */
int Entity_Function_Init(void)
{

```

```

extern double c[];
extern double s[];
extern double b[];
double pi4 = 3.14159265/4.;
double theta;
int tj;
int ok; /* logical flag */

if (DEBUG > 1) {
    printf ("Entered <Entity_Function_Init>...\n");
}

if (TMap_Size > TMAP_TABLES_SIZE) {
    printf ("TMap_Size too big for Entity_Function_Init \n");
    return (-1);
}
ok = 0;
for (tj=0; tj<TMap_Size; tj++) {
    if (fabs(Ar_X-(theta=(TMap_Offset_X+TMap_Scale_X*tj)))<=TMap_Scale_X/2.) {
        theta = (0.5+theta)*pi4;
        c[tj] = cos(theta);
        s[tj] = sin(theta);
        ok+=1;
    }
    else {
        c[tj] = -2.;
        s[tj] = 0.;
    }
    b[tj] = 0.5 + TMap_Offset_Y + TMap_Scale_Y * tj;
}
if (ok!=1) printf("Entity_Function_Init stinks... check it out.");
if (DEBUG > 2) {
    printf ("TMap_Size = %i \n Cos: Sin: b:\n",TMap_Size);
    for (tj=0; tj<TMap_Size; tj++) {
        printf ("%4f, %4f, %4f\n",c[tj],s[tj],b[tj]);
    }
}

return (0);
}

/* *****
* Target 2.0 D.Havelock 08.11.87
*
* The edge is encoded by object boundary coding (OBC) whereby the pixels
* which are just inside (on the dark side) of the edge are used to represent
* the edge. For given values of (x,y,z)=(theta,"b","noise level", the
* "Decode_Length" pixels defining the edge are determined.
* The sampling window for the edge is positive-centered at the origin, so
* that indices range from -N to +(N+1) when Decode_Length is even.
* Note that -pi<theta<pi will ensure that the floor function selects the
* OBC pixel within the object.
*
** */

int Target(theta, b, z)
double theta,b,z;
{
    extern double Edge_Level[];
    extern int Decode_Index[][?], Decode_Length;
    extern unsigned char Decode_Signal[];

```

```

double  nse, temp, y_cos, delta_b;
int      i, m;

if (DEBUG > 1) {
    printf ("Entered <Target(%.3f,%.3f,%.2f)>...\n",theta/(3.14159265/4.),b,z);
}
/*
 * Clear the index and signal arrays.
 */
for (i=0; i<ACTIVE_PIXELS; i++){
    Decode_Index[i][0] = -1;
    Decode_Index[i][1] = -1;
    Decode_Signal[i] = (unsigned char) 255;
}
nse = 0.;
/*
 * Condition the processing parameters.
 */
if (Decode_Length > ACTIVE_PIXELS) {
    printf ("Error: Maximum number of active pixels (%i) exceeded: %i\n",\
        ACTIVE_PIXELS, Decode_Length);
    return (-1);
}
if (Decode_Length%2 != 0) {
    printf ("ERROR: <Target>: Odd number of active pixels;%i\n",\
        Decode_Length--);
    printf ("          : Adjusted down.\n");
}

for (i=0; i<Decode_Length; i+=2) {
    temp = cos(theta);
    if (z > 0.) nse = Noise(z);

/*    Compute pixel index and value for lower boundary pixel      */
    Decode_Index[i][0] = (i/2-Decode_Length/4); /* center on origin*/
    y_cos = b + Decode_Index[i][0]*sin(theta) - nse;
    Decode_Index[i][1] = (int) floor(y_cos/temp);
    delta_b = Decode_Index[i][1]*temp - y_cos;
    for (m=0; m<EDGE_LEVELS; m++) if (delta_b < Edge_Level[m]) break;
    Decode_Signal[i] = (unsigned char) m;

/*    Compute pixel index and value for upper boundary pixel      */
    Decode_Index[i+1][0] = Decode_Index[i][0]; /* bound edge */
    Decode_Index[i+1][1] = Decode_Index[i][1] +1;
    delta_b += temp;
    for (m=0; m<EDGE_LEVELS; m++) if (delta_b < Edge_Level[m]) break;
    Decode_Signal[i+1] = (unsigned char) m;

if (DEBUG > 2) {
    printf ("b(%2i,%2i)= %.4f => s=%3i\n",\
        Decode_Index[i][0], Decode_Index[i][1], \
        delta_b-temp, Decode_Signal[i]);
    printf ("b(%2i,%2i)= %.4f => s=%3i\n",\
        Decode_Index[i+1][0], Decode_Index[i+1][1], \
        delta_b, Decode_Signal[i+1]);
}
}

return (0);
}
/*****

```



```

*   Test_Decode           1.e           D.Havelock
*
*   Driving routine to test the position decoding algorithm.
** */

void Test_Decode()
{
    extern unsigned char Decode_Signal [];
    extern int          Decode_Index [][2], Decode_Length;
    extern int          Mesh_Size;
    extern int          Select_Errors;
    extern int          Histo[HISTO_SIZE], Histo_Size;
    extern float        Histo_Max;
    extern double       Noise_b;
    extern double       At_X, At_Y;          /* -.5 < x < .5      */
    extern double       Err_Max_x, Err_Rms_x, Err_Bias_x;
    extern double       Err_Max_y, Err_Rms_y, Err_Bias_y;
    extern double       Avg1_Max_x, Avg1_Rms_x, Avg1_Bias_x;
    extern double       Avg1_Max_y, Avg1_Rms_y, Avg1_Bias_y;
    extern float        Position_Est[2];

    int                index;
    double             at_x, at_y;          /* 0 < x < pi4      */
    double             mesh_step_x, mesh_step_y;
    double             xerror, yerror, error;
    double             pi4=3.14159265/4.;

/*
*   Initialize statistics variables.
*/
    for (index=0 ; index <= Histo_Size ; Histo[index++]=0) {};
    Avg1_Max_x = -1.;
    Avg1_Max_y = -1.;
    Avg1_Rms_x = 0.;
    Avg1_Rms_y = 0.;
    Avg1_Bias_x = 0.;
    Avg1_Bias_y = 0.;
    Err_Max_x = -1.;
    Err_Max_y = -1.;
    Err_Rms_x = 0.;
    Err_Rms_y = 0.;
    Err_Bias_x = 0.;
    Err_Bias_y = 0.;
    Select_Errors= 0 ;

    mesh_step_x = pi4/Mesh_Size;
    Report(0);

    for (at_x = 0.; at_x < pi4; at_x += mesh_step_x) {
        mesh_step_y = cos(at_x)/(Mesh_Size*10);
        for (at_y = 0.; at_y < cos(at_x); at_y += mesh_step_y) {
            At_X = (at_x/pi4 -0.5);
            Target(at_x, at_y, Noise_b);          /* NOTE: NO NOISE IN TEST. */
            if (Decode ()) {
                Turn_Video_Off();
                printf("Decode failed... error values will be biased.\n");
            }
            Total_Calls += (double)XXX2; /* sum all calls to entity function */
        }
    }

/*
*   Gather interim error statistics
*/

```

```

        error = (at_x/pi4-0.5) - Position_Est[0];
        xerror = fabs(error);
        Err_Bias_x += xerror;                /* for V1.20 ONLY */
        Err_Rms_x += error*error;
        if (xerror > Err_Max_x) Err_Max_x = xerror;

        error = (at_y-0.5) - Position_Est[1];
        yerror = fabs(error);
        Err_Bias_y += yerror;                /* for V1.21 ONLY */
        Err_Rms_y += error*error;
        if (yerror > Err_Max_y) Err_Max_y = yerror;
/*
* Check that error is within selected locale bounds.
*/
        if ( xerror > (TMap_Max_J-TMap_Min_J+1)*TMap_Scale_X ||
            ( yerror > (TMap_Max_K-TMap_Min_K+1)*TMap_Scale_Y) ) {
            Select_Errors++;
        }
/*
* adjust histogram of errors.
*/
        if(xerror > Histo_Max) xerror=Histo_Max;
        Histo[ (int) (Histo_Size * (xerror/Histo_Max)) ] += 1;

    } /* End inner loop on mesh scan (at_y) */
/*
* report Interim results.
*/
    Avg1_Rms_x += Err_Rms_x;
    Avg1_Bias_x += Err_Bias_x;
    if (Err_Max_x > Avg1_Max_x) Avg1_Max_x = Err_Max_x;

    Avg1_Rms_y += Err_Rms_y;
    Avg1_Bias_y += Err_Bias_y;
    if (Err_Max_y > Avg1_Max_y) Avg1_Max_y = Err_Max_y;

    Err_Rms_x = sqrt(Err_Rms_x/(Mesh_Size*10));
    Err_Rms_y = sqrt(Err_Rms_y/(Mesh_Size*10));
    Err_Bias_x /= (Mesh_Size*10);
    Err_Bias_y /= (Mesh_Size*10);
    Total_Calls /= Mesh_Size*10;

    Report (1);

    Total_Calls = 0.;
    Err_Rms_x =0.;
    Err_Rms_y =0.;
    Err_Bias_x =0.;
    Err_Bias_y =0.;
    Err_Max_x =-1.;
    Err_Max_y =-1.;
} /* End outer loop on mesh scan (at_x) */

Err_Rms_x = sqrt(Avg1_Rms_x/10.)/Mesh_Size;
Err_Rms_y = sqrt(Avg1_Rms_y/10.)/Mesh_Size;
Err_Bias_x = Avg1_Bias_x /(Mesh_Size*Mesh_Size*10);
Err_Bias_y = Avg1_Bias_y /(Mesh_Size*Mesh_Size*10);
Err_Max_x = Avg1_Max_x;
Err_Max_y = Avg1_Max_y;

Report(10);

```

```

    Report(100);      /* report Histogram results */
    return;
}

```

C.5 The Center-of-Mass Position Estimator

```

c file: comnoise.for
c
c purpose:
c   A straightforward center of mass calculation is used to estimate the
c   position of a Gaussian target.
c
c Modifications:
c   2.1 Additive noise is uniformly distributed on [-1,+1).
c       Output format made importable to Cricket Graph utility.
c
      REAL*8  XSUM
      INTEGER*4  ISEED
      BYTE      TAB                                !2.1
      DATA     TAB/9/                             !2.1

      OPEN (UNIT=1, FILE='COMNOISE.DAT',status='new') !2.1
      WRITE(5,*) ' CENTER OF MASS ESTIMATE FOR GAUSS. (WITH NOISE)'
      WRITE(5,*) ' Ver 2.1  10 Oct. 1988'           !2.1
      WRITE(5,*) 'ENTER Start Amplitude, Step size, and Max. A'
               !!!
      READ (5,*) A0,DA,AMAX
      WRITE(1,*) ' CENTER OF MASS ESTIMATE FOR GAUSS. (WITH NOISE)'
      WRITE(1,*) ' Ver 2.1  10 Oct. 1988'           !2.1
      WRITE(1,*) 'Start Amplitude, Step size, and Max. A',A0,DA,AMAX !2.1
      WRITE(1,*) 'A,RMS ERROR IN X:'               !2.1

      ISEED=999
      A=A0-DA
      NX=100
      NY=100
10    A=A+DA
      IF (A.GT.AMAX) STOP
      NP=1010 !increment for debugging peak at results.

c A grid with 100 points on a side, on the unit patch [-.5,.5] x [-.5,.5],
c is used. The target Gaussian is positioned at each point and its position
c estimated.
      XO=-0.5
      DX=1./NX                                !2.1
      ITERX=NX                                !2.1
      YO=-0.5
      DY=1./NY                                !2.1
      ITERY=NY                                !2.1
      XSUM=0

```

```

c The target positions are offset from the regular grid points so that
c different results are obtained in all quadrants. Radial symmetry of the
c target would otherwise make much of the simulation redundant.
  Y=Y0-DY/3.
  DO 500 ITERY=1,NY
  Y=Y+DY
  X=X0-DX/4.      !Skew from y-axis adjustment to reduce symmetry  !2.1
  DO 500 ITERX=1,NX
  X=X+DX

c Clear the storage
  XIHAT = 0.
  YIHAT = 0.
  XISUM = 0.
  YISUM = 0.

c Determine a 3 by 3 patch of pixel values for the Gaussian at (X,Y)
c and generate the position estimate in the same loop.
  DO 100 I=-1,1
  DO 100 J=-1,1
    TERM=A*EXP(-(X+I)**2-(Y+J)**2) + 2.*RAN(ISEED)-1.
    ITERM=TERM
    XIHAT = XIHAT + ITERM*I
    YIHAT = YIHAT + ITERM*J
    XISUM = XISUM + ITERM
    YISUM = YISUM + ITERM
100  CONTINUE
    IF (XISUM.EQ.0.) THEN
      XISUM = 0.
    ELSE
      XIHAT = -XIHAT/XISUM
    ENDIF

    IF (YISUM.EQ.0.) THEN
      YIHAT = 0.
    ELSE
      YIHAT = -YIHAT/YISUM
    ENDIF

    XSUM = XSUM + (XIHAT-X)**2
500  CONTINUE

  WRITE(1,1001) A,TAB,SQRT(SNGL(XSUM/(FLOAT(ITERY)*ITERX)))
1001  FORMAT(X,F8.2,A1,F8.4)
  WRITE(5,*) 'A, RMSEx: ', A, SQRT(SNGL(XSUM/(FLOAT(ITERY)*ITERX)))
  GOTO 10
END

```

C.6 The Fourier-Phase Position Estimator

```

c file: phasenoise.for
c
c purpose:
c   A Fourier phase estimation technique is applied to estimate the
c   position of a Gaussian function. The symmetric Gaussian has a real valued
c   Fourier transform when it is centered at the origin. A shift of one
c   sample value introduces a linear phase factor. A least squares estimate is

```

```

c made of the linear phase component of the Fourier transform and this
c estimate is used to obtain a position estimate for the Gaussian.
c   The target is evaluated on a 5 by 5 image patch.
c
c Modifications:
c   2.1 Additive noise is uniformly distributed on [-1,+1).
c       Output format made importable to Cricket Graph utility.
c
      REAL*8  RMSEX
      COMPLEX FSUM(5,5),XSUM,FO(5,5),CTAB(5)
      BYTE    TAB
      DATA   TAB/0/
                                     !2.1
      OPEN (UNIT=1,FILE='PHASNOISE.DAT',TYPE='NEW')
      WRITE(1,*) 'FOURIER-PHASE POSITION ESTIMATION WITH NOISE ADDED'
      OPEN (UNIT=1,FILE='PHASNOISE.DAT',TYPE='NEW')
      WRITE(6,*) 'FOURIER-PHASE POSITION ESTIMATION WITH NOISE ADDED'
      WRITE(6,*) ' Ver 2.1  10 Oct. 1988'
                                     !2.1
      WRITE(6,*) 'ENTER Start A, Step size, and Max. A'
      READ (6,*) AO,DA,AMAX
      WRITE(1,*) 'Start A, Step size, and Max. A = ',AO,DA,AMAX
      WRITE(1,*) 'A.RMS ERROR IN X:'
                                     !2.1
      A=AO-DA
10    A=A+DA
      IF(A.GT.AMAX) STOP
      TPI=2.*3.14159265/5.

C Define the Fourier transform coefficient table.
      DO 20 I=0,4
20    CTAB(I+1)=CMPLX(COS(TPI*I),SIN(TPI*I))

C Zero the storage and define centered entity DFT
      RMSEX=0.
      DO 100 M=-2,2
      DO 100 L=-2,2
      FO(L+3,M+3)=CMPLX(0.,0.)
      DO 100 I=-2,2
      DO 100 J=-2,2
      FO(L+3,M+3)=FO(L+3,M+3) + FLOAT(INT(A*EXP(-FLOAT(I**2+J**2))))*
1      CTAB(MOD(I*L+J*M+10,5)+1)
      IF(FO(L+3,M+3).EQ.CMPLX(0.,0.))
1      FO(L+3,M+3)=CMPLX(A*1.E-10,A*1.E-10)
100   CONTINUE

c Set up raster loop over unit interval. Points are skewed slightly to
c avoid redundancy between quadrants and x=y reflections.
      X0=-.5
      NX=100
      DX=1./NX
                                     !2.1
      Y0=-.5
      NY=100
      DY=1./NY
                                     !2.1
      Y=Y0-DY/3.
                                     !2.1
      DO 500 ITERY=1,NY
      Y=Y+DY
      X=X0-DX/4.
                                     !2.1
      DO 500 ITERX=1,NX
      X=X+DX

```

```

c Reset accumulators
DO 300 L=1,5
DO 300 M=1,5
300  FSUM(L,M)=CMPLX(0.,0.)
      XSUM=CMPLX(0.,0.)
      XEST=0.

c Compute the DFT
DO 400 I=-2,2
DO 400 J=-2,2
F = INT(A*EXP(-(I-X)**2-(J-Y)**2) + 2.*RAN(ISEED)-1. ) !2.1
DO 400 M=-2,2
DO 400 L=-2,2
FSUM(L+3,M+3)=FSUM(L+3,M+3) + F*CTAB(MOD(I*L+J*M+10,5)+1)
400  CONTINUE

c Compute the phase array, using FSUM for storage
DO 450 I=1,5
DO 450 J=1,5
IF (I.EQ.3) GOTO 450
FSUM(I,J)=FSUM(I,J)/FO(I,J)
450  FSUM(I,J)=CMPLX(ATAN2(AIMAG(FSUM(I,J)),REAL(FSUM(I,J)))/TPI,0.)

c Estimate the x position using least squares (linear regression).

      XEST=0.
DO 460 I=1,5
DO 460 J=1,5
IF (I.EQ.3) GOTO 460
460  XEST = XEST + FSUM(I,J)*I
      XEST = XEST/50.
      RMSERX = RMSERX + (X-XEST)**2

500  CONTINUE

WRITE(6,*) 'A,RMSEx:', A, SQRT(SNGL(RMSERX/(FLOAT(ITERX)*ITERX)))
WRITE(1,1001) A, TAB, SQRT(SNGL(RMSERX/(FLOAT(ITERX)*ITERX))) '2.1
1001  FORMAT(X,F8.2,A1,F8.4) '2.1

GOTO 10
END

```

C.7 The Exponential-Interpolation Position Estimator

```

c file: lrnoise.for
c
c purpose:
c   This position estimation algorithm performs linear interpolation
c   in the exponent of the function EXP(xx+yy). In the absence of quantization,
c   this provides the algebraic solution for the target position.
c
c Modifications:
c   2.1      Additive noise is uniformly distributed on [-1,+1).

```

```

c          Output format made importable to Cricket Graph utility.
c
      REAL*8  XSUM
      INTEGER*4  ISEED
      BYTE      TAB
      DATA      TAB/9/
                                     !2.1
      OPEN (UNIT=1, FILE='LRINOISE.DAT',status='new')
                                     !!!
      WRITE(6,*) ' LRINOISE Ver. 2.1, 10 Oct. 1987'
                                     !2.1
      WRITE(6,*) ' ENTER Start A, Step Size, and Max A'
                                     !2.1
      WRITE(1,*) ' LRINOISE Ver. 2.1, 10 Oct. 1987'
                                     !2.1
      WRITE(1,*) ' ENTER Start A, Step Size, and Max A'
                                     !2.1
      READ (5,*) AO,DA,AMAX
                                     !!!
      WRITE(1,*) ' Start A, Step size and Max A',AO,DA,AMAX
                                     !2.1
      WRITE(1,*) ' Amplitude RMSEr'
                                     !2.1

      ISEED=999
      A=AO-DA
      NX=100
      NY=100
10      IF (A.GT.AMAX) STOP
      A=A+DA
      NP=1010 !for debug purposes only

      XO=-0.5
      DX=1./NX
                                     !2.1
      YO=-0.5
      DY=1./NY
                                     !2.1
      XSUM=0

      Y=YO-DY/3.
                                     !!!
      DO 500 ITERY=1,NY
      Y=Y+DY
      X=XO-DX/4.
                                     !2.1
      DO 500 ITERX=1,NX
      X=X+DX

      VRIGHT = 1.
      VLEFT = 1.
      DO 100 J=-1,1
      VLEFT = VLEFT * (MAXO (0, INT( A*EXP(-(X-1)**2-(Y+J)**2) +
1          2.*RAN(ISEED)-1. ) ) +.5)
                                     !2.1
      VRIGHT = VRIGHT * (MAXO (0, INT( A*EXP(-(X+1)**2-(Y+J)**2) +
1          2.*RAN(ISEED)-1. ) ) +.5)
                                     !2.1
ccc
ccc      noise-free equations are commented out below.
ccc      VLEFT = VLEFT * (INT( A*EXP(-(X-1)**2-(Y+J)**2) ) + 0.5)
ccc      VRIGHT = VRIGHT * (INT( A*EXP(-(X+1)**2-(Y+J)**2) ) + 0.5)
ccc
100      CONTINUE

      XHAT = LOG(VLEFT/VRIGHT)/12.

      XSUM = XSUM + (XHAT-X)**2
c      IF (MOD(ITERX+(ITERY-1)*NY,NP) .NE. 0) GOTO 500
c      WRITE(5,*) 'X,Y,ERR:',X,Y,(X-XHAT)
500      CONTINUE

      WRITE(1,1001) A, TAB, SQRT(SNGL(XSUM/(FLOAT(ITERY)*ITERX)))
                                     !2.1
1001      FORMAT(X,F8.2,A1,F8.4)

```

```
WRITE(5,*) 'A, RMS-X:', A, SQRT(SMGL(XSUM/(FLOAT(ITERX)*ITERX)))  
GOTO 10  
END
```


Bibliography

- [Akey83] Mark L. Akey, O.R. Mitchell, and A.P. Reeves. "A moment based two-dimensional edge operator," In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 312-317, Washington D.C., June 1983.
- [AMS86] Ellen Swanson. *MATHEMATICS INTO TYPE*. American Mathematical Society, Providence, RI, USA, 1986.
- [Ander85] Timothy A. Anderson and Chul E. Kim. "Representation of digital line segments and their preimages," *Computer Vision, Graphics and Image Processing*, vol. 30, pages 279-288, 1985.
- [Anuta76] P.E. Anuta. "Spatial registration of multispectral digital imagery using fast Fourier transformation techniques," *IEEE Trans. Geoscience and Electronics*, vol. 8, pages 153-368, October 1970.
- [Ball81] D.H. Ballard. "Generalizing the Hough transform," *Pattern Recognition*, vol. 13, no. 2, pages 111-122, 1981.
- [Barna80] Steven T. Barnard and William B. Thompson. "Disparity analysis of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, pages 333-340, July 1980.
- [Barne72] D.T. Barnea and H.F. Silverman. "A class of algorithms for fast digital image registration," *IEEE Trans. on Computers*, pages 179-186, February 1972.
- [Beren87] C.A. Berenstein, L.N. Kanal, D. Lavine, and E.C. Olson. "A geometric approach to subpixel registration accuracy," *Computer Vision, Graphics and Image Processing*, vol. 40, pages 334-360, 1987.
- [Beren88] Carlos A. Berenstein and David Lavine. "On the number of digital straight line segments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, pages 880-887, November 1988.

- [Bergm82] H.C. Bergmann. "Displacement estimation based on the correlation of image segments." In *Int. Conf. on Electronic Image Processing*, 1982.
- [Berma89] Mark Berman. "Large sample bias in least squares estimators of a circular arc center and its radius," *Computer Vision, Graphics and Image Processing*, vol. 45, pages 126-128, 1989.
- [Berzi84] Valdis Berzins. "Accuracy of Laplacian edge detectors," *Computer Vision, Graphics and Image Processing*, vol. 27, pages 195-210, 1984.
- [Beyer87] Horst A. Beyer. "Some aspects of the geometric calibration of CCD-cameras," In *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 2-4 June*, pages 68-81, 1987.
- [Beyer88] Horst A. Beyer. "Linejitter and geometric calibration of CCD - cameras," In: *ISPRS XVI Congress, Kyoto, Japan, 1-10 July*, 1988.
- [Bicke77] P.J. Bickel and K.A. Doksum. MATHEMATICAL STATISTICS: BASIC'S IDEAS AND SELECTED TOPICS. *Holden-Day Series in Probability and Statistics*, Holden-Day, 1977.
- [Blach79] T.J. Blachut et al. THE ANAPLOT; HISTORY, BASIC FEATURES AND PERFORMANCE. Technical Report NRC17281, National Research Council of Canada, Ottawa, Canada, June 1979. Also in: *The Canadian Surveyor*, June 1979.
- [Blanz88] W.E. Blanz, J.L.C. Sanz, and E.B. Hinkle. "Image analysis methods for solder-ball inspection of integrated circuit manufacturing," *IEEE Trans. on Robotics and Automation*, vol. 4, pages 129-139, April 1988.
- [Blouk85] M.M. Blouke, D.L. Heidtmann, B. Corrie, M.L. Lust, and J.R. Janesick. "Large area CCD image sensors for scientific applications," *Proceedings of the SPIE*, vol. 570, Solid State Imaging Arrays, pages 82-88, 1985.
- [Bobro74] L.S. Bobrow and M.A. Arbib. DISCRETE MATHEMATICS: APPLIED ALGEBRA FOR COMPUTER AND INFORMATION SCIENCE. W.B. Saunders, 1974.
- [Boden86] J. Bodenstab. "Machine vision for electronics manufacturing," *Robotics Engineering*, pages 21-25, September 1986.
- [Bovin87] A.C. Bovin, T.S. Huang, and D.C. Munson, Jr. "The effect of median filtering on edge estimation and detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pages 181-193, March 1987.

- [Bruck87] A.M. Bruckstein. "On optimal image digitization," *IEEE Trans. on Acoustic Speech and Signal Processing*, vol. 35, pages 553-555, April 1987.
- [Casta71] F. Castanie. "Signal processing by random reference quantizing," *Signal Processing*, vol. 1, pages 27-43, 1971.
- [Chamb78] Savvas G. Chamberlain and David H. Harper. "MTF simulation including transmittance effects and experimental results of charge-coupled imagers," *IEEE Trans. on Electronic Devices*, vol. 25, pages 145-154, February 1978.
- [Chen88] Ling-Hwei Chen and Wen-Hsiang Tsai. "Moment-preserving line detection," *Pattern Recognition*, vol. 21, no. 1, pages 45-53, 1988.
- [Chou86] Huei-Lin Chou and Wen-Hsiang Tsai. "A new approach to robot location by house corners," *Pattern Recognition*, vol. 19, no. 6, pages 439-451, 1986.
- [Colem85] A. Coleman. "Robotics research," *Robotics Age, The Journal of Intelligent Machines*, vol. 7, pages 14-19, February 1985.
- [Curry86] Sean Curry, Sheldon Baumring, and J.M. Anderson. "Calibration of an array camera," *Photogrammetric Engineering and Remote Sensing*, vol. 52, pages 627-636, May 1986.
- [Dahle87] J. Dähler. "Problems in digital image acquisition with CCD cameras," In *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 2-4 June*, pages 48-59, 1987.
- [Davie87a] E.R. Davies. "The effect of noise on edge orientation computations," *Pattern Recognition Letters*, vol. 6, pages 315-322, December 1987.
- [Davie87b] E.R. Davies. "A high speed algorithm for circular object location," *Pattern Recognition Letters*, vol. 6, pages 323-333, December 1987.
- [Davie87c] E.R. Davies. "A new framework for analysing the properties of the generalised Hough transform," *Pattern Recognition Letters*, vol. 6, pages 1-7, June 1987.
- [Davis75] L.S. Davis. "A survey on edge detection techniques," *Computer Vision, Graphics and Image Processing*, vol. 4, pages 248-270, 1975.
- [Day88] Chia Day. "Robot accuracy issues and methods of improvement," *Robotics Today*, vol. 1, no. 1, pages 1-9, 1988.
- [Dijak85] J.T. Dijak. "A method for correcting geometric distortion in video cameras," In *Proc. IEEE 1985 Nat. Aerosp. and Elect. Conf.*, pages 1382-1388, NAECON, Dayton OH, May 1985.

- [Doros84] Marek Doros. "On some properties of the generation of discrete circular arcs on a square grid," *Computer Vision, Graphics and Image Processing*, vol. 28, pages 377-383, 1984.
- [Dorst84d] L. Dorst and R.P.W. Duin. "Spirograph theory: a framework for calculations on digitized straight lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pages 632-639, September 1984.
- [Dorst84s] Leo Dorst and Arnold W.M. Smeulders. "Discrete representation of straight lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pages 450-463, July 1984.
- [Dorst86a] Leo Dorst. "The accuracy of the digital representation of a straight line," In E. Bracker, R.P.W. Duin, E.S. Gelsema, C. Kamminga, J.M. Valetton, and A.M. Vossepoel, editors, *First quinquennial review 1981-1986*, pages 183-199, Dutch society for pattern recognition and image processing, 1986.
- [Dorst86s] Leo Dorst and Arnold W.M. Smeulders. "Best linear unbiased estimators for properties of digitized straight lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, pages 276-282, March 1986.
- [Dorst87] Leo Dorst and Arnold W.M. Smeulders. "Length estimation for digital contours," *Computer Vision, Graphics and Image Processing*, vol. 40, pages 311-333, 1987.
- [Dvorn83] V.N. Dvornychenko. "Bounds on (deterministic) correlation functions with application to registration," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, pages 206-213, March 1983.
- [Ehler82] M. Ehlers. "Increase in correlation accuracy of remote sensing imagery by digital filtering," *Photogrammetric Engineering and Remote Sensing*, vol. 48, no. 3, pages 415-420, 1982.
- [ElHak86] S.F. El-Hakim. "Real-time image metrology with CCD cameras," *Photogrammetric Engineering and Remote Sensing*, vol. 11, no. 52, pages 1757-1766, 1986.
- [ElHak88] El-Hakim. "Limiting factors of real-time image metrology," In *ISPRS XVI Congress, Kyoto, Japan, 1-10 July, 1988*.
- [Elias86] L.L. Eliason and J.K. West. "Vision-based robotics for process management," *Robotics Engineering*, pages 12-14, September 1986.

- [Fairc81] Fairchild. **CCD SOLID STATE IMAGING TECHNOLOGY**. Parat, Munich, Fairchild CCD Imaging, 4001 Miranda Ave., Palo Alto, Ca., 1981. 159 pages.
- [Finke82] L. Finkelstein. "Theory and philosophy of measurement," **HANDBOOK OF MEASUREMENT SCIENCE**, chapter 1, pages 1-30, John Wiley and Sons Ltd., 1982.
- [Finke84] L. Finkelstein and M.S. Leaning. "A review of the fundamental concepts of measurement," *Measurement*, vol. 2, pages 25-34, January 1984.
- [Ford85] G.E. Ford and C.I. Zanelli. "Analysis and quantification of errors in the geometric correction of satellite images," *Photogrammetric Engineering and Remote Sensing*, vol. 51, pages 1725-1734, November 1985.
- [Forst82] W. Förstner. "On the geometric precision of digital correlation," In *Proc. ISPRS Comm. III Symposium*, pages 176-189, Helsinki. June 1982.
- [Forst85] W. Förstner. "Prinzip und leistungsfähigkeit der korrelation und zuordnung digitaler bilder [principles and performance of correlation and matching of digital images]," In *Proceeding of the 40th Photogrammetric week*, Stuttgart, 1985. (Unpublished translation by Hartmut Ziemann).
- [Forst87] W. Förstner. "Reliability analysis of parametric estimation in linear models with applications to mensuration problems in computer vision," *Computer Vision, Graphics and Image Processing*, vol. 40, pages 273-310, 1987.
- [Freem61] Herbert Freeman. "On the encoding of arbitrary geometric configurations," *IRE Trans. on Electronic Computers*, vol. EC-10, pages 260-268, June 1961.
- [Freem69] H. Freeman and J.M. Glass. "On the quantization of line drawing data," *IEEE Trans. on Systems Science and Cybernetics*, vol. 5, January 1969.
- [Gagli76] R.M. Gagliardi and S. Karp. **OPTICAL COMMUNICATIONS**. John Wiley and Sons, 1976.
- [Geary81] J.C. Geary and S.M. Kent. "Imaging characteristics of the RCA 512x320 charge-coupled device (CCD)," *Proceedings of the SPIE*, vol. 290. 1981.
- [Gonsa76] R.A. Gonsalves. "Cramer-rao bounds on mensuration errors," *Applied Optics*, vol. 15, pages 1270-1275, May 1976.
- [Grieb86] U. Griebel and R.A. Higger. "Analyzing robot motion performance," *International Journal of Robotics and Automation*, vol. 1, no. 3, pages 71-76, 1986.

- [Grunb86] Branko Grünbaum and G.C. Shepard. *TILINGS AND PATTERNS*. W.H. Freeman and Co., New York, 1986.
- [Hall79] Ernest L. Hall. *COMPUTER IMAGE PROCESSING AND RECOGNITION*. *Computer Science and Applied Mathematics Series*, Academic Press, 1979.
- [Hamer81] James R. Hamerly and Rober M. Springer. "Raggedness of edges," *Journal of the Optical Society of America*, vol. 71, pages 182-196, March ,1981.
- [Haral86] R.M. Haralick. "Computer vision theory: the lack thereof," *Computer Vision, Graphics and Image Processing*, vol. 36, pages 372-386, 1986.
- [Haral87] R.M. Haralick, S.R. Sternberg, and X. Zhuang. "Image analysis using mathematical morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pages 532-550, July 1987.
- [Havel89] D.I. Havelock. "Geometric precision in noise-free digital images," to be published in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 9, September 1989.
- [HBIIE] R.J. Farrell and J.M. Booth. *DESIGN HANDBOOK FOR IMAGE INTERPRETATION EQUIPMENT*. Boeing Aerospace Co., Seattle, Wa, February 1984.
- [HBMS] P.H. Sydenham, editor. *HANDBOOK OF MEASUREMENT SCIENCE*. Volume 1, John Wiley and Sons, 1982.
- [HBSPSE] Jr. Woodlief Thomas, editor. *SPSE HANDBOOK OF PHOTOGRAPHIC SCIENCE AND ENGINEERING*. *A Wiley-Interscience Publication*, John-Wiley and Sons, 1973.
- [Hecht79] Eugene Hecht and Alfred Zajac. *OPTICS*. *Addison-Wesley series in Physics*, Addison-Wesley, 1979.
- [Hill80] J. W. Hill. *Dimensional measurement for quantized images*. SRI Project 8487, pages 75-105, SRI, November 1980. In: *Machine Intelligence Research Applied to Industrial Automation*, NSF/RA-800426, Tenth report to the National Science Foundation, Washington. NTSI:PB81-180085.
- [Ho83] Chih-Shing Ho. "Precision of digital vision systems," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, pages 593-601, November 1983.
- [Hough62] P.V.C. Hough. "Method and means for recognizing complex patterns," U.S. Patent 3069654, 1962.

- [Hughe85] R.C. Hughes. *Accuracy of centroid determination for high contrast image with vidicon and solid state array cameras*. Lab. Memo. ST-371, National Aeronautical Establishment, Structures and Materials Lab., Ottawa, Canada, December 1985.
- [Illin87] J. Illingworth and J. Kittler. "The adaptive Hough transform," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pages 690-698, September 1987.
- [James77] T.H. James, editor. *THE THEORY OF THE PHOTOGRAPHIC PROCESS*. Macmillan and Collier Macmillan Co., fourth edition, 1977.
- [Janes87] J.R. Janesick, T. Elliott, S. Collins, M.M. Blouke, and J. Freeman. "Scientific charge-coupled devices," *Optical Engineering*, vol. 26, pages 692-714, 1987.
- [Jenki50] Francis A. Jenkins and Harvey E. White. *FUNDAMENTALS OF OPTICS*. McGraw-Hill, 1950.
- [Jenki87] Tudor E. Jenkins. *OPTICAL SENSING TECHNIQUES AND SIGNAL PROCESSING*. Prentice-Hall International, 1987.
- [Kanal85] Lavine N. Kanal, Carlos A. Berenstein, and D. Lavine. "Error analysis of subpixel registration," In *Digest - 1985 International Geoscience and Remote Sensing Symposium, (IGARSS '85)*, Amherst MA, October 1985.
- [Katze62] Jacob Katzenelson. "On errors introduced by combined sampling and quantization," *IRE Trans. Automatic Control*, pages 58-68, April 1962.
- [Klaas75] H. Klaasman. "Some aspects of the accuracy of the approximated position of a straight line on a square grid," *Computer Vision, Graphics and Image Processing*, vol. 4, pages 225-235, 1975.
- [Klett85] Reinhard Klette. "The m-dimensional grid point space," *Computer Vision, Graphics and Image Processing*, vol. 30, no. 1, pages 1-12, 1985.
- [Koend84] A.J. van Doorn, W.A. van de Grind, and J.J. Koenderink, editors. *LIMITS IN PERCEPTION*, page 536. VNU Science press BV, Utrecht, The Netherlands. 1984.
- [Koplo81] J. Koplowitz. "On the performance of chain codes for the quantization of line drawings," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 3, March 1981.

- [Koplo87] J. Koplowitz and Sundar Raj A.P. "A robust filtering algorithm for subpixel reconstruction of chain coded line drawings," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pages 451-457, May 1987.
- [Koson88] W.F. Kosonocky. "Advances in solid-state image sensors," Course Notes T33, SPIE O-E LASE 88, Los Angeles, Ca., 10-17 January 1988.
- [Krant71] D.H Krantz, R.D. Luce, P. Suppes, and A. Tversky. FOUNDATIONS OF MEASUREMENT: ADDITIVE AND POLYNOMIAL REPRESENTATIONS. Volume I, Academic Press, New York, 1971.
- [Kratk71] V. Kratky and M.C. van Wijk. "Photogrammetry used to determine the motion of a vehicle crashing into a highway barrier," In *Bulletin No.42 de la Société Française de Photogrammétrie*, pages 23-34, April 1971.
- [Kugli75] C.D. Kuglin and D.C. Hines. "The phase correlation image alignment method," In *Proc. of IEEE International Conference On Cybernetics and Society*, pages 163-165, 1975.
- [Landa87] U.M Landau. "Estimation of a circular arc center and its radius," *Computer Vision, Graphics and Image Processing*, vol. 38, pages 317-326, 1987.
- [Lawle85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. THE TRAVELING SALESMAN PROBLEM. *Wiley-Interscience Series*, John Wiley and Sons, 1985.
- [Leani84] M.S. Leaning. "Workshop on fundamental logical concepts of measurement," *Measurement*, vol. 2, pages 23-24, January 1984.
- [Lenz87] Reimar Lenz. "Lens distortion corrected CCD-camera calibration with coplanar calibration points for real-time 3-D measurements," In *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 2-4 June*, pages 60-67, 1987.
- [Lenz88a] Reimar Lenz. "On the accuracy of videometry," In *ISPRS XVI Congress, Kyoto, Japan, 1-10 July*, 1988.
- [Lenz88b] R.K. Lenz and R.Y. Tsai. "Technique for calibration of the scale factor and image center for high accuracy 3-D machine vision metrology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, pages 713-719, September 1988.

- [Luhma87] T. Luhmann. "On geometric calibration of digitized video images of CCD arrays," In *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 2-4 June, 1987*.
- [Marr82] David Marr. *VISION*. W.H. Freeman, 1982.
- [McGil77] Clare D. McGillem and Martin Svedlow. "Optimum filter for minimization of image registration error variance," *IEEE Trans. Geoscience and Electronics*, vol. 15, no. 4, page October, 1977.
- [Medio85] Gerard Medioni and Ramakant Nevatia. "Segment-based stereo matching," *Computer Vision, Graphics and Image Processing*, vol. 31, pages 2-18, 1985.
- [Merli75] P.M. Merlin and D.J. Farber. "A parallel mechanism for detecting curves in pictures," *IEEE Trans. Comput.*, vol. 24, pages 96-98, 1975.
- [Mikha83] Edward M. Mikhail. "Photogrammetric target location to subpixel accuracy in digital images," In *39th photogrammetric week*, ISPRS, Stuttgart, FDR, September 1983.
- [Mikha85] E.M. Mikhail and Dante B. Cantiller. "Geometric effects of digital image processing operations," In *Annual meeting of the American Society of Photogrammetry and Remote Sensing*, pages 717-724, ASPRS, WASHINGTON D.C., March 1985.
- [Mitch84] O.R. Mitchell and M.L. Akey. "Detection and sub-pixel location of objects in digitized aerial imagery," In *7th International Conference on Pattern Recognition*, vol. 1, Montreal, Canada, 30 July - 2 August, pages 411-414, Montreal, Canada, July 1984. see also, *Photogrammetria*, 39(3):63-83, September 1984.
- [Munte81] C. Munteanu. "Evaluation of the sequential similarity detection algorithm applied to binary images," *Pattern Recognition*, vol. 13, no. 2, pages 167-175, 1981.
- [Nakam84] Akira Nakamura and Kunio Aizawa. "Digital circles," *Computer Vision, Graphics and Image Processing*, vol. 26, pages 242-255, 1984.
- [Nakam85] Akira Nakamura and Kunio Aizawa. "Digital images of geometric pictures," *Computer Vision, Graphics and Image Processing*, vol. 30, pages 107-120, 1985.
- [Nalwa86] V.S. Nalwa and T.O. Binford. "On detecting edges," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, pages 699-714, November 1986.

- [Nalwa87] V.S. Nalwa. "Edge-detector resolution improvement by image interpolation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pages 446-451, May 1987.
- [Negin84] Michael Negin. "Computer vision requirements for microelectronic assembly," In *Robotics Research: the next five years and beyond*, pages MS84-502, Bethlehem, Pennsylvania, August 1984.
- [Neuho85] D.L. Neuhoff and K.G. Castor. "A rate distortion analysis of chain codes for line drawings," *IEEE Trans. on Information Theory*, vol. 31, January 1985.
- [Niels84] L. Nielsen, K.J. Astrom, and E.I. Jury. "Optimal digitization of 2-D images," *IEEE Trans. on Acoustic Speech and Signal Processing*, vol. 32, pages 1247-1249, December 1984.
- [Nishi88] H.K. Nishihara and P.A. Crossley. "Measuring photolithographic overlay accuracy and critical dimensions by correlating binarized Laplacian of Gaussian convolutions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, pages 17-30, January 1988.
- [Nyys80] D. Nyysönen. "Calibration of optical systems for line width measurement on wafers," *Proceedings of the SPIE*, vol. 221, pages 119-126, 1980.
- [OConn67] D.C. O'Connor. *Visual factors affecting the precision of coordinate measurements*. Civil Engineering Studies, Photogrammetry Series 6, University of Illinois, Urbana, IL, March 1967.
- [Oppen75] A.V. Oppenheim and R.W. Schaffer. *DIGITAL SIGNAL PROCESSING*. Prentice-Hall, 1975.
- [Papou65] A. Papoulis. *PROBABILITY, RANDOM VARIABLES AND STOCHASTIC PROCESSES*. McGraw-Hill, 1965.
- [Pfan68] J. Pfanzagl. *THEORY OF MEASUREMENT*. Physica Verlag, Wurzburg-Vienna, 1968.
- [Pratt74] W.K. Pratt. "Correlation techniques of image registration," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 10, pages 353-358, May 1974.
- [Pratt79] W.K. Pratt, editor. *IMAGE TRANSMISSION TECHNIQUES*. Academic Press, 1979.
- [Purll83] D.J. Purll. "Assessment of silicon imaging array performance," *Proceedings of the SPIE*, vol. 467, pages 122-133, 1983.

- [Quinn88] T.J. Quinn. "Precision measurement and advanced technology," July 1988. Introductory Lecture, 1988 Conference on Precision Electromagnetic Measurements (CPEM), Tsukuba, Japan. Author is with Bureau Int'l des Poids et Mesures, France.
- [Richt69] F.K. Richtmyer, E.H. Kennard, and J.N. Cooper. **INTRODUCTION TO MODERN PHYSICS**. McGraw-Hill, sixth edition, 1969.
- [Rober62] L.G. Roberts. "Picture coding using pseudo-random noise," *IRE Trans. Information theory*, vol. 8, pages 145-154, February 1962.
- [Rosen77] A. Rosenfeld and G.J. Vanderbrug. "Coarse-fine template matching," *IEEE Trans. on Science, Man and Cybernetics*, pages 104-107, February 1977.
- [Rosen86] Asriel Rosenfeld, N. Yokobori, and P. Yeh. "Sub-pixel geometric correction of picture by calibration and decalibration," In *Proc. 1986 IEEE int. conference on Robotics and Automation*, 1986.
- [Rudin64] Walter Rudin. **PRINCIPLES OF MATHEMATICAL ANALYSIS**. McGraw-Hill Book Company, second edition, 1964.
- [Ryan80] T.W. Ryan, R.T. Gray, and B.R. Hunt. "Prediction of correlation errors in stereo-pair images," *Optical Engineering*, vol. 19, pages 312-322, May 1980.
- [Sakri68] D.J. Sakrison. **COMMUNICATION THEORY: TRANSMISSION OF WAVEFORMS AND DIGITAL INFORMATION**. John Wiley and Sons, 1968.
- [Shann49] C.E. Shannon. "Communication in the presence of noise," *Proc. IRE*, vol. 37, pages 10-21, January 1949.
- [Shih89] F.Y.-C Shih and O.R. Mitchell. "Threshold decomposition of gray-scale morphology into binary morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, pages 31-42, January 1989.
- [Shu88] D.B. Shu, J.F. Mancuso, and Y.N. Sun. "A line extraction method for automated SEM inspection of VLSI resist," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, pages 117-120, January 1988.
- [Skill74] Marjorie E. Skillin, Robert M. Gay, et al. **WORDS INTO TYPE**. Prentice-Hall, New Jersey, third edition, 1974.
- [Slama80] Chester C. Slama, editor. **MANUAL OF PHOTOGRAMMETRY**. American Society of Photogrammetry, Falls Church, Va., fourth edition, 1980.

- [Sluze88] Andrej Šluzek. "Using moment invariants to recognize and locate partially occluded 2D objects," *Pattern Recognition Letters*, vol. 7, pages 253-257, April 1988.
- [Smith66] Warren J. Smith. MODERN OPTICAL ENGINEERING. McGraw-Hill, 1966.
- [Sobel74] I. Sobel. "On calibrating computer controlled cameras for perceiving 3-D scenes," *Artificial Intelligence*, vol. 5, pages 185-198, 1974.
- [Srira89] R. Sriraman, J. Koplowitz, and S. Mohan. "Tree searched chain coding for subpixel reconstruction of planar curves," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, pages 95-103, January 1989.
- [Tabat84] Ali J. Tabatabai and O. Robert Mitchell. "Edge location to subpixel values in digital imagery," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pages 188-201, March 1984.
- [Tanim81] S.L. Tanimoto. "Template matching in pyramids," *Computer Vision, Graphics and Image Processing*, vol. 16, pages 356-369, 1981.
- [Therr86] Charles Therrien, Thomas G. Quatieri, and Dan E. Dudgeon. "Statistical model-based algorithms for image analysis," *Proceedings of the IEEE*, vol. 74, pages 532-551, April 1986.
- [Thoma89] S.M. Thomas and Y.T. Chan. "A simple approach for the estimation of circular arc center and its radius," *Computer Vision, Graphics and Image Processing*, vol. 45, pages 362-370, 1989.
- [Thurg82a] J.D. Thurgood and E.M. Mikhail. "Photogrammetric analysis of digital images," *International Archives for Photogrammetry*, vol. 24-III, pages 576-590, 1982.
- [Thurg82b] J.D. Thurgood and E.M. Mikhail. *Subpixel mensuration of photogrammetric targets in digital images*. Purdue University publication CE-PH-82-2, Purdue University, August 1982.
- [Tian86] Qi Tian and M.N. Huhns. "Algorithms for subpixel registration," *Computer Vision, Graphics and Image Processing*, vol. 35, pages 220-233, August 1986.
- [Unruh82] J.E. Unruh and E.M. Mikhail. "Mensuration tests using digital images," *Photogrammetric Engineering and Remote Sensing*, vol. 48, pages 1343-1349, August 1982.

- [Vanie88] Jacques Vanier. "Standards of physical measurement: the role of the National Research Council," *Physics in Canada*, vol. 44, pages 3-13, January 1988.
- [Veen81] T.M. van Veen and F.C.A. Groen. "Discretization errors in the Hough transform," *Pattern Recognition*, vol. 14, nos. 1-6, pages 137-145, 1981.
- [Vosse82] A.M. Vossepoel and A.W.M. Smeulders. "Vector code probability and metrization error in the representation of straight lines of finite length," *Computer Vision, Graphics and Image Processing*, vol. 20, pages 347-364, 1982.
- [Vosse88] George Vosselman and Wolfgang Förstner. "The precision of a digital camera," In *ISPRS XVI Congress, Kyoto, Japan, 1-10 July*, 1988.
- [Watts61] D.G. Watts. *A general theory of amplitude quantization with applications to correlation determination*. Monograph 481M, Institute of Electrical Engineers, November 1961.
- [Wells71] Mark B. Wells. *ELEMENTS OF COMBINATORIAL COMPUTING*. Pergamon Press, 1971.
- [Widrow65] Bernard Widrow. "A study of rough amplitude quantization by means of nyquist sampling theory," *IRE Trans. on circuit theory*, vol. 3, pages 266-276, 1965.
- [Wong78] R.Y. Wong and Hall E.L. "Sequential heirarchical scene matching," *IEEE Trans. on Computers*, vol. 27, pages 359-366, April 1978.
- [Woods87] P.W. Woods, C.J. Taylor, D.H. Cooper, and R.N. Dixon. "The use of geometric and grey-level models for industrial inspection," *Pattern Recognition Letters*, vol. 5, pages 11-17, 1987.
- [Xiabo85] Li Xiabo and Richard C. Dubes. "The first stage in two-stage template matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 7, pages 700-708, November 1985.
- [Yang81] G.J. Yang and Thomas.S. Huang. "The effect of median filtering on edge location estimation," *Computer Vision, Graphics and Image Processing*, vol. 15, pages 224-245, 1981.
- [Ziemann82] Hartmut Ziemann. "A photogrammetric test image," In *International Archives of Photogrammetry*, vol. 24, part 1 (Commission I symposium, Canberra), pages 230-233, ISPRS, 1982. Also in: *Science Dimension*, vol. 1, no. 4, pp. 16-19, October, 1969.

- [Ziema87] Hartmut Ziemann. "Optical transfer functions as image quality measure in photogrammetry," In *Proceedings of ISPRS colloquim on Use of space photographic data for mapping*, pages 59-75, Kammer Der Technik, 4-6 September 1987.

END

14|09.9|0

FIN