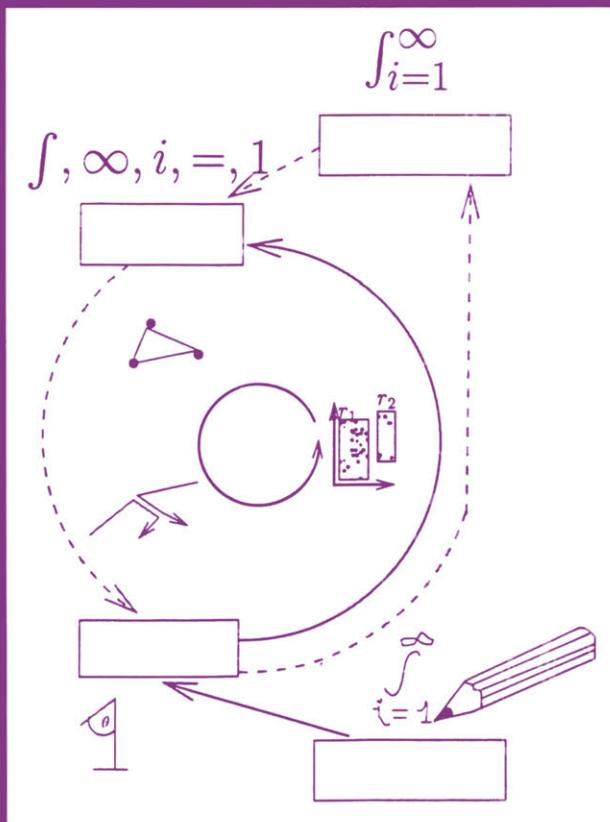


# Machine Learning and Image Interpretation



**Terry Caelli  
and  
Walter F. Bischof**

# **Machine Learning and Image Interpretation**

# **ADVANCES IN COMPUTER VISION AND MACHINE INTELLIGENCE**

Series Editor: **Martin D. Levine**  
*McGill University*  
*Montreal, Québec, Canada*

---

## **COMPUTATIONAL ANALYSIS OF VISUAL MOTION**

Amar Mitiche

## **COMPUTER VISION FOR ELECTRONICS MANUFACTURING**

L. F. Pau

## **HUMAN ENGINEERING IN STEREOSCOPIC VIEWING DEVICES**

Daniel B. Diner and Derek H. Fender

## **MACHINE LEARNING AND IMAGE INTERPRETATION**

Terry Caelli and Walter F. Bischof

## **PYRAMIDAL ARCHITECTURES FOR COMPUTER VISION**

Virginio Cantoni and Marco Ferretti

## **SEMANTIC NETWORKS FOR UNDERSTANDING SCENES**

Gerhard Sagerer and Heinrich Niemann

## **SIGMA: A Knowledge-Based Aerial Image Understanding System**

Takahashi Matsuyama and Vincent Shang-Shouq Hwang

---

A Continuation Order Plan is available for this series. A continuation order will bring delivery of each new volume immediately upon publication. Volumes are billed only upon actual shipment. For further information please contact the publisher.

# **Machine Learning and Image Interpretation**

**TERRY CAELLI and  
WALTER F. BISCHOF**

*Curtin University of Technology  
Perth, Western Australia, Australia*

**SPRINGER SCIENCE+BUSINESS MEDIA, LLC**

**Library of Congress Cataloging-in-Publication Data**

---

Caeli, Terry.

Machine learning and image interpretation / Terry Caeli and

Walter F. Bischof.

p. cm. -- (Advances in computer vision and machine  
intelligence)

Includes bibliographical references and index.

ISBN 978-1-4899-1818-5

1. Image processing--Digital techniques. 2. Optical pattern  
recognition. 3. Machine learning. I. Bischof, Walter F.

II. Title. III. Series.

TA1637.C34 1997

006.3'7--dc21

97-36803

CIP

---

ISBN 978-1-4899-1818-5  
DOI 10.1007/978-1-4899-1816-1

ISBN 978-1-4899-1816-1 (eBook)

© Springer Science+Business Media New York 1997  
Originally published by Plenum Press, New York in 1997  
Softcover reprint of the hardcover 1st edition 1997

10 9 8 7 6 5 4 3 2 1

<http://www.plenum.com>

All rights reserved

No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form  
or by any means, electronic, mechanical, photocopying, microfilming, recording, or otherwise,  
without written permission from the Publisher

## **Acknowledgments**

This work could not have been completed without the support of Curtin University of Technology, the University of Melbourne, the Australian Research Council, National Science and Engineering Research Council of Canada and, finally, but of great importance, families and friends of the research team.

# Preface

This book represents a six-year effort of the authors and their doctoral students towards developing technologies and specific systems which can interpret image data with respect to domain knowledge. Although each chapter offers a different aspect or perspective to image interpretation, there is one common theme which underpins our approach. That is, image interpretation processes which reflect how humans apply world knowledge to image data must involve perceptual learning in terms of automated knowledge acquisition (induction) and application (deduction and abduction) as well as feedback and consistency checks between encoding, feature extraction and known knowledge structures in a given application domain.

The book also represents the results of many debates, bottles of wine and, put simply, a solid community effort to build such complex systems. Finally, we wish to thank Garry Briscoe for helping with getting the book into a professional shape.

# Contents

<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxiii</b>
<b>1 An Overview and Perspective of Image Interpretation</b>	<b>1</b>
1.1 Introduction . . . . .	2
1.2 Learning Image Interpretation . . . . .	5
1.3 Image Interpretation Systems . . . . .	10
1.4 Conclusions . . . . .	15
<b>2 Fuzzy Conditional Rule Generation for the Learning and Recognition of 3D Objects from 2D Images</b>	<b>17</b>
2.1 Introduction . . . . .	18
2.2 Literature Review . . . . .	19
2.3 Input Data . . . . .	23
2.3.1 Segmentation . . . . .	25
2.3.1.1 Edge Extraction . . . . .	27
2.3.1.2 The Adaptive Segmentation Procedure . . . . .	27
2.3.2 Stereo . . . . .	28
2.4 Features and Their Attributes . . . . .	31
2.4.1 Unary Attributes . . . . .	31
2.4.1.1 Hu Invariants . . . . .	31
2.4.1.2 General Shape Attributes . . . . .	32
2.4.1.3 Arboricity Measures . . . . .	33
2.4.1.4 Fourier Descriptors . . . . .	34
2.4.1.5 Sinusoid Descriptors . . . . .	35
2.4.1.6 Hadamard Descriptors . . . . .	36
2.4.2 Binary Attributes . . . . .	37
2.5 The Fuzzy Conditional Rule Generation (FCRG) Classifier	40
2.5.1 The Learning Phase . . . . .	40
2.5.1.1 Cluster Refinement . . . . .	42
2.5.2 The Recognition Phase . . . . .	43

2.5.3	Compatibility Analysis . . . . .	46
2.5.4	Chain Permutation Constraint . . . . .	46
2.5.5	Single Classification Constraint . . . . .	46
2.5.6	Inter-chain Compatibility Analysis . . . . .	47
2.5.7	Intra-chain Compatibility Analysis . . . . .	48
2.5.8	General Comments and a New Compatibility Measure . . . . .	48
2.5.9	The Overall Recognition Algorithm . . . . .	50
2.6	Hypothesis Verification . . . . .	51
2.7	Results . . . . .	52
2.7.1	Comparison of CRG and FCRG . . . . .	52
2.7.2	Evaluating the Unary Attributes . . . . .	53
2.7.3	Depth Attribute Evaluation . . . . .	54
2.7.4	Tree Depth Experiments . . . . .	56
2.7.5	Occlusion Experiments . . . . .	58
2.7.6	Scaling Experiments . . . . .	60
2.7.7	Multiple Object Scenes . . . . .	61
2.8	Conclusion . . . . .	62
<b>3</b>	<b>Relational Evidence Theory and Interpreting Schematics</b>	<b>67</b>
3.1	Introduction . . . . .	68
3.2	Recognition by Parts . . . . .	70
3.2.1	Attribute-indexed representations . . . . .	70
3.2.2	Part-indexed representations . . . . .	70
3.3	Relational Learning . . . . .	72
3.3.1	Conditional Rule Generation . . . . .	73
3.3.2	Rulegraph Matching . . . . .	76
3.4	The Consolidated Relational Learning Algorithm (CLARET) . . . . .	78
3.4.1	Step 1: Generate Relations . . . . .	79
3.4.2	Step 2: Attribute Learning . . . . .	81
3.4.3	Step 3: Graph Matching . . . . .	83
3.4.4	Step 4: Relational Learning . . . . .	88
3.5	Relational Evidence and Hierarchical Modelling . . . . .	91
3.5.1	Conditioning over Relational Rules . . . . .	93
3.5.2	Relational Probabilities . . . . .	93
3.5.3	Observable Feature Probabilities . . . . .	94
3.6	Finite Interpretation . . . . .	94
3.6.1	Dynamic Programming . . . . .	95
3.6.2	Comparison with other Algorithms . . . . .	97
3.7	Schematic Interpretation . . . . .	98
3.7.1	Operational Schema . . . . .	101
3.8	Performance Comparison . . . . .	106
3.8.1	Scientific Symbols . . . . .	106

3.8.2	Interpreting Chemical Symbols . . . . .	112
3.9	Conclusion . . . . .	116
<b>4</b>	<b>Cite—Scene Understanding and Object Recognition</b>	<b>119</b>
4.1	Recent Systems and Proposed Theory . . . . .	120
4.1.1	Proposed Theory . . . . .	122
4.2	World Knowledge . . . . .	124
4.2.1	Knowledge Base Structure . . . . .	124
4.3	Interpretation Structures . . . . .	127
4.3.1	Visual Interpretation . . . . .	127
4.3.2	Scene Interpretation . . . . .	128
4.3.3	A Formal Definition of the VI, SI and KB Structures	131
4.4	Operational Overview . . . . .	131
4.4.1	A Simple Example . . . . .	133
4.4.2	Operator Scheduling . . . . .	135
4.4.3	Overview of Operators . . . . .	136
4.5	Learning World Knowledge . . . . .	138
4.5.1	Overview of the Learning Strategy in <i>Cite</i> . . . . .	140
4.5.2	Unary Learning Algorithms . . . . .	141
4.5.3	Binary Learning Algorithms . . . . .	147
4.5.4	Interaction of Unary and Binary Matching with Relaxation Labelling . . . . .	149
4.6	Hypothesis Generation . . . . .	151
4.6.1	Hypothesis Types and Storage . . . . .	151
4.6.2	Hypothesis Generation Procedures . . . . .	154
4.6.3	Generating VI Nodes from Image Data . . . . .	154
4.6.4	Generating VI Nodes from the VI Graph . . . . .	155
4.6.5	Generating VI Nodes from the SI Graph . . . . .	155
4.6.6	Generating SI Nodes from the VI Graph . . . . .	155
4.6.7	Generating SI Nodes from the KB Graph . . . . .	156
4.6.8	Binary Hypothesis Generation by Basic Unary Matching . . . . .	156
4.6.9	Resolving Clique Membership . . . . .	156
4.6.10	Hypothesis Generation by KB Matching . . . . .	160
4.7	Relaxation Labelling with Hierarchical Constraints . . . . .	160
4.7.1	Non-Hierarchical Relaxation Labelling . . . . .	160
4.7.2	Relaxation Labelling in <i>Cite</i> . . . . .	161
4.8	Knowledge Driven Segmentation . . . . .	164
4.8.1	The Segmentation Cycle . . . . .	165
4.8.2	Evidence of the Power of Re-segmentation . . . . .	167
4.8.3	Segmentation Algorithms . . . . .	167
4.9	Feature Extraction . . . . .	169
4.9.1	Feature Storage and Calculation . . . . .	169
4.9.2	Unary Features . . . . .	170

4.9.3	Binary Features . . . . .	171
4.9.4	Feature Invariances . . . . .	172
4.10	System Performance and Results . . . . .	173
4.10.1	Views of Street Scenes . . . . .	173
4.10.2	Views of Office Objects . . . . .	176
4.10.3	Aerial Views of Airports . . . . .	179
4.11	System Operation . . . . .	182
4.12	Conclusion . . . . .	186
<b>5</b>	<b>See<sup>++</sup>: An Object Oriented Theory of Task Specific Vision</b>	<b>189</b>
5.1	Introduction . . . . .	190
5.1.1	Teleoperation Scenario . . . . .	191
5.2	See <sup>++</sup> Theory of Vision . . . . .	191
5.2.1	What is Knowledge? . . . . .	191
5.2.2	Vision Paradigms . . . . .	192
5.2.3	Low-level Vision and Feedback . . . . .	193
5.2.4	Phenomena and Noumena . . . . .	194
5.2.5	Image as Database . . . . .	195
5.2.6	Knowledge Representation and Acquisition . . . . .	196
5.2.6.1	Procedural vs. Declarative . . . . .	196
5.2.6.2	Discrimination vs. Modelling . . . . .	196
5.2.6.3	Top-down vs. Bottom-up . . . . .	197
5.2.6.4	2D vs. 3D Object Recognition . . . . .	197
5.2.6.5	Machine Learning . . . . .	197
5.3	System Architecture . . . . .	198
5.3.1	Information Flow . . . . .	198
5.3.1.1	Image Query Language (IQL) Subsystem	198
5.3.1.2	Knowledge Base (KB) Subsystem . . . . .	198
5.3.1.3	Feedback . . . . .	199
5.3.1.4	Machine Learning (ML) Subsystem . . . . .	200
5.3.2	Software Architecture . . . . .	200
5.3.2.1	Image Query Language . . . . .	202
5.3.2.2	Knowledge Base . . . . .	202
5.3.2.3	Machine Learning . . . . .	202
5.3.2.4	Embedding See <sup>++</sup> within Systems . . . . .	202
5.4	Image Query Language . . . . .	202
5.4.1	Justification . . . . .	203
5.4.2	The IQL Query Class Hierarchy . . . . .	203
5.4.3	IQL Client and Server . . . . .	204
5.4.4	IQL Tokens . . . . .	204
5.4.5	IQL in Action . . . . .	204
5.4.6	An Example - Colour Space Segmentation . . . . .	205
5.5	Knowledge Base . . . . .	207

5.5.1	Justification . . . . .	208
5.5.2	Interactions . . . . .	209
5.5.3	The Language of Knowledge . . . . .	210
5.5.4	Knowledge Based Resegmentation . . . . .	211
5.6	Machine Learning . . . . .	212
5.6.1	Justification . . . . .	212
5.6.2	Machine Learning in Object Recognition . . . . .	213
5.6.3	Entropy Based Techniques . . . . .	213
5.6.4	The IGLUE Algorithm . . . . .	215
5.6.4.1	Bounded Storage Requirements . . . . .	216
5.6.4.2	Modified Entropy Measure . . . . .	217
5.6.4.3	Decision Graphs . . . . .	218
5.6.4.4	Stochastic Functions . . . . .	219
5.6.5	Relational Learning in See <sup>++</sup> . . . . .	220
5.7	See <sup>++</sup> in Action . . . . .	221
5.7.1	Implementing the Generic Application . . . . .	221
5.7.1.1	Current Interpretation . . . . .	221
5.7.1.2	Knowledge Base . . . . .	222
5.7.2	Results . . . . .	223
5.7.2.1	Special Considerations . . . . .	224
5.7.3	Future Directions . . . . .	224
<b>6</b>	<b>SOO-PIN: Picture Interpretation Networks</b>	<b>225</b>
6.1	Introduction . . . . .	226
6.2	The SOO-PIN System . . . . .	229
6.2.1	The SOO-PIN Architecture . . . . .	230
6.2.1.1	The Concept-Frame Structure . . . . .	233
6.2.2	Handling Uncertainty . . . . .	234
6.2.2.1	Belief and Vision . . . . .	236
6.2.2.2	Implementation of Belief Pairs . . . . .	236
6.2.3	Dynamic Image Sequences . . . . .	237
6.2.3.1	Situatedness and Intentionality . . . . .	238
6.2.3.2	A Dynamic Symbolic Interpreter . . . . .	238
6.3	Interpretation of Traffic Scenes . . . . .	239
6.3.1	Primitive Concept-Frames . . . . .	241
6.3.2	Turn Concept-Frames . . . . .	242
6.3.3	Give-Way Concept-Frames . . . . .	242
6.3.4	Low-level Processing: Extracting Attributes . . . . .	243
6.3.5	Computing Velocities . . . . .	243
6.3.5.1	Using Velocities . . . . .	245
6.3.6	Results . . . . .	247
6.3.7	Summary . . . . .	249
6.4	Interpretation of Traffic Sequences . . . . .	250
6.4.1	Dynamic Network Implementation and Results . . . . .	252

6.5	Conclusion . . . . .	254
<b>7</b>	<b>Invariance Signatures for Two-Dimensional Contours</b>	<b>255</b>
7.1	Introduction . . . . .	256
7.1.1	Methods for Invariant Pattern Recognition . . . . .	256
7.1.1.1	Integral Transforms . . . . .	257
7.1.1.2	Moments . . . . .	257
7.1.1.3	Cross-Correlation . . . . .	257
7.1.1.4	Parts and Relationships . . . . .	257
7.1.1.5	Contour-Based Methods . . . . .	258
7.2	Lie Transformation Groups and Invariance . . . . .	260
7.2.1	Definition of a Group . . . . .	260
7.2.2	One Parameter Lie Groups in Two Dimensions . . . . .	261
7.2.3	From Infinitesimal to Finite Transformations . . . . .	261
7.2.4	Derivation of the Rotation Transformation . . . . .	262
7.2.5	Functions Invariant under Lie Transformations . . . . .	264
7.2.6	From Local Invariance Measures to Global Invariance . . . . .	265
7.2.7	The Local Measure of Consistency . . . . .	265
7.2.8	The Invariance Measure Density Function . . . . .	266
7.2.8.1	Invariance Measure Densities for Specific Contours . . . . .	269
7.2.9	Invariance Space: Combining Invariance Measure Densities . . . . .	271
7.2.9.1	Vector Fields Corresponding to Rotation, Dilation and Translation Groups . . . . .	272
7.2.9.2	Uniqueness . . . . .	273
7.2.10	Discrete Invariance Signatures . . . . .	273
7.3	The Invariance Signature Neural Network Classifier . . . . .	274
7.3.1	Lie Vector Field Generation . . . . .	276
7.3.1.1	Calculation of a Centroid Image . . . . .	276
7.3.1.2	Gating of Vector Fields . . . . .	278
7.3.2	Local Orientation Extraction . . . . .	279
7.3.2.1	A Tangent Estimate . . . . .	279
7.3.2.2	Discontinuities . . . . .	280
7.3.2.3	Training a Neural Orientation Extraction Module . . . . .	281
7.3.3	Calculation of the Local Measure of Consistency . . . . .	282
7.3.4	Calculation of the Invariance Signature . . . . .	283
7.4	Character Recognition with Invariance Signature Networks . . . . .	285
7.4.1	Perfect Data . . . . .	286
7.4.1.1	Departures from Exact Invariance . . . . .	286
7.4.1.2	The Data Set . . . . .	286
7.4.1.3	Selected Networks Applied to this Problem . . . . .	287
7.4.1.4	Reduction of Data Dimensionality . . . . .	287

7.4.1.5	Perfect and Estimated Local Orientation	289
7.4.2	Optical Character Recognition . . . . .	294
7.4.2.1	The Data Set . . . . .	296
7.4.2.2	Selected Networks Employed for this Problem . . . . .	302
7.4.2.3	Results for Traditional Neural Networks . .	302
7.4.2.4	Results for Invariance Signature Neural Network Classifiers . . . . .	302
7.5	Conclusion . . . . .	305
<b>8</b>	<b>ABC: Biologically Motivated Image Understanding</b>	<b>309</b>
8.1	Basic Physiology and Anatomy of the Visual System . . . . .	310
8.1.1	Eye Movements . . . . .	314
8.2	Current Theories of Visual Recognition . . . . .	315
8.2.1	Template Matching . . . . .	317
8.2.2	Feature Models . . . . .	318
8.2.3	Fourier Models . . . . .	319
8.2.4	Structural Models . . . . .	320
8.3	The Computational Theory of Marr . . . . .	321
8.4	Overall Problems with Existing Models as Related to Biological Processes . . . . .	323
8.5	Biological Considerations . . . . .	327
8.5.1	Hierarchical Processing & Priming Effects . . . . .	328
8.5.2	Adaptation to Discordant Stimulation . . . . .	328
8.5.3	Invariance . . . . .	330
8.5.4	Saccades, Fixation & Spatial Fusion . . . . .	330
8.5.5	The World as Outside Memory . . . . .	333
8.5.6	Semantic Integration . . . . .	336
8.5.7	Global Influences on Perception . . . . .	337
8.5.8	Neuroanatomy & Neurophysiology . . . . .	340
8.6	Evolutionary Considerations . . . . .	341
8.7	Scanpaths . . . . .	343
8.8	The ABC Temporal Paradigm . . . . .	346
8.8.1	ABC Temporal Learning Model . . . . .	346
8.8.2	Temporal Learning Experiments . . . . .	349
8.8.3	Temporal Experiments and Vision . . . . .	351
8.9	ABC Vision Model . . . . .	358
8.10	ABC Model Avoids Problems of Traditional Models . . . .	366
8.10.1	Recognition . . . . .	367
8.11	Further Discussion of the Model . . . . .	368
8.11.1	Relational Learning . . . . .	368
8.11.2	World Filters . . . . .	369
8.11.3	Receptive Fields in Periphery . . . . .	370
8.11.4	Treisman Figures . . . . .	371

8.11.5 Integration of Local Features . . . . .	372
8.11.6 Polysemous Figures . . . . .	373
8.11.7 Reinforcement & Learning . . . . .	373
8.12 Conclusion . . . . .	375
<b>References</b>	<b>377</b>
<b>Author Index</b>	<b>409</b>
<b>Subject Index</b>	<b>417</b>

# List of Figures

1.1	Components of an image interpretation system. . . . .	4
1.2	Attribute-space partitioning models . . . . .	8
2.1	Examples of all the objects used in the experiment . . . . .	24
2.2	All poses of the calculator object . . . . .	25
2.3	Some examples of the multiple object scenes . . . . .	26
2.4	Examples of the segmentation . . . . .	29
2.5	An example of the stereo algorithm . . . . .	30
2.6	Arboricity measurements . . . . .	34
2.7	Definition of Cumulative Curvature. . . . .	35
2.8	Extracting binary attributes from two object parts. . . . .	38
2.9	An example of two image parts being projected onto planes	39
2.10	The CRG classifier . . . . .	41
2.11	The overall learning system. . . . .	44
2.12	The overall recognition system. . . . .	51
2.13	Comparison of the performance of a number of attribute sets	54
2.14	Results of the Depth Evaluation Experiments . . . . .	55
2.15	Varying the depth of the tree—accuracy performance. . . .	57
2.16	Effect of depth of cluster tree on generalisation performance	58
2.17	Effect of depth of cluster tree on execution time (CPU time)	59
2.18	Effect of maximum depth of cluster tree on size of tree . .	59
2.19	Results of the Occlusion Tests . . . . .	60
2.20	Results of the Scaling Tests . . . . .	61
2.21	The mouse and the stapler . . . . .	62
2.22	The stapler and the sticky tape holder. . . . .	63
2.23	The teapot, iron and the model car. . . . .	63
2.24	The stapler, trophy and the teapot. . . . .	64
3.1	“Scientific Symbols” Application . . . . .	69
3.2	Isomorphism Example . . . . .	71
3.3	Conditional Rule Generation . . . . .	75
3.4	Rulegraph Matching . . . . .	77
3.5	CLARET Algorithm . . . . .	80

3.6	Binary Relations for Unknown Pattern . . . . .	81
3.7	Generating Binary Relations . . . . .	82
3.8	Attribute Learning . . . . .	84
3.9	Graph Matching . . . . .	86
3.10	Interpretations . . . . .	87
3.11	Relational Learning . . . . .	89
3.12	Search for Finite Interpretation . . . . .	90
3.13	Hierarchical Modelling . . . . .	92
3.14	Schematic Interpretation Hierarchy using CLARET . . . . .	99
3.15	Spatial Segmentation and Relational Feature Extraction . . . . .	103
3.16	Rotation, Scale and Shift Invariance . . . . .	104
3.17	A Finite Interpretation Example . . . . .	105
3.18	FOIL Output Example . . . . .	108
3.19	“Scientific Symbols” Performance . . . . .	109
3.20	Distortion and Missing Parts . . . . .	111
3.21	“Chemicals” Application . . . . .	113
3.22	“Chemicals” Performance . . . . .	114
3.23	“Chemicals” Scene Task . . . . .	115
4.1	Overview of <i>Cite</i> Architecture . . . . .	123
4.2	Knowledge Base Graph Example . . . . .	124
4.3	Illustration of Inadequacy of Direct Labelling . . . . .	129
4.4	Hierarchical Compatibility Sets via the Insertion of the SI Structure . . . . .	130
4.5	Overview of Data Structures and Operators within <i>Cite</i> . . . . .	132
4.6	Norfolk Island Pine Image and Botanist Knowledge Base . . . . .	133
4.7	Initial Segmentation and Corresponding VI Graph . . . . .	133
4.8	Example Data Graphs after the Second Segmentation . . . . .	134
4.9	Final Labelled Segmentation of the Example Image . . . . .	135
4.10	Text Description of Example Scene . . . . .	135
4.11	Two Objects Indistinguishable Without Unary and Binary Indexing . . . . .	139
4.12	UnaryBounded Learning Algorithm Flow Chart . . . . .	143
4.13	Example of a Feature Space and the Computed Binary Decision Tree . . . . .	145
4.14	Incremental Learning Rule and Least Generalisation Model . . . . .	146
4.15	Direct Implementation of Bipartite Matching . . . . .	148
4.16	Bipartite Matching Extended to Include Partial Relational Information . . . . .	149
4.17	Interaction of Unary and Binary Matching, Relaxation Labelling and Hypothesis Generation . . . . .	150
4.18	Hypothesis Notation in <i>Cite</i> . . . . .	153
4.19	Description of ProcessSIFromKB Algorithm . . . . .	157
4.20	Description of ProcessSIFromVI Algorithm . . . . .	158

---

4.21 Illustration of Variables for Clique Membership Algorithm . . . . .	159
4.22 Update Diagram for $P_{i,j}^{SK}$ . . . . .	163
4.23 Hypothesis Graph Window Illustrating Selected Hypothesis Weights . . . . .	166
4.24 Sample of Images used in Botanist Knowledge Base . . . . .	167
4.25 Storage of Unary and Binary Features in VI Nodes . . . . .	170
4.26 Two Methods for Computing Shared Boundary Ratio . . . . .	172
4.27 Simple Street Scene . . . . .	174
4.28 Text Description of Street Scene . . . . .	174
4.29 Complex Street Scene . . . . .	175
4.30 Text Description of Street Scene . . . . .	175
4.31 Text Description of Office Knowledge Base . . . . .	176
4.32 CD, Keyset, Gluestick and Drink Scene . . . . .	177
4.33 Text Description of CD, Keyset, Gluestick and Drink Scene . . . . .	177
4.34 Two Pens, Two Cups and Holepunch Scene . . . . .	178
4.35 Text Description of Pens, Cups and Holepunch Scene . . . . .	178
4.36 Full Knowledge Base used in Airport Analysis . . . . .	179
4.37 Refuelling Image and Analysis Graph . . . . .	180
4.38 Text Description of Refuelling Scene . . . . .	180
4.39 Emergency and Load Image and Analysis Graph . . . . .	181
4.40 Text Description of Emergency and Load Scene . . . . .	181
4.41 The Main Command Window in <i>Cite</i> . . . . .	182
4.42 The Image and Segmentation Window . . . . .	183
4.43 The Data Graph Window . . . . .	184
4.44 The Hypothesis Graphing Window . . . . .	185
4.45 The Operator Profile Window . . . . .	186
4.46 The Results Window . . . . .	187
5.1 Conventional and knowledge based communication . . . . .	191
5.2 Flow of data in See <sup>++</sup> . . . . .	199
5.3 Software architecture of See <sup>++</sup> . . . . .	201
5.4 IQL subsystem . . . . .	205
5.5 Colour space segmentation showing classes . . . . .	206
5.6 The segmentation process . . . . .	208
5.7 Knowledge base schematic . . . . .	210
5.8 The resegmentation process . . . . .	212
5.9 IGLUE class structure . . . . .	215
5.10 Building a decision graph . . . . .	219
5.11 Relational learning in See <sup>++</sup> . . . . .	220
5.12 Principal system displays . . . . .	222
5.13 Sequence of images from the training set . . . . .	223
5.14 Typical output from See <sup>++</sup> . . . . .	224
6.1 The SOO-PIN system concept . . . . .	231

6.2	The implemented system architecture . . . . .	232
6.3	Typical traffic scene processed by the SOO-PIN system . .	240
6.4	The traffic scenario network . . . . .	241
6.5	The support and plausibility values . . . . .	243
6.6	Example of 3 successive frames . . . . .	244
6.7	A pair of cars in successive frames . . . . .	245
6.8	Diagram showing 3 cars turning right, over 3 frames . . .	246
6.9	Collins & Exhibition Sts., frame 116 . . . . .	247
6.10	Lygon & Queensberry Sts., frame 150 . . . . .	249
6.11	Swanston & Faraday Sts., frame 316 . . . . .	250
6.12	The dynamic traffic scenario network . . . . .	251
6.13	One frame's output from Java processed velocity data . .	253
7.1	Infinitesimal transformation leading to rotation. . . . .	263
7.2	Local Measure of Consistency as a function of arc length. .	267
7.3	Square of side $2L$ . . . . .	270
7.4	Invariance Density Measure with respect to rotation for a square. . . . .	271
7.5	Example of a sampled contour and its estimated tangents.	274
7.6	20 bin discrete Invariance Signatures for the contour in Figure 6.5. . . . .	274
7.7	Invariance Signature-based contour recognition system. .	275
7.8	$\bar{x}$ module. . . . .	278
7.9	Coordinate matching module for node $x_i$ . Output is 1 when $ x_i - \bar{x}  < \theta$ . . . . .	278
7.10	Tangent estimation with varying window sizes, using the eigenvector corresponding to the largest eigenvalue of the covariance matrix. . . . .	280
7.11	Tangents estimated with a window size of $35 \times 35$ . . . . .	280
7.12	Calculation of the dot product of the tangent estimate $\theta$ and the vector field corresponding to rotation, for the image point at coordinates $(i, j)$ . . . . .	283
7.13	Neural module which calculates the absolute value of its input. . . . .	284
7.14	Neural binning module. . . . .	285
7.15	Training set of canonical examples of unambiguous characters. . . . .	287
7.16	Test set of ideally shifted, rotated and reflected letters. .	288
7.17	Test patterns classified correctly by the TNNs. . . . .	290
7.18	Classification performance and errors of TNN No. 2 during training. . . . .	291
7.19	Test Patterns Misclassified by the 5 Bin Invariance Signature Neural Network Classifiers, and the training examples as which they were incorrectly classified. . . . .	294

7.20	The characters scanned to generate the real data set. The box shows the border of an A4 page (210mm × 297mm) so that the size of the original characters may be judged. The dashed line shown the partition into training and test data	295
7.21	Examples from the training set of thinned, scanned characters.	297
7.22	Examples from the test set of thinned, scanned characters.	297
7.23	Tangents estimated for training examples of the letter “a”.	298
7.24	5 bin Invariance Signatures for training examples of the letter “a”.	299
7.25	Tangents estimated for training examples of the letter “x”.	300
7.26	5 bin Invariance Signatures for training examples of the letter “x”.	301
8.1	Subcortical and Cortical Processing in the Macaque.	312
8.2	Functional Segregation In The Primate Visual System.	313
8.3	‘Wife and Mother-in-Law’ Polysemous Image	316
8.4	Difficult Image	317
8.5	Typical Computational Vision System Structure.	322
8.6	Internal Image Representation.	332
8.7	Trans-Saccadic Fusion Experiment	334
8.8	Perceptual Span In Reading	336
8.9	Bistable Motion Perception.	338
8.10	Virtual Occlusion.	339
8.11	Feedforward And Backprojection Connections	341
8.12	Saccade Scanpaths.	345
8.13	Temporal Learning Components.	347
8.14	Actual Neuron Layout and Connections.	348
8.15	Hebbian Update of Motor Control Weights.	348
8.16	Bidirectional Link.	350
8.17	Actual FSA Learned By System—Extended Learning.	350
8.18	Simple Spatial Filters.	352
8.19	Saccade/Fixation Points.	353
8.20	Vision Network.	356
8.21	Occluded Objects.	357
8.22	Occlusion Filters.	357
8.23	Vision Network With Learned Saccades.	358
8.24	ABC Associative Layer.	362
8.25	Associative Layer And External World Filters.	363
8.26	Equivalence Of Diagrammatic Forms.	364
8.27	ABC Vision Model.	365
8.28	Vision System Compared	367
8.29	Minsky-Papert Figures.	369
8.30	Minsky-Papert Figures And Temporal Learning	369

8.31 Receptive Fields . . . . .	371
8.32 Treisman Visual Search. . . . .	372

# List of Tables

1.1	Representative machine learning techniques for computer vision . . . . .	6
1.2	Image interpretation systems. . . . .	11
4.1	Notation of Hypothesis Types . . . . .	152
4.2	Notation of Node Index Sets . . . . .	153
4.3	Botanist Knowledge Base Classification Results . . . . .	168
4.4	Summary of Unary Features . . . . .	170
4.5	Summary of Binary Features . . . . .	171
4.6	Summary of Results Presented . . . . .	173
7.1	Classification performance (% correct) of traditional neural network classifiers trained for 1000 iterations with the data shown in Figure 6.15 and tested with the “perfect” data set in Figure 6.16. . . . .	290
7.2	Performance at iteration at which best performance on test data occurred for traditional neural network classifiers trained with the data shown in Figure 6.15 and tested with the “perfect” data set in Figure 6.16. . . . .	291
7.3	Classification performance (% correct) of Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 6.15 and tested with the “perfect” data set in Figure 6.16. . . . .	292
7.4	Classification performance (% correct) of 5 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 6.15 and tested with the “perfect” data set in Figure 6.16. . . . .	293

7.5	Classification performance (% correct) of 10 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 6.15 and tested with the “perfect” data set in Figure 6.16. . . . .	293
7.6	Classification performance (% correct) of Traditional Neural Network Classifiers trained for 200 iterations with the data described in Section 6.4.2.1. . . . .	303
7.7	Classification performance (% correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 40000 iterations with the data described in Section 6.4.2.1. . . . .	303
7.8	Classification performance (% correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 10000 iterations with the data described in Section 6.4.2.1, modified to label characters which can be transformed into each other as the same character. . . . .	304
7.9	Failure analysis for Network 5 from Table 6.8. . . . .	304
8.1	Vector Values for Line Filter . . . . .	353
8.2	Vector Values for Colour Filter . . . . .	354
8.3	Recognised Concept Scores. . . . .	358
8.4	Exteroceptive Modalities and Sub-modalities. . . . .	361

# **Chapter 1**

# **An Overview and Perspective of Image Interpretation**

**Terry Caelli and Walter F. Bischof**

## **Abstract**

One ultimate aim of computer vision is to develop systems which can interpret image information in ways which are consistent with how humans bind domain knowledge with what is sensed or visualised. To attain this goal, computer systems need to encode images, extract features and associate their attributes with specific types of knowledge. This chapter sets the stage for this book which describes the results of our efforts to develop technologies which we see as necessary for the building of image interpretation systems which are adaptive and are capable of learning how to attain these goals.

## 1.1 Introduction

In general, image interpretation refers to the process of labelling image data (typically in the form of image regions or features) with respect to domain knowledge. This knowledge is either represented *explicitly* or is encoded *implicitly* in specific algorithms and constraints used by an interpretation system. An example of the former can be found in systems for the interpretation of aerial photographs containing roads and buildings where specific image features are linked to a representation of urban domain knowledge. In the latter case, the feature extraction process itself is derived directly from a physical model of the imaging process, such as in determination of edge features in aeromagnetic data corresponding to dykes. In both cases, however, image interpretation centres on the problem of how extracted image features are bound to domain knowledge.

Representations of domain knowledge depend on the the specific interpretation problem that is to be solved by an image interpretation system (IIS). In the following chapters, we present a variety of representational approaches. In chapters 2-5, we discuss representations used in outdoor and multi-object scene interpretation where objects are typically defined by parts and their relations, both being defined by lists of attribute values. Symbolic representations in the form of logics and grammars have been explored, at least theoretically, over the past decade [78, 282]. In chapter 6, we illustrate how logic programming can be used as an interpretation method in a system for the interpretation of sequences of traffic images. Finally, in chapters 7-8, we examine neural-network approaches to image interpretation with model-based neural networks and with recurrent, hierarchical self-organizing (Kohonen) maps. In both these approaches, feature extraction, learning and domain knowledge are all encoded within self-similar perceptron-like systems.

The image interpretation systems of the late 1980's, such as Schema [89], Sigma [198], and others (see chapters 2-5 for numerical and 6 for symbolic examples) were similar to the systems presented here, both with respect to the knowledge representations and with respect to the capability of producing symbolic descriptions of images. However, the systems presented here differ from those systems in that they rely extensively on general adaptive procedures and machine learning, i.e. they rely on the use of induction and deduction to generate and/or instantiate knowledge in sensed data.

More recently, work on multi-media database systems [210, 255] has refocused interests in image interpretation and annotation. These systems learn to "track" the behaviour of users of the image query process and learn to provide sets of image choices dependent on previous decisions. They share the "agent-oriented" approach of the systems presented here, where feature extraction techniques, classification techniques, and interpretation techniques are selected adaptively. The multi-media database systems are,

however, simpler in two respects. First, they work at a simpler representational level, in that they typically address locating regions, features or images defined by image characteristics (e.g., “a reddish or green textured region”) rather by domain-specific characterisations (e.g. “illegally turning cars”). They thus avoid the difficult problem of how to bind domain knowledge to image features in a non-trivial way, and how to do so in an efficient and robust way. Second, these systems are typically non-relational and thus can rely on simpler decision and interpretation processes.

## Image Interpretation Paradigm

Image Interpretation has been traditionally viewed as an assignment of symbolic labels to image data without explicit reference to the method or algorithm that is used to derive a certain labelling. In this book, we explore a paradigm of image interpretation where image descriptions are defined by the set of procedures that constitute the interpretation method. Thus, an image interpretation is linked to, and is understood in terms of a complete representation of the interpretation system, which includes the encoding and feature extraction methods, the knowledge representation and domain knowledge, and the learning sub-system. Figure 1.1 shows a general framework of the image interpretation paradigm we have explored in the systems discussed in the following chapters.

Recent work on image interpretation is by no means restricted to visual images. Given the developments in computer graphics, geographic information systems, and multi-media technologies, image data can consist of range (depth), colour, magnetic, ultrasound multi-spectral and even epidemiological data, and the “image interpretation” applies equally to all these “images”. In this book, however, we restrict our interest mainly to images which are “naturally” visible even though we have applied some of our learning and feature extraction techniques to range data as well as intensity and colour information. It should also be pointed out that the (sensed) visual input is not necessarily an optimal image code and is often best replaced by a more advanced code. For example, in scene labelling of 3D-objects based on range data, complete, view-independent surface coordinates (in Euclidean coordinates) are not appropriate for recognition since these values are not invariant to rigid motions. Recoding the input in terms of mean and Gaussian curvature would be much more appropriate in this case.

All image interpretation methods rely to some extent on image segmentation and subsequent feature extraction. Image segmentation is one of the most active areas of computer vision, and a multitude of boundary-driven and region-driven methods have been proposed. Boundary-driven methods extract features such as edges, lines, corners or curves that are typically derived via filtering models which model or regularise differen-

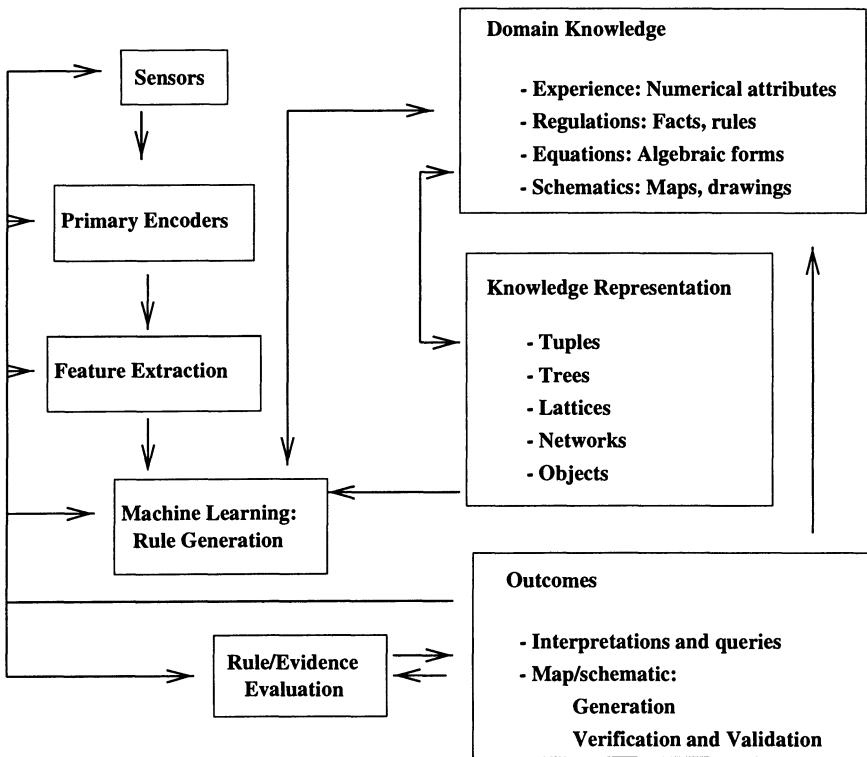


Figure 1.1: Components of an image interpretation system.

tial operators in various ways [56]. Region-based methods typically involve clustering, region-growing, or statistical models such as partial-modelling methods [58]. In chapter 2 a new technique is discussed where boundary-based and region-based methods are combined into an adaptive hierarchical feature extraction/segmentation model which partitions images into regions as a function of how these partitions can minimise the statistical variations within feature regions.

However, this book is not focused upon low-level vision processes per se. Rather, we emphasise the integration of features with domain knowledge and their update as a function of the interpretation processes. It is therefore critical to our paradigm of image interpretation that we can develop robust and adaptive feature extraction methods which can be updated via domain knowledge.

## Knowledge Representation

Our image interpretation paradigm deals with linking sensed data to domain knowledge via machine learning techniques. In turn, we use the

knowledge structures to determine, constrain and control the learning and hypothesis-instantiation processes. For this reason, our learning methods are typically “explanation-based” [43], and require a precise definition of knowledge types and interpretation architectures. The knowledge forms presented in the following chapters include

- Empirical facts and rules in the form of Horn clauses (i.e. conjunctions of predicates with no more than one negation) that are derived from observations, for example  

$$\text{House}() \Leftarrow (\text{Part } i \text{ } (\text{rectangular}()), \text{ colour-values}()) \\ \text{and } \text{Connection}(i,j) \text{ } (\text{centroid-distance}(), \text{ shared-boundary}()) \\ \text{and } \text{Part } j \text{ } (\text{parallelogram}()) \dots)$$
- Algebraic rules, especially rules from solid geometry, for example  

$$\text{Perspective-projection}(x,y) \Leftarrow f(x-r)/z \text{ and } f(y-r)/z \dots$$
- Symbolic rules , for example  

$$\text{Illegal-left-turn}() \Leftarrow \text{Turn-left}() \text{ and Red-lights}() \dots$$

Such facts and rules are encoded in data structures (architectures) such as directed, labelled and attributed graphs, trees (acyclic graphs) or lattices, networks of frames, or distributed connectionist architectures.

## 1.2 Learning Image Interpretation

The aim of machine learning is to create rules from observations, guided by the knowledge representations required for a specific system. In this section, we give a brief overview of machine learning techniques that are used in the development and optimisation of image interpretation systems. As discussed before, image interpretation refers to the process of labelling image data with respect to a set of criteria that can either be known *explicitly* or can be encoded *implicitly* in the specific algorithms and constraints used by the system. In both cases, machine learning techniques are useful for making generalisations from training data (e.g. predicting labels for newly observed image data), for estimating parameters of sub-processes (e.g. parameters controlling image segmentation), or for optimising the procedures for searching and labelling images.

Machine learning covers problems in inductive and deductive learning [43]. Induction refers to the process of generalising from known examples, while deduction refers to learning to reason about data from given models. “Explanation-based learning” combines both aspect of the learning problem. Most known applications of machine learning to computer vision involve, however, induction learning and, in the following we provide a brief overview over the more popular approaches.

A representative sample of general inductive machine learning techniques already in use in image interpretation is shown in table 1.1. The

techniques can be grouped into “supervised” and “unsupervised” learning. In supervised learning, the output states of the image interpretation system (for example, the labels to be assigned to different image parts) are known explicitly and the system is aimed at predicting these states. In unsupervised learning, the output states are defined only implicitly in the specific algorithms and built-in constraints.

## Unsupervised learning

Classic examples of models include clustering techniques where input data is grouped into sets using the proximity of example attributes [152]. The result of the grouping process depends on the characteristics of the proximity models and the parameterisation. Parametric techniques include parametric Bayesian clustering where (probabilistic) cluster membership is modelled using, for example, multivariate Gaussian functions [61]. Alternatively, the functions can be used to synthesise observed data (images), as is the case with the radial basis function formulation [259]. In both cases, clustering involves determining position and shape parameters of the best fitting functions for modelling the clusters, and the algorithms for finding them typically use gradient descent with respect to the function parameters.

unsupervised learning		
	model	major application
parametric	Bayesian classifiers radial basis functions	segmentation image synthesis
non-parametric	K-means clustering self-organizing maps vector quantisation COBWEB	feature grouping segmentation image compression visual taxonomies

supervised learning		
	model	major application
attribute-indexed	discriminant functions decision trees feed-forward neural networks evidenced-based systems	classification recognition feature extraction recognition
part-indexed	rulegraphs conditional rule generation FOIL	recognition by parts recognition by parts symbolic descriptions

Table 1.1: Representative machine learning techniques for computer vision

Parametric clustering techniques have several drawbacks, including the complexity of the search procedure required, the lack of unique solutions and, most importantly, the degree to which the parameterisation fits the data. For these reasons, non-parametric clustering techniques have been more popular. Typically, clustering is aimed at partitioning data such that within-cluster distances are minimised and between-cluster distances are maximised. The methods shown in table 1.1 share common features to achieve this goal. For the K-means method, the formulation is direct in that the search for clusters is guided by the minimax constraint (minimising within-cluster distances and maximising between-cluster distances). Kohonen maps enact a similar process [8] through the formation of attractors in the data-attribute space. In vector quantisation [74] clustering is achieved through binning techniques [102] while conceptual clustering systems like COBWEB [108] create partitions in attribute space by maximising category utility measures.

For image interpretation systems the benefits of clustering lie in their ability to summarise data which, in turn, can be used for evidencing different objects or structures. Such processes are extensively used in chapters 2-5 and 8.

## Supervised learning

Supervised learning differs from unsupervised in so far as criteria for labelling data (e.g. classification labels) are known explicitly. Given some training data described in terms of a set of features (attributes) and their class labels, the goal of supervised learning is to find a (simple) partitioning of the feature (attribute) space that allows correct classification of all training data, as well as generalisation from training data to unseen, similar data. Supervised learning methods differ with respect to the feature space partitioning, and so the types of generalisations that they produce (see Figure 1.2).

One class of learning techniques partitions the attribute space using perceptron-like linear decision functions. These include classical discriminant function methods [91], linear decision trees, decision tree and decision trees based on neural networks [244]. Elaborations of these methods involve neural network-based approaches such as cascade-correlation [99], radial basis functions [259], or vector quantisation [74]. These methods allow arbitrary splitting of attributes to result in generalisations which cannot be easily expressed as conjunctions of bounds of each attribute but are aimed at maximally evidence each class as represented in the training data (see figure 1.2).

A second class of learning techniques, sometimes referred to as automatic rule generation, also partition attribute spaces with respect to class labels but the partition boundaries are constrained to orientations parallel

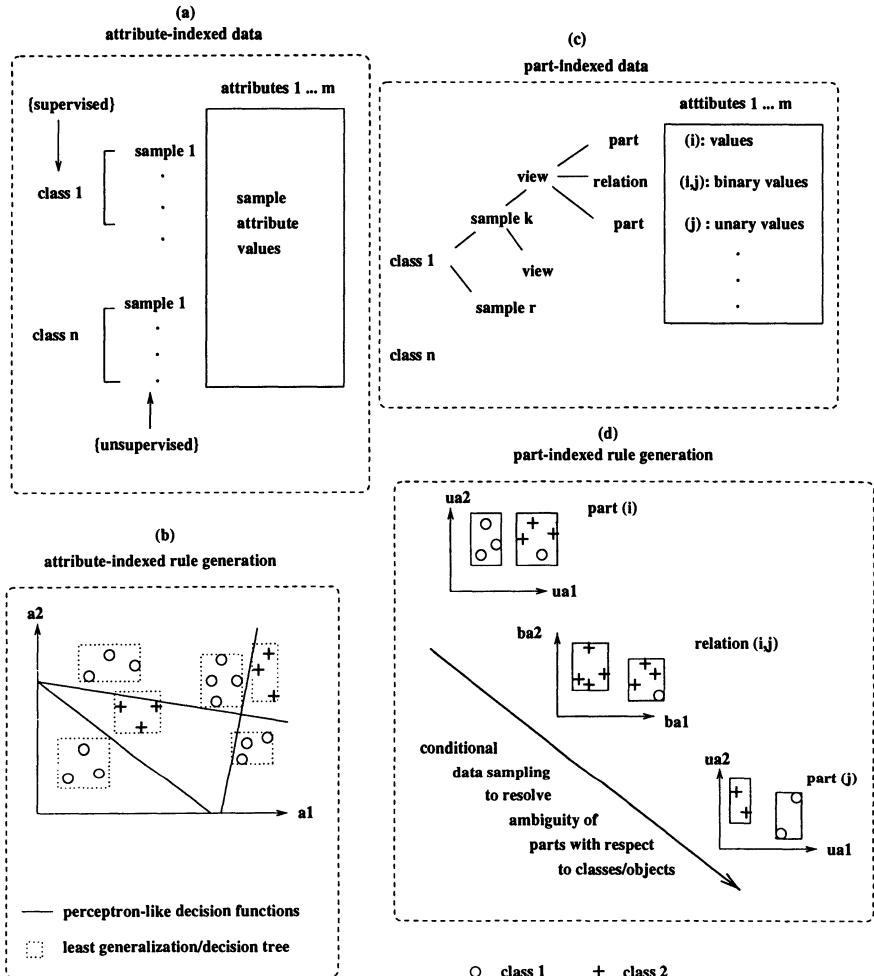


Figure 1.2: Attribute-space partitioning models. (a) attribute-indexed data structures, (b) attribute-indexed rule generation, (c) part-indexed data structures, (d) part-indexed rule generation.

to the axes. This is done to allow extraction of rules of the Horn clause form

*if attribute1 ∈ bounds1 and attribute2 ∈ bounds2 ...  
then evidence weights for each class are ...*

where the rule condition (*if*-part) is defined by a conjunction of attribute bounds and the rule actions (*then*-part) are defined in terms of fuzzy class memberships, Bayesian posterior probabilities, or as evidence weights derived from a neural network [50]. Popular rule-based methods include decision trees [270] and evidence-based systems [50, 153].

Both classes of learning techniques are only partially adequate for image interpretation systems since they cannot generate descriptions of structures with missing parts or features or descriptions of multiple objects in a scene. Their rule form is not rich enough to capture relational, hierarchical, or object-oriented nature of image descriptions. For the recognition of partial objects, i.e. a “recognition-by-parts” approach [26], we need to generate descriptions of structures like

*if part<sub>i</sub> has these attributes  
and the relation between part<sub>i</sub> and part<sub>j</sub> has these attributes  
and part<sub>j</sub> has these attributes and ...  
then part<sub>i</sub> is likely to be part<sub>x</sub> of object<sub>y</sub>,*

The two classes of rule representations are sometimes referred to as *attribute-indexed* and *part-indexed*. The attribute-indexed representation relies on a single representational domain, the attribute space, and classification rules are indexed as regions in this space. Part-indexed representations, on the other hand, rely on multiple attribute domains corresponding to unary attributes of parts and binary attributes of part relations, and classification rules are indexed by sequences of parts and their relations. In general, part-indexed representations have a greater representational power but are inferior to attribute-indexed representations with respect to the efficiency of the associated matching procedures [30].

Learning of recognition rules is heavily dependent on context. For this reason, we extended current learning methods and developed new forms of relational learning based upon the “explanation-based learning” approach. These new learning procedures are examined in detail in chapters 2 to 5. These learning methods need to address the following problems, which occur frequently in image interpretation.

*Recognition from partial data:* Many real image interpretation problems involve the recognition of patterns from partial data. We claim that in order to optimise this process, rule generation and recognition methods are required that are based on a “recognition-by-parts” approach. In this approach, pattern fragments or individual pattern parts (and their associated attributes) are learned to provide sufficient evidence for the classification of patterns.

*Recognition from distorted data:* Most image interpretation problems involve identification of known classes or patterns in data which is not identical to any of the training data. Solutions involve the ability of learning systems to generalise from training data. The problem is to optimise generalisation while retaining recognition accuracy.

*Finding patterns in complex data:* One of the more difficult problems in image interpretation is that of finding patterns (or objects) in complex multiple object scenes. Solutions must involve evidencing distinct pattern regions in an efficient way.

*Correspondence between data parts and model parts:* Beyond a mere classification of patterns or objects, it may be required to determine the correspondence between the parts of observed image data and those of stored models. This may be required to determine the pose of objects or in order to predict the existence of (hidden) pattern parts. Solutions to this problem depend on whether learning or encoding training data also includes pattern or object part labels. Part-indexed data structures implicitly solve the pose problem while simple attribute-indexed systems, such as evidence-based systems [153] do not. These methods may have to be supplemented by methods for additional hypothesis testing or model projection to solve the pose problem. This will be explored further in the following sections.

### 1.3 Image Interpretation Systems

Table 1.2 summarises the systems discussed in the following chapters in terms of their knowledge representation, feature extraction, knowledge acquisition and processing, and applications.

#### Chapter 2 - FCRG: Fuzzy Conditional Rule Generation

This chapter deals with a “recognition-by-part” approach to 3D object recognition and introduces a new technique for generating descriptions of objects in terms of parts and their relations using fuzzy clustering procedures. The method is based on the conditional rule generation (CRG) technique developed in [30]. Both methods generate rules for the identification of object parts in the following way:

1. The attribute vectors of all parts are placed in a (unary) attribute space and the attribute space is then partitioned into regions using either methods used in decision trees [270] (in the case of CRG) or using fuzzy clustering (in FCRG).
2. If a region in attribute space has only members of a single class then we know how to classify those parts, and the region bounds define the limits of generalisation (see Figure 1.2).
3. If there is more than one class represented in one region we need additional evidence to resolve class membership. In attribute-based learning (for example, decision trees and neural networks) the only solution is to split attribute regions. In CRG and FCRG we have another option. We can return to the original data and search for relations between the “unresolved” parts and other parts and use the (binary) attributes of these relations to resolve classification ambiguities. If this is still insufficient we can, in turn, consider (unary) attributes of those other parts. This expansion through chains of unary and binary attribute vectors continues until the classification for a given part is resolved.

Chapter: System	Knowledge Representation	Feature Extraction	Knowledge Acquisition and Processing	Application
2: FCRG 3: CLARET 4: CITE 5: SEE++	labelled and attributed graphs (including lattices)	pixel, texture region clustering edge and motion models	induction over attributes and relational structures	2D and 3D object recognition scene annotation
6: SOO-PIN	network of frames (objects):	edges, corners, motion	supplied by expert	reasoning about traffic.
7. MBNN	model-based neural network	feedforward and recurrent networks	supervised	invariant pattern recognition
8. ABC	self-organizing maps	hierarchical recurrent self-organizing maps	supervised and unsupervised	image understanding

Table 1.2: Image interpretation systems.

4. In both, CRG and FCRG, objects are defined by lists of rules with each rule condition describing attributes of parts and their relations. In the case of CRG, the rule conditions are defined by regions in unary and binary attribute spaces with crisp boundaries, whereas in FCRG fuzzy membership functions are used.

The FCRG system also introduces novel methods for combining evidence from part labels to result in full object recognition and scene labelling, and it is applied to single and multi-object scenes.

## Chapter 3 - CLARET: Relational Evidence Theory

This chapter discusses the foundations of relational structure learning using both Bayesian and evidence theory formulations. This theory is concerned with the problem of using machine learning to generate descriptions of image structures in terms of attributed, labelled parts and their relations. Current numerical learning methods do not allow for attribute induction (generalisations) over relational structures. CLARET, on the other hand, is a method that generates relational descriptions and minimum length rules which are sufficient to match observed relational structures with known ones. We show that the complexity of matching is on the order of the number of classification rules rather than the number of data points, which implies that matching can be very efficient. CLARET relies on a much more sophisticated rule generation method than FCRG (and CRG). During learning, CLARET checks for the uniqueness of the current attribute resolution in terms of a “rulegraph”, the current graph of bounded attributes connected by compatible labels, before expanding to add additional relational properties or splitting attributes.

We apply this theory to the problem of interpreting hand-drawn schematics. Of particular interest is the incorporation of hierarchical description of spatio-temporal contours (time-indexed drawings) into CLARET which results in rules relating characteristics of contours at different levels of resolution. We also compare this approach to recent symbolic relational rule generation models, such as FOIL [271]. The comparison not only indicates the superiority of this approach but also demonstrates its superior ability for determining minimum length descriptions of the rules while, at the same time, generalising over attributes.

## Chapter 4 - CITE: Scene understanding and Object Recognition

In this chapter, we show how recent machine learning techniques can be used in building adaptive and trainable image annotation systems using deeper structures than in the two systems described above. In this case, the annotation is defined with respect to image regions, hierarchies and

cliques which are of specific importance to the application. Our concern has been to use machine learning to bind domain knowledge with data-driven sensing and to investigate how both, domain knowledge and sensed data can constrain learning. Specifically, we show when conventional induction techniques can be used and when they cannot be used.

In the CITE system, it is important to recognise or hypothesise about structures from partial data. Induction occurs simultaneously over structures at different scales, and it is controlled by the types of (parent and sibling) relationships within the knowledge base. CITE automatically generates descriptions in terms of hierarchical lists of labelled and attributed parent and sibling nodes which also contain relational information. The system architecture consists of three different structures, the image model, the scene model, and the knowledge representation. The system uses incremental learning and during a learning phase example objects, structures and complete images are presented.

CITE segments image data using a constrained form of clustering. The segments are then used to build knowledge about specific structures as, for example, buildings, components of buildings and how buildings relate to other scene characteristics. Using an expert during this phase, CITE learns to re-segment the data as a function of the degree to which the parts are consistent with how the expert defines, annotates, the given scene data with respect to known domain knowledge. As consistency filters through the hierarchical system, each node extracts common attributes of the training data and generates rules with attribute bounds which summarise what has been presented. Compatibility between new examples and past learnt models is checked using hierarchical relaxation in order to determine if a given image is novel and to build up new representations if it is indeed novel.

## Chapter 5 - SEE++: Learning and Querying in dynamic environments

This chapter deals with the development of a system for the on-line annotation or querying of specific structures in dynamic image data. It investigates the use of machine learning to generate structural descriptions, which can be used for the automated interpretation of scenes in terms of previously explored environments. In particular, it focuses on the integration of low-level image processing and high-level image and scene interpretation into a complete system and the use of machine learning to achieve this integration in a task-specific way. Low-level image analysis is controlled in a closed-loop fashion where high-level task-specific knowledge is used to recognise failures in low-level image analysis (e.g. image segmentation) and to send corrective requests to these low-level stages.

The SEE++ architecture is one in which task-specific knowledge is em-

bedded within a lattice structure. This knowledge base is primarily responsible for applying high-level reasoning and constructing scene interpretations. The knowledge base is served by a low-level vision system which provides a structured environment for image acquisition, processing and segmentation.

## Chapter 6 - SOO-PIN: Symbolic picture interpretation

SOO-PIN symbolic picture interpretation This chapter introduces a system for high-level interpretation of images, and is illustrated with an application to the interpretation of traffic scenes. The system extracts events from sequences of images (e.g. the movement of cars at intersections), and attempts to find high-level interpretations in these event sequences ("car 356 is making an illegal left turn"). The system consists of a network of agents or frames, each asynchronously working on its own aspect of the interpretation and passing messages to other agents. Knowledge is represented by networks of frames where each frame consists of facts and rules which define different sub-interpretation processes defined by facts and rules depicting different rules and scenarios. Some of the major characteristics of the system include:

- The use of object-oriented concurrent logic programming for scene analysis.
- The use of uncertainty calculus (e.g. Dempster-Shafer) to keep track of interpretations and the system's confidence in their validity.
- The distinction between symbolic and spatial data, and recognising the need to ground symbolic predicates in the (image) data.
- Class hierarchies and inheritance can be formed using "filters" which actively pass messages from a class to superclass when the method to handle a message is not available in the class.

The system is implemented in Parlog++, an object-oriented parallel version of the language Prolog.

Recently, Ollila and Caelli [236] have replaced such symbolic rules by algebraic equations and techniques from geometric reasoning [165] to result in what we have termed "algebraic metrology." A practical application of this system is that of "proving" that scientific drawings of 3D objects are correct or not. The reader is referred to Ollila and Caelli [236] for more detail.

## Chapter 7 - MBNN: Model-based Neural Networks for invariant feature extraction

Chapters 2 to 6 have dealt with the development of technologies which allow us to label images with respect to domain knowledge based upon current machine learning and object-oriented models and procedures. This chapter deals with a different approach, namely, the design and training of neural networks in ways which allow the designer to incorporate expert domain knowledge into the network. The networks are modular, and each module performs an *a priori* specified sub-task. Module performance can be specified by parameterising the weights, training on a sub-task, or direct hand-coding of the weights. The networks can thus be guaranteed to find solutions that are consistent with domain constraints. This approach is illustrated with a network that computes invariant vector fields and is able to recognise 2D-patterns invariant to position, orientation and size—important for image interpretation.

## Chapter 8 - ABC: Biologically motivated image understanding

In this chapter, we consider a biologically oriented approach to image interpretation. It deals with issues related to the development of a system which is consistent with the known neurophysiology of the visual system and is aimed at predicting actions of humans when perceiving images. These include issues related to focal eye movements and to the automated generation of perceptual categories necessary for the symbolic description of images.

What is important in this chapter is the exploration of, essentially a single model, which, when made hierarchical, recurrent and with reinforcement it can attain a wide range of perceptual activity. The system consists of hierarchies of self-organizing maps which are updated by recurrent feedback and which associate information from different modalities and extracts more advanced representations of the sensory input.

## 1.4 Conclusions

We hope that the reader will learn to appreciate in the following chapters how challenging problems in image interpretation can be. We also hope to show what types of data structures, learning and matching procedures are needed to solve such interpretation problems. We hope this work will set the stage for a new way of conceiving and implementing image interpretation systems which explore machine learning to bind knowledge with data-driven processes. We leave it to the readers to make their own conclusions as to whether hybrid or self-similar architectures are more appropriate

or theoretically elegant. We hope that the book will initiate the integration of image annotation, image interpretation with other areas of computer vision, pattern recognition and artificial intelligence.

## Chapter 2

# Fuzzy Conditional Rule Generation for the Learning and Recognition of 3D Objects from 2D Images

Brendan McCane and Terry Caelli

### Abstract

In this chapter we explore the use of machine learning (ML) for the recognition of 3D objects in isolation or embedded in scenes. Of particular interest is the use of a recent ML technique (specifically CRG - Conditional Rule Generation) which generates descriptions of objects in terms of object parts and part relational attribute bounds. We show how this technique can be combined with intensity-based model and scene views to locate objects and their pose. The major contributions of this paper are: the extension of the CRG classifier to incorporate fuzzy decisions (FCRG), the application of the FCRG classifier to the problem of learning 3D objects from 2D intensity images, the study of the usefulness of sparse depth data in regards to recognition performance, and the implementation of a complete object recognition system that does not rely on perfect or synthetic data. We report a recognition rate of 80% for unseen single object scenes in a database of 18 non-trivial objects.

## 2.1 Introduction

Three-dimensional object recognition and scene understanding are important goals in computer vision research. Many object recognition systems have been proposed and can be differentiated in terms of:

1. how object models are created,
2. the types of sensing technologies involved,
3. the methods for encoding model and image features,
4. how model features are instantiated in image data to enable model identification.

Models are either computer generated (e.g., by CAD) or sensed by active or passive means, resulting in a number of model “views”. Such views typically consist of intensity, intensity and (sparse) depth, or range data. In each case, model features are usually extracted in the form of region or boundary predicates and are represented as relational graphs describing part and part relation attributes. Consequently, recognition algorithms can be considered as constraint propagation methods implemented as constrained tree search, geometric hashing or (parallel) rule evaluation.

It is important to note that many applications of object recognition are not conducive to computer-generated models. In this paper we address the issues of object recognition from a Machine Learning (ML) perspective—that is, object models are learnt from view-dependent sensed data. Both range and intensity sensing are view-dependent and, in this paper, we use intensity-based model data, though the essential new developments equally apply to range data. Recently Bischof and Caelli [30, 31] (see also Chapter 1) have described a learning system called Conditional Rule Generation (CRG), for learning objects from examples. This system uses “learning-from-examples” and is based on the “recognition-by-parts” paradigm. Also, since it is a rule-based tree type classifier, it can efficiently recognise a large number of objects, alone, or, in complex scenes. Hence, the CRG classifier can be used to solve many of the problems inherent in the recognition of 3D objects from 2D images. The CRG classifier is described more fully in Section 2.5. The primary contributions of this chapter are:

1. The extension of the CRG classifier to a fuzzy technique (FCRG - Fuzzy CRG).
2. The application of FCRG to the problem of learning 3D objects from 2D intensity images and the careful study of different aspects of this problem.
3. The evaluation of a number of different unary attribute sets.

4. The study of the usefulness of depth data in recognition obtained by using a depth-from-stereo algorithm.
5. The implementation of a complete object recognition system that does not rely on “perfect” or synthetic data.

This chapter is organised as follows. In the following section, a brief literature review is given. Section 2.3 describes the input data, the segmentation and stereo algorithms used in the experiments. Section 2.4 describes the attributes used to learn and recognise objects. Section 2.5 describes the CRG and FCRG classifiers. Section 2.6 describes the hypothesis verification procedure which is used. Finally, results are presented in Section 2.7 and conclusions are drawn in Section 2.8.

## 2.2 Literature Review

Recently, Bergevin and Levine [24] describe PARVO, a system that performs generic object recognition from 2D line drawings. This system is an attempt to implement the “recognition-by-components” theory developed by Biederman [29]. PARVO recognises single objects from line drawings by first generating a coarse part-based volumetric description of the object which, in turn, is achieved by first segmenting the line drawing into parts using perceptual organisation techniques (cotermination, parallelism, symmetry). Each part is then labelled as one of a number of volumetric primitives (“geons”) using an hypothesis-and-test procedure. The object models to be recognised are specified to the PARVO system prior to object recognition via a generic graph representation (the geon type is represented as a graph node, and relations between the graphs are represented as the graph arcs). Since model and scene representations are equivalent, the problem becomes a sub-graph isomorphism problem. However, due to the computational complexity of such an approach, only objects that contain all geons found in the image are considered as possible interpretations of the image. From the remaining possible object models, a similarity measure is computed between the object model and the scene. The model which scores the highest similarity is considered the most likely object in the scene—a type of “generalisation” procedure. The two major limitations of this approach are that PARVO assumes that objects in the scene have already been detected and the input is in the form of “perfect”, or near-perfect line drawings.

Dickinson, Pentland and Rosenfeld [83, 84] also describe a system which implements the “recognition-by-components” theory, although using different techniques. Objects are modelled using 3D volumetric primitives in a CAD-like model. Test scenes are presented to the system using “perfect” 2D line drawings and the system can apparently handle multiple object

scenes. The main form of recognition is an indexing approach and their major contribution is the means by which 3D volumetric primitives are extracted from a 2D image. The system uses a set of 10 volumetric primitives similar to the geons described by Biederman [29]. The primitives are represented by a hierarchy of aspects, faces and boundary groups. Each element in the aspect hierarchy is assigned a probability indicating the likelihood of that element being visible in an image, which can be calculated from the object models. In general, the first step in the recognition phase is to recognise the primitives that are visible in the image. The authors suggest that this can be done using one of the many segmentation algorithms reported in the literature, but to simplify the problem, they used only synthetic images (line drawings). Primitives were recognised using a hierarchical graph matching procedure and since there were typically few primitives, this was not computationally expensive. From the extracted primitives, a hashing scheme was used to generate object hypotheses which were then verified using an hypothesis verification scheme. The advantages of such a scheme are many. Multiple object scenes are handled the same as single object scenes. Expensive graph matching is only performed on a small set of volumetric primitives, thus ensuring the extraction of primitives is computationally feasible. Also, the scheme is flexible and can be applied to either intensity or range images in a similar manner. Although the system is general, its testing has not been thorough as yet and hence, the usefulness of the system is still in question. Also, the possible problems inherent at lower levels of processing (image segmentation, contour segmentation) were not addressed in the paper except in the context of synthetic images. So, although the system appears promising, it is difficult to judge how successful it will be until more comprehensive tests have been performed in more realistic situations.

Both of the strategies and systems discussed so far have been model-based systems and representative of the prevalent approach for recognising 3D objects from 2D intensity images. Although this approach is reasonable for CAD/CAM applications, it is not general enough for other domains (free-form objects) nor representative of human performance. In the latter case, it is well documented that humans learn objects from exposure to different views [94]. Humans seem to be able to learn to recognise 3D objects using information gathered mostly from 2D images. Whether the problem is solved by first “mentally constructing” a 3D object model, or by using mainly 2D representations remains an open question. A significant amount of research has attempted to reconstruct 3D depth maps from 2D images (stereo, focus, shading), but relatively little work has been done in the area of learning to recognise 3D objects from 2D intensity images.

Poggio and Edelman [260] describe a neural network that utilises generalised radial basis functions for learning object models from examples. Their system suffers from a number of disadvantages. Firstly, the objects used for testing the system are wireframe objects for which all features are

visible from each view. Hence, the problem is simplified because no self-occlusion is evident and the object parts are simple and can be represented by the vertices of the wireframe. Secondly, for each object a new network is required for the learning and recognition of the object, and this would be a prohibitive approach if the object database contained many objects. Thirdly, only one object is presented to each network at a time, so presumably the object detection problem would have to be solved prior to recognition. Edelman and Weinshall [93] describe a similar approach that utilises a two layer network of thresholded summation units and an unsupervised Hebbian learning strategy. However, they also use only wireframe objects and only present one object at a time to the system.

More recently, Mundy et al. [223, 224] describe a complex 3D object recognition system (MORSE) that uses geometric invariances for description and matching. Object models are acquired from multiple 2D intensity images and recognition is performed on single 2D intensity scenes. Models are represented by a number of affine 3D geometric invariants including: cross-ratios of points on rotationally symmetric image features; symmetrical feature correspondences on bilateral symmetries and more general symmetries; projective invariants of algebraic surfaces, etc. The system recognises object classes where each class is described geometrically (surfaces of revolution and polyhedrons are used in [224]). MORSE uses a fairly standard control system involving segmentation and grouping, indexing and hypothesis generation, and verification. At the grouping stage, pixel level features are grouped into linked edges from which geometric features such as algebraic curves, smooth curves and concavities are extracted. Generic grouping is then performed on the geometric features to create a database of groupings that include near incidence, collinearity, marking bi-tangent points and affine or projective equivalence of curve segments. Finally, each class has an associated “grouper” which interrogates the generic groupings and attempts to build groupings appropriate for its class. Indexing and hypothesis verification is handled by a series of hash tables, one per class, that associate the invariants of a system of generalised features with models in the model-base. Verification is performed at many levels of the hypothesis formation process, including verification of classes and lower-level groupings (of polyhedral faces). It is premature to evaluate MORSE, since it is a prototype system which is still under development. However, in its current form [224], it would be more accurate to describe it as a “shape” recognition system rather than an object recognition system. This is because it recognises classes of shapes such as surfaces of revolution and polyhedral objects. To be considered a true “object” recognition system, MORSE must be able to handle complex objects which include conglomerations of different free-form shape classes. Although this has not yet been demonstrated, it could presumably be achieved by modifying the indexing scheme to allow different shape class hash tables to index into different

object models.

In the area of range-based object recognition, two systems have been developed which function with free-form objects and which use Machine Learning techniques to generate object descriptions from multiple views. In the original system developed by Jain and Hoffman [153], attributes of segmented (range) view parts are used to determine feature bounds which evidence different objects and these rules are then evaluated over individual object samples to result in an overall classification of a given sample. This evidential system was extended by Caelli and Dreier [50] to generate more discriminating rules by using different methods for generating attribute bounds (on parts and relations) and evidence values. Both systems [50, 153] have been quite successful in recognising isolated objects when many (more than 20) objects have been used. However, they were not designed to efficiently handle multiple object scenes.

Porrill et al. [264] describe a 3D vision system (TINA) for locating objects in a scene and picking them up using a robot arm. The system is not an object recognition system per se, although it could be extended in a fairly straightforward manner, but is important here as it the only system (to the author's knowledge) that uses 3D data obtained from stereo images to match objects. Stereo data is extracted from a pair of stereo images using the PMF algorithm [261]. Straight lines, planes and space curves are then extracted from the depth edge data, and a 3D geometric wireframe description of the scene is produced. A match is then performed between the model and the scene using a focused-feature graph-matching procedure. From the results of the matching process, and other image information, the model is updated from the scene using statistical combination of the model and scene. This can occur across several poses of the object to form a complete object model. The test images reported in [264] were scenes of man made objects (with prominent edges and simple surface shading) in good illumination, and the system works very well under these conditions. It would be interesting to see the performance of TINA under less ideal conditions.

In summary, most intensity based object recognition systems presented to date have some limitations and do not fully address all the problems inherent in such a complex domain. The main criticisms of the systems described so far is not in the methods used for solving the problems, but rather in the problems themselves. Many authors use synthetic images to simplify the problem and thus ignore the problems inherent at the lower levels of processing (segmentation etc.) for which it appears there will be no significantly better solution than current solutions present in the literature. We do not claim that incremental improvements to low level problems will not be made, only that significantly better solutions seem unlikely. In the system described in this chapter, we have not used synthetic images for recognition or learning, and thus have attempted to solve a more

realistic problem. Also, very few systems attempt to solve the learning-from-example problem, and those that do are typically demonstrated only in simplistic problem domains. Moreover, we present a system which addresses many of the problems listed in Bowyer et al. [38]. Specifically, we address the problems of learning with structured representations, handling uncertainty, handling partial information and learning with many classes. We have adopted a structured representation for learning 3D objects and have handled uncertainty by using fuzzy clustering techniques. The technique can handle partial information (indeed different views of an object are in fact partial information). Finally, the technique is designed to handle many classes, and can handle multiple classes in one test image in a natural way.

## 2.3 Input Data

As was stated in the Introduction, the problem domain is the learning and recognition of 3D objects from 2D intensity images of multiple object scenes. The problem is an extremely difficult one, and to reduce the complexity, it was somewhat simplified by presenting objects on a black background, thus removing the need for performing extensive figure-ground separation. There are two classes of images which are presented to the system, single-object scenes and multiple-object scenes. The single-object scenes are used primarily for learning, although recognition results are also presented with these scenes using a cross-validation procedure (see Section 2.7). The multiple-object scenes are used for testing the efficacy of the system in more complex environments.

The objects used in these scenes were chosen such that they were complex enough to make the recognition task difficult yet still possible. Some of the objects are quite similar in shape, and this helps to illustrate the discrimination power of the classifier. Figure 2.1 displays all the objects used in the experiment from a typical viewpoint. There were 18 objects in all, and for the single object scenes, each object was presented in 5 or 6 different poses, at approximately equal angles over an approximately 90° angular variation (see Figure 2.2). Images were taken of each of the poses using a single camera mounted on a sliding tripod which was setup to take stereo images using a horizontal stereo baseline of 10cm. The camera was approximately one metre away from the objects. Figure 2.2 displays all the views of the calculator object.

The multiple object scenes contained between 2 and 5 objects, with each object being presented at a pose which was not necessarily the same as one of the single object scenes, but nevertheless occurred within the range of possible angular variation. Figure 2.3 displays some of the images used for multiple object scenes. Again, stereo images of each scene were used.

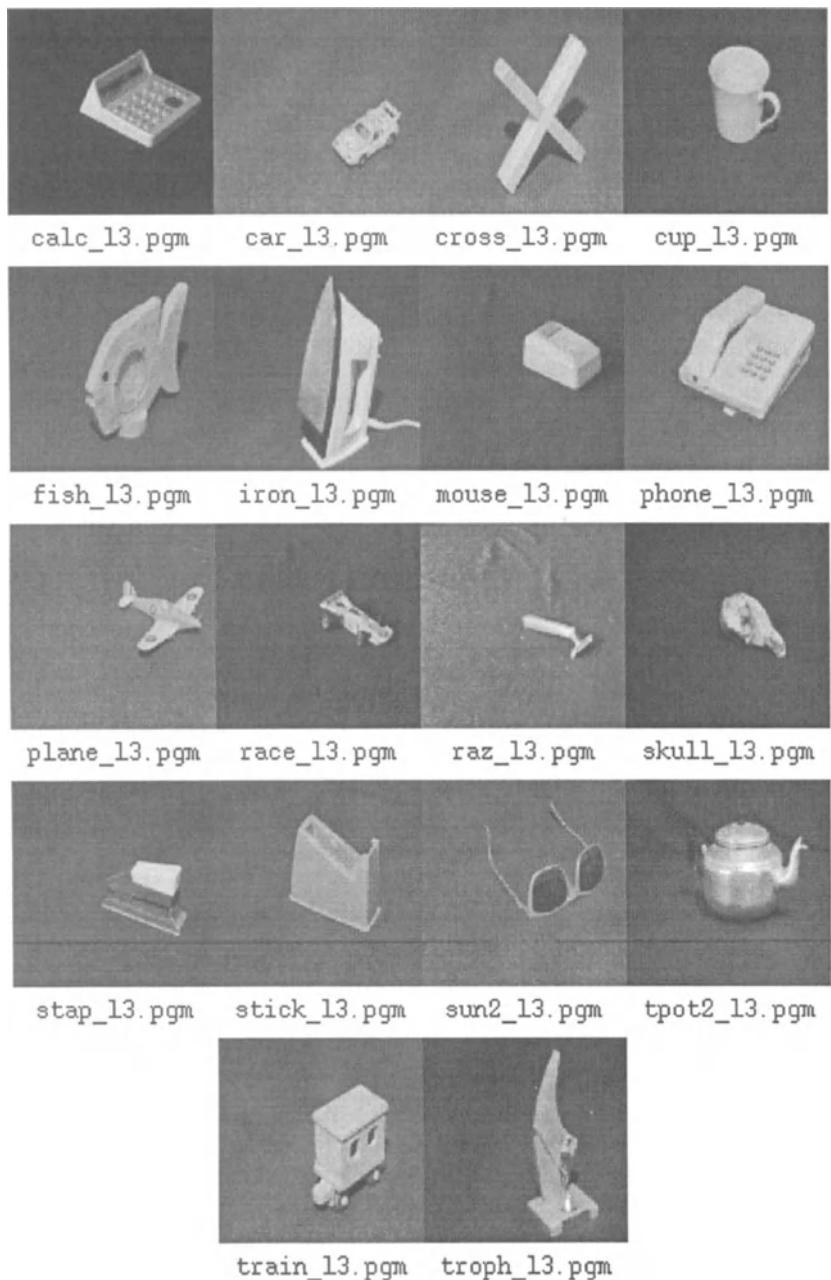


Figure 2.1: Examples of all the objects used in the experiment

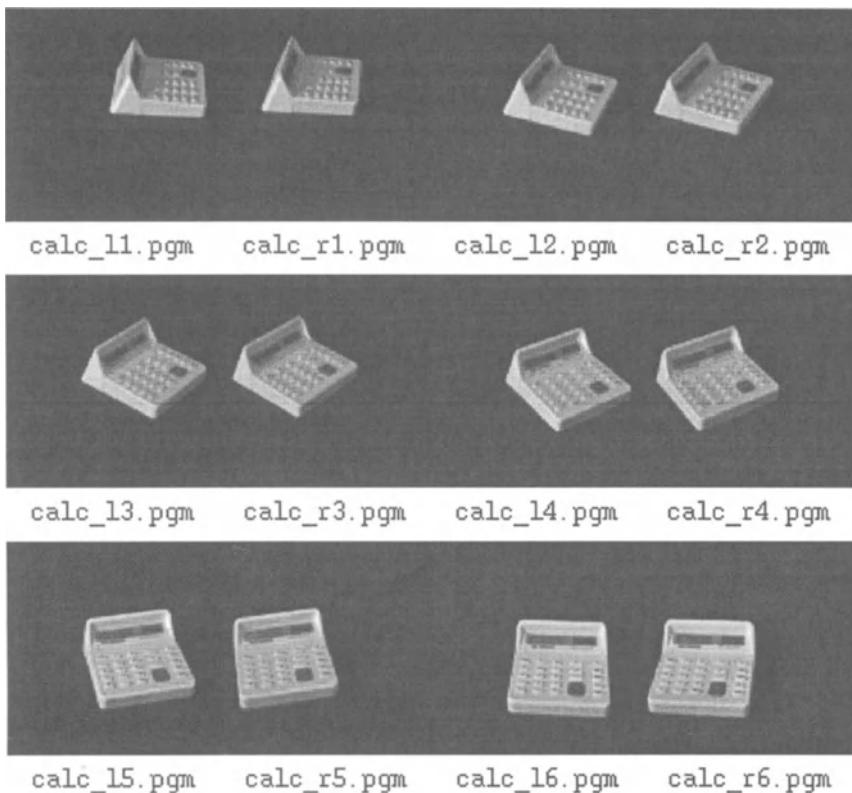


Figure 2.2: All poses of the calculator object, corresponding stereo views are displayed next to each other

### 2.3.1 Segmentation

Since we are using the “recognition-by-parts” paradigm for recognising objects, it is necessary to segment a given image into its constituent parts. Two major approaches to the segmentation problem include edge-based and region-based methods. Edge-based approaches find edges in an image, but “edge elements often exist where no meaningful scene boundary does, and conversely are often absent where a boundary is” [18, page 119], and hence do not always produce good regions (i.e., they may not produce closed regions). On the other hand, region-based methods produce closed regions, but typically do not make use of edge information. Taking these problems into account, it is reasonable to suggest that “region-finding and line-finding techniques can cooperate to produce a more reliable segmentation” [18, page 149]. Other authors have also investigated such procedures,

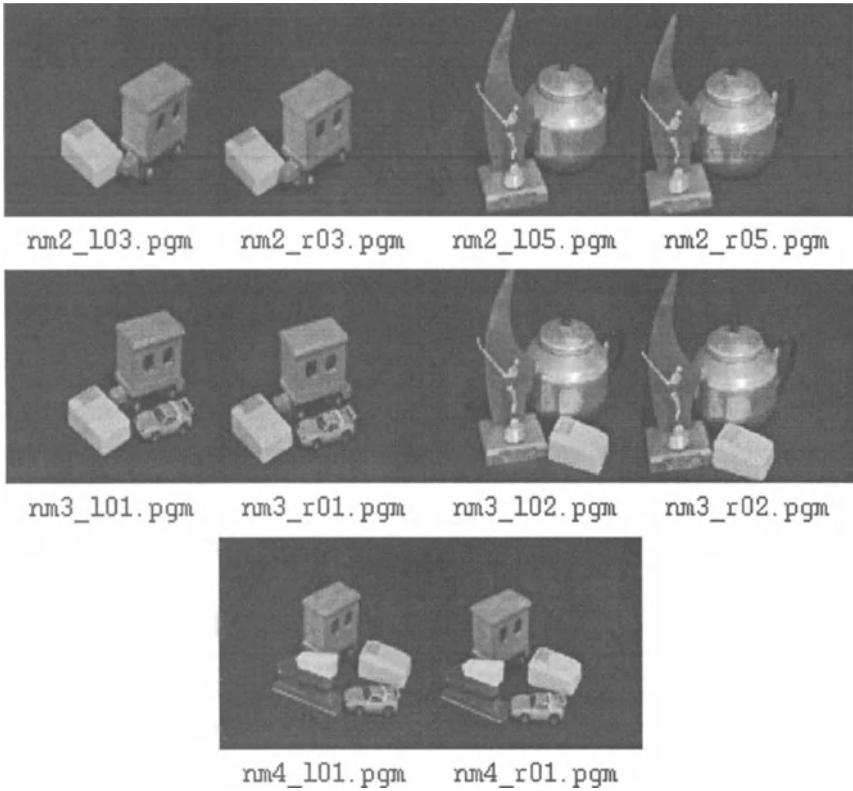


Figure 2.3: Some examples of the multiple object scenes used in the current study.

including Pavlidis and Liow [245], Haddon and Boyce [131], and Chu and Aggarwal [66].

The algorithm used here for segmentation of the input images was developed by McCane and Caelli [200]. Prior to applying the segmentation algorithm, a figure-ground separation procedure is used to separate the objects from the background. Since there is typically more background than foreground for the images used here, the largest peak in the intensity histogram is found and assumed to be the background. A threshold point is then set at the next minimum (frequency) in the histogram, below which are background pixels, and above which are foreground pixels. This procedure is not meant to be general and is used for aiding the object recognition system. The segmentation algorithm is valid with or without figure-ground separation.

The segmenter utilises a multi-scale approach to segmentation. The

procedure uses both edge and region information to adaptively and automatically choose the best scale at which to segment different parts of an image. In this way, images with widely varying characteristics can be successfully segmented using one procedure. The basic segmentation algorithm is defined recursively as follows:

1. Edges are extracted from an input image at a given (largest) scale using Canny's edge detection algorithm [56, 57].
2. A parallel edge growing procedure is performed to close any small gaps in the edges.
3. Regions are formed using a simple boundary fill algorithm on the edge image.
4. The intensity variance of the region is measured and the region is further segmented at the next scale down (recursively calling this procedure).
5. If the average variance of the child regions is less than the variance of the current region, then the current region is replaced by the child regions.

### 2.3.1.1 Edge Extraction

Initially, edge points are found in an input image using Canny's edge detection algorithm [56, 57]. These are then linked into edges using an hysteresis linking routine. For the current set of images, the low hysteresis threshold was set to 0 and the high set to 25 (the input images are greyscale with intensities ranging from 0 to 255). Junctions are then formed at points where two or more edges meet. Those edges with isolated endpoints are extended in a parallel fashion until they hit another edge, or until a maximum extension limit is reached. Although it would be useful sometimes if small edges could be extrapolated over large gaps, more often such edges would be misleading. Hence, the maximum extension possible for a given edge is limited by the length of that edge. In a sense, this procedure can be considered a type of primitive perceptual organisation where previously unrelated edges are joined to form larger edges. The current procedure uses linear extrapolation, but extrapolation techniques using more complex curves could be used. This was found to be unnecessary for the current application, since the maximum extension of an edge is typically small.

### 2.3.1.2 The Adaptive Segmentation Procedure

The adaptive segmentation procedure is recursive and makes use of both edge and region information to perform the segmentation. The actual segmentation is performed using edge information, but the decision on whether

to further segment a region is based upon region statistics. In this case, the statistic used is the variance of the intensity inside the region, but other statistics (texture statistics, for example) are equally applicable. The decision to further segment a given region is not based on a thresholding of the region statistic. Instead, a decision is made by comparing the region statistics of the sub-regions at the next smallest scale to the region statistic at the current scale. In this way, the segmentation process occurs automatically in a self-similar way, similar to the method introduced by Lowe [186] for segmenting edges into linear edge segments.

The region statistic which has been used is the *mean absolute deviation* or the *mean deviation* [269] of the intensity values within a region. This measure tends to be more robust than the variance in that it is more stable in the presence of outliers (e.g., salt and pepper noise), especially with small sample sizes. The *mean absolute deviation* is defined as:

$$ADev(x_1 \dots x_N) = \frac{1}{N} \sum_{j=1}^N |x_j - \bar{x}| \quad (2.1)$$

The result of applying this algorithm to the entire image is a segmented image (represented as a list of regions) where each region corresponds to a segmentation of the input image at one of the input scales. Different regions may correspond to segmentations at different scales. Hence, the final segmentation is not limited to one scale, but may include information from a number of scales. A more detailed description of the technique can be found in [201]. Figure 2.4 displays examples of the resultant segmentation using two scales defined by  $\sigma_1 = 2$  pixels and  $\sigma_2 = 1$  pixel.

### 2.3.2 Stereo

The stereo algorithm used here was developed by McCane and de Vel [204] and is based on matching edge segments in the left and right images using cluster analysis. The idea of the algorithm is to find the nearest neighbours in attribute space between the left and right images. The attributes used were the number of pixels in an edge segment, the coordinates of the centroid of the segment, the first and second moments (mean and variance) of the distance of each pixel in the segment from the centroid, and the orientation of the major axis of the segment. A more detailed discussion of the stereo algorithm can be found in [204] who report an accuracy of 90% for correct matches of the algorithm over a variety of objects and disparities. Some results are shown in Figure 2.5.

The procedure matches edges well and produces a disparity map as output. In the current system, the stereo data is only used for calculating relational attributes (relationships between parts in a scene), hence it is not necessary to recover absolute depth values from the disparity map.

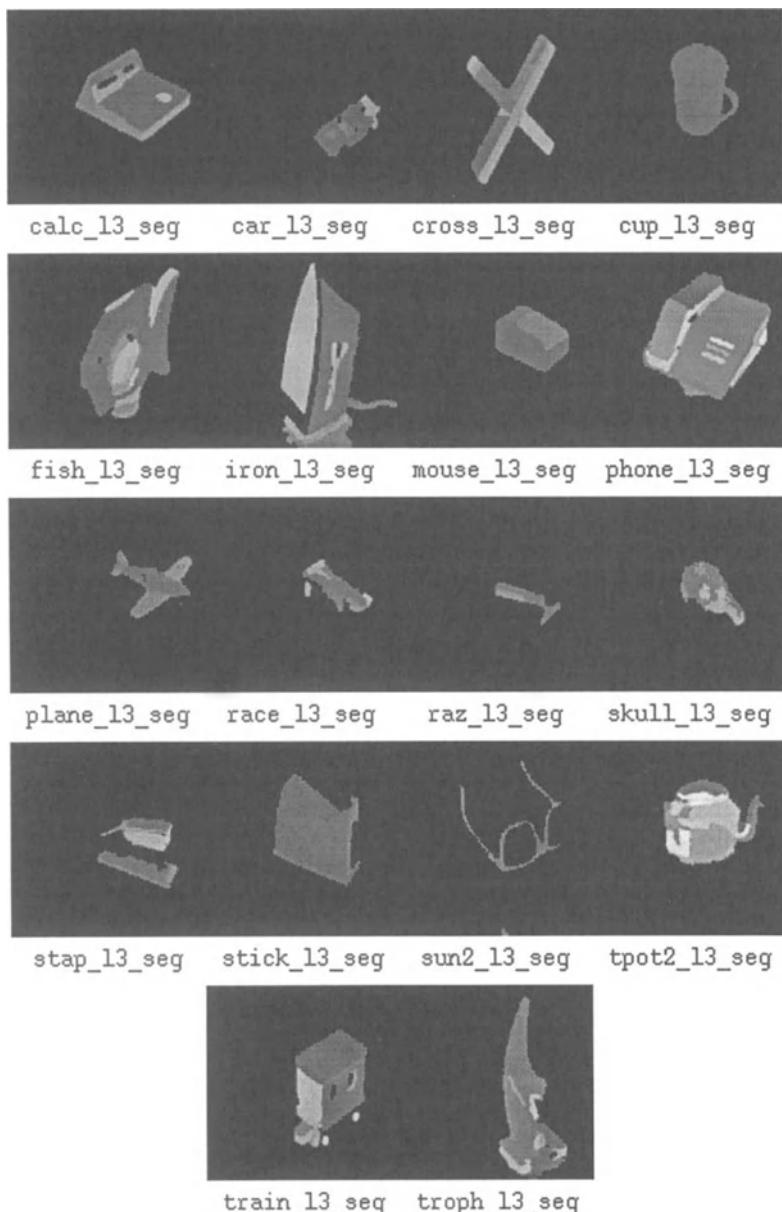


Figure 2.4: Examples of the segmentation ( $\sigma_1 = 2.0$  and  $\sigma_2 = 1.0$ ) of the left stereo images displayed in Figure 2.3

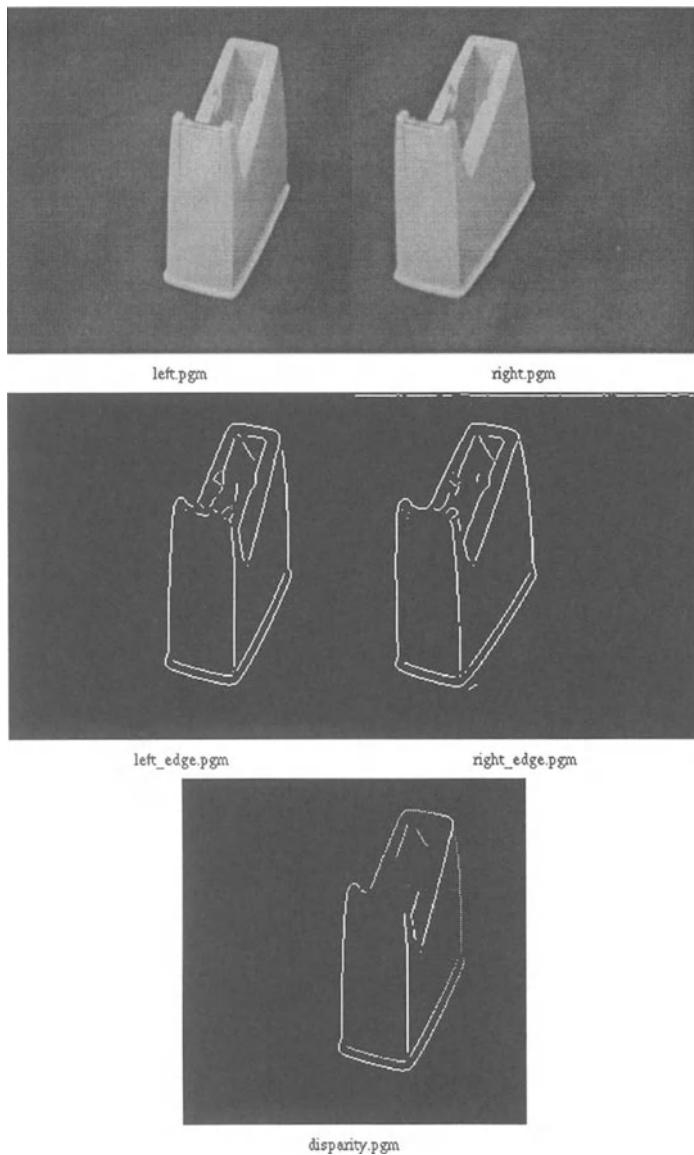


Figure 2.5: An example of the stereo algorithm: (a) and (b) are the left and right images; (c) and (d) are the corresponding edge images; and (e) is the resulting disparity map—brighter edges are closer to the viewer than darker ones.

However, such information could be used for estimating the absolute size of the objects in question, or as a cue for size invariant object recognition.

## 2.4 Features and their Attributes

The fuzzy CRG classifier makes use of both unary (part) and binary (part relation) attributes. In the current system, parts correspond to image regions which are generated by the segmentation procedure described in the previous section. In order to generate robust rules to describe the training objects, it is necessary to select part attributes which are invariant to two-dimensional transformations and are maximally discriminating between objects. Since this system is concerned with two-dimensional projections of models and data, fully three-dimensional invariant attributes may not contribute to the recognition problem. In fact, Burns et al. [49] prove that there exists “no function of the perspective projection on  $n$  3-D points for any integer  $n$  that is both a general-case view invariant and nontrivial” (see [49] for more details). Nevertheless, attributes which are invariant over a number of poses may be useful, and we evaluate different unary and binary attribute sets to determine the best set of attributes for the problem.

### 2.4.1 Unary Attributes

In this section a number of different sets of unary attributes are described. These include Hu invariants, Fourier descriptors of the boundary, Hadamard descriptors of the boundary, and arboricity measures. Each of the attribute sets was evaluated in terms of recognition performance and the results are presented in Section 2.7.

#### 2.4.1.1 Hu Invariants

There has been significant research effort into the development of moment invariants which were first introduced by Hu [143] and further studied by various researchers [1, 2, 22, 193, 335]. The basic idea of moment invariants is to define a set of measures which are invariant to scale, rotation, translation, and contrast changes in a 2D plane. The justification for using them in the current context is that they are valid for recognising a part from a particular pose which may be rotated, translated or scaled from the original training pose. The following definitions have been taken from Maitra [193].

Given a 2D image intensity distribution  $f(x, y)$ , the moments of this function are defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy, \text{ for } p, q = 0, 1, 2, \dots \quad (2.2)$$

The above invariants can be modified to include translational invariance in the following way:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy, \quad (2.3)$$

where  $\bar{x} = \frac{m_{10}}{m_{00}}$ , and  $\bar{y} = \frac{m_{01}}{m_{00}}$ . Scale invariant moments can be derived from the above to give a set of normalised central moments:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \text{ where } \gamma = \frac{p+q}{2} + 1. \quad (2.4)$$

A set of 7 functions can then be defined which are invariant to translation, rotation, and scale changes in the image plane:

$$\phi(1) = \eta_{20} + \eta_{02} \quad (2.5)$$

$$\phi(2) = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (2.6)$$

$$\phi(3) = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (2.7)$$

$$\phi(4) = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (2.8)$$

$$\begin{aligned} \phi(5) = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (2.9)$$

$$\begin{aligned} \phi(6) = & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned} \quad (2.10)$$

$$\begin{aligned} \phi(7) = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (2.11)$$

Contrast invariance can also be included (see [193]), but a discussion of contrast invariance will not be included here.

#### 2.4.1.2 General Shape Attributes

The general shape attributes were used in an attempt to capture an intuitive measure of shape. Many of the attributes extracted rely on the extraction of the major and minor axes of a region which are extracted by calculating its two principal components ( $\vec{v}_1, \vec{v}_2$ ) [123]. We have chosen the following region attributes derived from such analysis: aspect ratio, mean and variance width difference, compactness ( $perimeter^2/area$ ), convex deficiency. The aspect ratio is calculated as the ratio of length of the minor axis to the length of the major axis (actually, ratios of the eigenvalues of the principal components). The mean and variance of the width difference has been designed as a measure of part symmetry. These attributes are calculated using the difference between the distances from the major axis and the point pairs on the boundary perpendicular to the major axis:

$$s = \frac{1}{N} \sum_{i=1}^N (d(\vec{A}_i, \vec{P}_{i1}) - d(\vec{A}_i, \vec{P}_{i2})), \quad (2.12)$$

where  $N$  is the number of pixels along the major axis  $A$ ,  $d(\vec{x}, \vec{y})$  is a Euclidean distance measure,  $\vec{A}_i$  is the  $i$ th point along the major axis  $A$ , and

$\vec{P}_{i1}$  and  $\vec{P}_{i2}$  are the two points which lie along the perimeter of the region  $P$  and lie on a line which passes through  $\vec{A}_i$  and is perpendicular to  $A$ . Such a measure is only valid for regions for which the following is true: for each point on the major axis there exist at most two points at which the line perpendicular to the major axis, and passing through the point, intersects with the region boundary. Fortunately, most regions in the current experiments satisfy this condition and for those that do not, the two points furthest away from the axis are chosen for the symmetry calculation. However, extensions which include extrema could be used if this definition was not appropriate. The compactness of the region provides a measure of how close the region is to a circle. The convex deficiency of a region is calculated as:

$$\frac{\mathcal{A}(C) - \mathcal{A}(R)}{\mathcal{A}(C)} \quad (2.13)$$

where  $R$  and  $C$  represent the region and its convex hull respectively, and the operator  $\mathcal{A}$  represents the area of a given region. The convex hull was calculated using an algorithm due to Graham [125, 268].

#### 2.4.1.3 Arboricity Measures

There are many variations that can be made to the general shape attributes approach in the hope of improving the recognition performance of the system. One such extension is the use of arboricity measures. Arboricity measures attempt to capture more complex symmetrical features which may occur in some shapes. They rely heavily on the notion of symmetrical axes evident within a shape and in this case are calculated from the principal axis of a region which is extracted as described above. After attributes have been extracted from the whole region, it is split into two regions (perpendicular to the principal axis) and further measurements are made on the regions thus created. Figure 2.6 depicts the essence of the idea.

The attributes used for each of the stages are the same as those described above in Section 2.4.1.2, thus hopefully providing a more powerful representation of each part. The question still remains about how many levels to split the parts into (all parts must be split the same amount to give comparable representations across different parts). Care must be taken here as using too many splits quickly introduces the “curse of dimensionality”. With 5 base attributes, even 1 splitting level results in 15 attributes—5 for the top level and 5 for each of the two sub-levels. With two levels the number of attributes is 35, so parts should definitely not be split more than 2 levels. Intuitively, this would seem to be enough to represent most shapes since the shapes typically are not too complex.

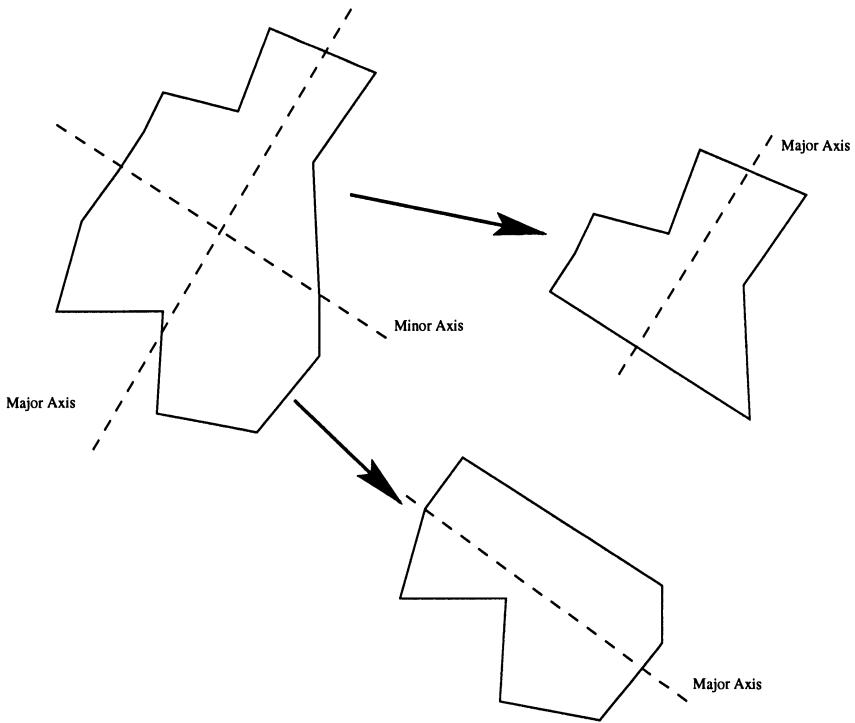


Figure 2.6: Arboricity measurements: measurements are taken initially about the major axis, the region is then split into two along the minor axis. For the two subregions, the major axis is computed and more measurements are taken. This procedure can continue for as long as required in a recursive manner.

#### 2.4.1.4 Fourier Descriptors

The final three attribute sets make use of a cumulative curvature boundary representation of each of the parts. Here, points along the boundary of a curve are represented by the difference between the tangent at the current point and the tangent at the starting point on the curve (wherever that is defined). Figure 2.7 displays the definition of cumulative curvature. Tangents can be calculated from discrete data by measuring the angle made by the current point and a point on the curve  $k$  points in front or behind the current point.

In the discrete domain the Fourier transform can be written as:

$$F_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{2\pi i k n / N} \quad (2.14)$$

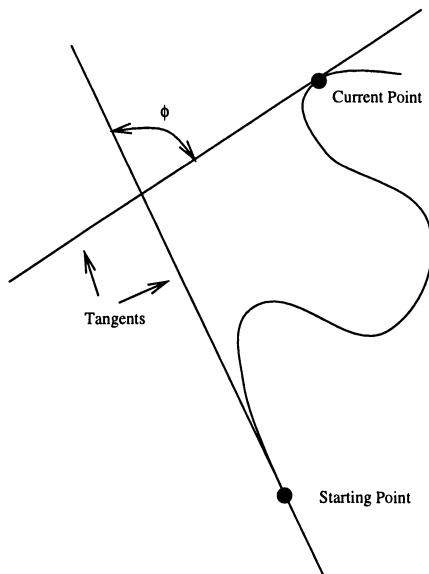


Figure 2.7: Definition of Cumulative Curvature.

where  $i = \sqrt{-1}$ , and  $n$  is defined over the same range as  $k$ . The idea of using Fourier descriptors is to truncate the Fourier series down to the first few coefficients, and use these to represent the curve. The curve can be reconstructed from the  $L$  lower-order coefficients using the following formula:

$$f_k = \sum_{n=0}^{L-1} F_n e^{-2\pi i k n / N} \quad (2.15)$$

It should be noted that all the values of  $f_k$  and  $F_n$  are complex numbers, so if  $L$  coefficients are used to represent a curve, then  $2L$  attributes will be needed (the real and imaginary parts of the  $F_n$ s). To obtain the initial sequence  $f_k$ , the cumulative curvature function is sampled at 256 data points along the curve. If there are less than 256 points in the curve, then linear interpolation is used to fill in the gaps.

#### 2.4.1.5 Sinusoid Descriptors

Unfortunately, there is some redundancy in using Fourier descriptors to represent real functions—the function is transformed from a real function containing  $N$  information points to a complex function containing  $2N$  information points. This problem can be overcome by utilising the sine transform which was introduced by Jain [154]. The sine transform (forward and

inverse) is defined by:

$$S_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} \sin \left[ \frac{(n+1)(k+1)\pi}{N+1} \right] \quad (2.16)$$

The original curve can be reconstructed from the  $L$  lower order sine descriptors in a similar way to Equation 2.15.

It is worth remarking that the sine transform was arbitrarily chosen in preference to the related cosine transform [3]. However, a few points about the relative advantages of each transform are pertinent. Firstly, it has been shown by Yip and Rao [373] that the discrete sine transform can be computed more efficiently than the discrete cosine transform, which would be of importance if building a real time system. However, in general, the discrete cosine transform performs better than the discrete sine transform in terms of decorrelating a signal based on a first order Markov random process [374], and as such has become a common method for image and signal compression [123, page 144]. In light of these facts, it would be interesting to evaluate the effectiveness of the discrete cosine transform in terms of recognition performance, and compare it to the other attribute sets discussed here. This has not been done, however, and remains a possible area of future work.

#### 2.4.1.6 Hadamard Descriptors

The cumulative curvature function for shapes with mostly straight edges (which are overwhelmingly evident in the current data set) are predominantly step-like in nature. That is, they are characterised by curves which are constant in all areas except near corner points, where sharp discontinuities occur. One transform which uses rectangular wave forms as opposed to sinusoid shaped wave forms is the Hadamard Transform. A good description of the Hadamard Transform is given in Pratt [267, pages 209-212]. The Hadamard transform can be used to perform the decomposition of a function into a set of rectangular waveforms and is defined by the following equation:

$$H(u) = \frac{1}{N} \sum_{k=0}^{N-1} h(k) (-1)^{q(k,u)} \quad (2.17)$$

where

$$q(k,u) = \sum_{i=0}^n g_i(u) j_i \quad (2.18)$$

and

$$g_0(u) = u_{n-1} \quad (2.19)$$

and

$$g_l(u) = u_{n-l} + u_{n-l-1}; \quad l > 0 \quad (2.20)$$

and  $x_i$  refers to the  $i^{th}$  bit of the binary representation of  $x$ . The inverse Hadamard transform is also computed using equation 2.17.

### 2.4.2 Binary Attributes

Although the unary attributes are used to discriminate between different types of regions, quite often it is the relationships between the regions which identifies a particular class or structure. In this implementation, we have used relative position of one region with respect to the other, relative orientation of the major axes, relative size of the two regions, maximum 2D distance between the two regions, percentage overlap of the boundaries of the two regions, maximum and minimum 3D distance between the two regions, and 3D angle difference.

The relative position of one region with respect to the other is calculated in the following manner:

$$u = \frac{(\vec{p}_3 - \vec{p}_1) \cdot (\vec{p}_2 - \vec{p}_1)}{|\vec{p}_2 - \vec{p}_1|^2}, \quad (2.21)$$

and

$$v = \frac{(\vec{p}_3 - \vec{p}_1) \cdot (\vec{p}_2 - \vec{p}_1)^\perp}{|\vec{p}_2 - \vec{p}_1|^2}, \quad (2.22)$$

where  $\vec{p}_1, \vec{p}_2, \vec{p}_3, \vec{p}_4$ , are the centroid and boundary points along the major axis of the two regions and are displayed in Figure 2.8, and  $(\vec{p}_2 - \vec{p}_1)^\perp$  is  $(\vec{p}_2 - \vec{p}_1)$  rotated by  $90^\circ$  in the image plane. The denominators in each of the equations normalises the measures to be invariant to size changes. The second binary attribute is the relative orientation of the two regions, and is calculated using the relative orientations of the major axes in the following manner:

$$\theta = \cos^{-1} \left( \frac{(\vec{p}_2 - \vec{p}_1) \cdot (\vec{p}_4 - \vec{p}_3)}{|\vec{p}_2 - \vec{p}_1| |\vec{p}_4 - \vec{p}_3|} \right). \quad (2.23)$$

These three attributes were used because they completely characterise the major axes of the two regions (except for relative size), and were shown by Burns et al. [49] to be relatively invariant over a large range of 3D affine transformations. The relative size of the regions is also affine invariant and is calculated simply as:

$$r = \frac{\mathcal{A}(R_1)}{\mathcal{A}(R_2)}, \quad (2.24)$$

where the function  $\mathcal{A}$  indicates the area of the region and is represented in this implementation as the number of pixels belonging to a region. Finally, the percentage overlap of the two boundaries is calculated using the following scheme:

$$o = \frac{\mathcal{L}(\mathcal{P}(R_1) \cap \mathcal{P}(R_2))}{\mathcal{L}(\mathcal{P}(R_1))}, \quad (2.25)$$

where  $\mathcal{P}(R)$  is the boundary of a region, and  $\mathcal{L}(P)$  is the length of a curve and is approximated, here, by the number of pixels in the curve. This attribute gives a measure of the connectedness of two regions and is potentially very important in distinguishing between differing objects.

The final three binary attributes (whose effectiveness will be discussed in Section 2.7) rely on information extracted using the stereo algorithm described in Section 2.3.2. For each part/region in an image, the stereo data relevant to that region is mapped onto the region, including the data on the region's boundary. This procedure produces very sparse depth data for each part, which is somewhat difficult to use in its current form. Given that most of the objects used in the current work consist of planar surfaces (and noting that more complex surfaces can be successfully approximated with plane surfaces), it was decided to fit a plane to the depth data for each region using a least squares fit (see Figure 2.9). The minimum and maximum depth distances were calculated based on this projection—providing

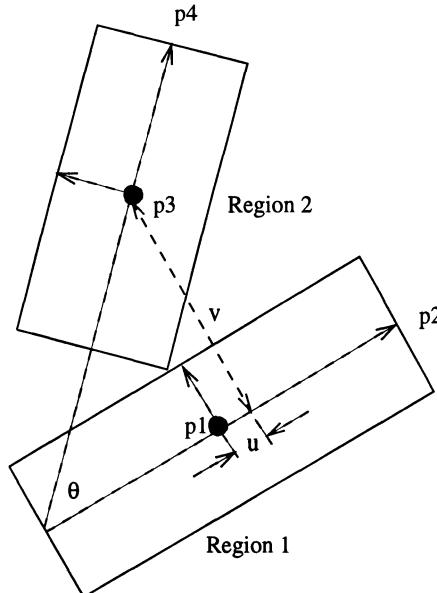


Figure 2.8: Extracting binary attributes from two object parts.

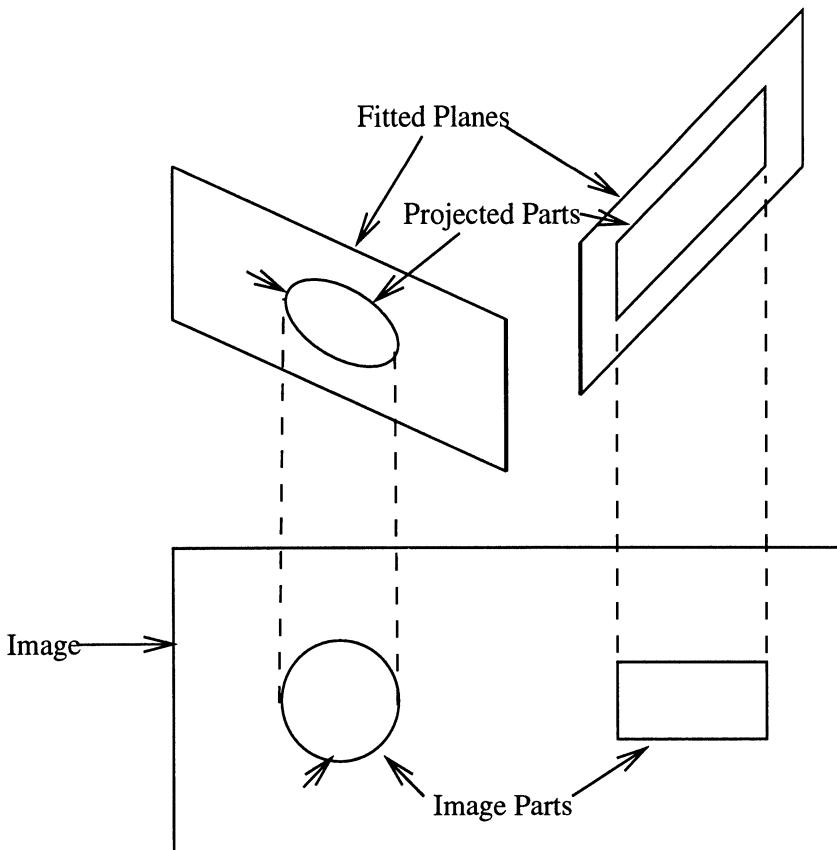


Figure 2.9: An example of two image parts being projected onto planes fitted from the depth data within those regions. The 3D attributes are then extracted from these (parallel) projected parts.

an approximation to the pose of each part. These distances were then normalised by the length of the major axis of the larger region, thus providing measures which are invariant to scenes taken at different depths with the same camera settings. The 3D angle difference was simply calculated as the angle between the normals of the two planes.

The final issue which must be addressed with regard to attribute extraction is that of the range over which the binary relationships are extracted and used for learning and recognition. Currently, a simple scheme has been employed which extracts binary relationships if the minimum distance between any two parts is less than a threshold (currently 10 pixels with respect to a  $256 \times 256$  image format).

## 2.5 The Fuzzy Conditional Rule Generation (FCRG) Classifier

The CRG classifier was introduced by Bischof and Caelli [30] for object recognition purposes. Specifically, it was designed to be an efficient and discriminating classifier which made use of unary and binary attributes. Here, the CRG classifier is further developed to incorporate fuzziness instead of using hard decision boundaries. Fuzzy decisions are extremely useful given the nature of the object recognition problem—especially when attempting to recognise 3D objects from 2D views. The FCRG classifier consists of two phases—the learning phase and the recognition phase which are described below.

### 2.5.1 The Learning Phase

In the learning phase it is assumed that each object is presented separately as a number of training views. All training views are presented serially, but not in a fixed order. The training views are in the form of intensity images from which regions are extracted. The regions are used as the object parts from which unary and binary attributes are extracted.

The FCRG technique begins by grouping unary attributes from all parts, of all views of all objects into a unary feature space  $U$ . Each point in the unary feature space is representative of a single part. The feature space  $U$  is then partitioned into a number of clusters  $U_i$ . Some clusters may be unique with respect to class membership, and these clusters uniquely define which class the corresponding part belongs to. Other clusters will contain multiple classes and these need to be examined further. For the non-unique clusters the binary features between a part and other parts in the view are calculated to form a binary feature space,  $UB_i$ . The binary feature space,  $UB_i$ , is clustered into a number of clusters  $UB_{ij}$ . Again, some clusters will be unique with respect to class membership while others will not. The unary features of the second part,  $p_s$ , are used to form another unary feature space  $UBU_{ij}$ , which is again clustered to form clusters  $UBU_{ijk}$ . This process can continue until all clusters are unique, resulting in clusters at the levels  $UBUB, UBUBU, UBUBUB, \dots$  but, in some cases, clusters may be unresolvable at any level. In this case, the clusters need to be refined at some level, either by reclustering or by splitting clusters. This will be discussed in the next section. An example of the clustering process is shown in Figure 2.10 [30]. Any sequence of clusters from the root to a leaf in the cluster tree is termed a “rule” (e.g.,  $\langle U_1, UB_{12}, UBU_{121}, \dots \rangle$  in Figure 2.10), and any sequence of parts from a single image (view) which index into the cluster tree, is termed a “part path”, or simply a “path” (e.g.  $\langle p_1, p_2, p_3 \rangle$  in Figure 2.10).

The clustering method which is used at each stage is not critical to the

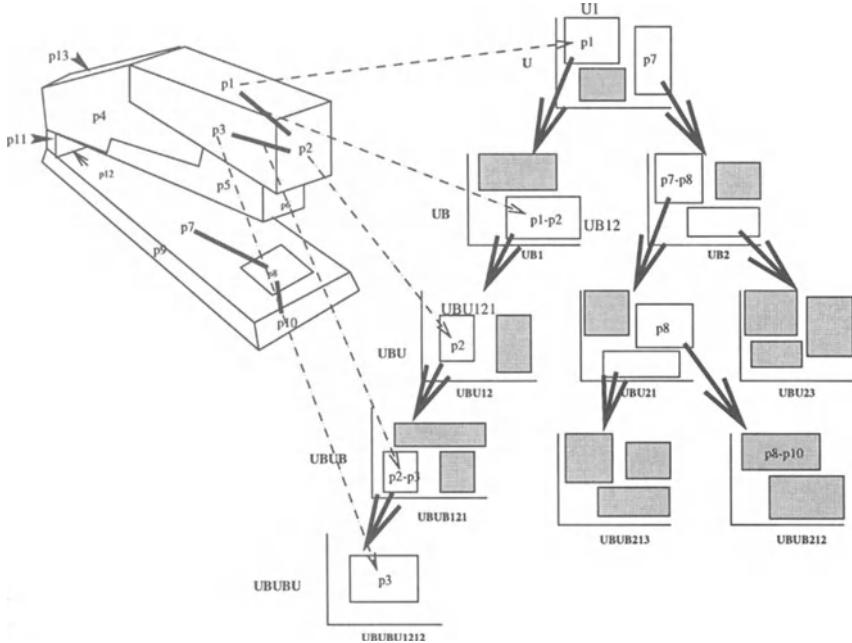


Figure 2.10: The CRG classifier: Displays the tree-like structure of the CRG classifier and the relationship between rules and parts (adapted from [30]). In this figure, grey clusters have zero entropy and are therefore not expanded as class evidence is unique. White clusters are expanded to the next level to gain more evidence to resolve class ambiguities.  $U$  and  $B$  refer to unary (part) and binary (part relation) attribute spaces, the boxes refer to attribute bounds on training data and correspond to least generalisation predicates. Each tree branch corresponds to a “rule” which maximally evidences a class (see text).

success of the FCRG technique, although some clustering methods may work better than others. In this case, we have employed a fuzzy clustering strategy based on a fuzzy K-means algorithm [25]. To determine the number of clusters to use at each stage of the tree building procedure, a cluster validity criterion is used based upon that introduced by Xie and Beni [370]:

$$S = \frac{\sum_{i=1}^c \sum_{j=1}^n \mu_{ij}^f \|V_i - X_j\|^2}{n \sum_{i=1}^c \min_j \|V_i - V_j\|^2 / c}. \quad (2.26)$$

where  $c$  is the number of clusters,  $n$  is the number of data points,  $\mu_{ij}$  is the fuzzy membership of point  $j$  in cluster  $i$ ,  $f$  is the fuzziness factor ( $1 \leq f \leq \infty$ , where  $f = 1$  means hard clustering, and  $f = \infty$  means

maximally fuzzy clustering. Typically, values of  $f$  between 1 and 2 are appropriate),  $V_i$  is cluster centre  $i$ , and  $X_j$  is data point  $j$ . The measure  $\sum_{i=1}^c \sum_{j=1}^n \mu_{ij}^f \|V_i - X_j\|^2/n$  refers to the compactness of the fuzzy partitioning, while the value of  $\min_j \|V_i - V_j\|^2$  is a measure of the separation of the clusters with the function  $\min_j$  indicating the minimum over all  $j$ . We use the fuzzy membership function introduced by Krishnapuram and Keller [169] which is defined as:

$$u_{ij} = \frac{1}{1 + \left(\frac{d_{ij}^2}{\eta_i}\right)^{1/(f-1)}}, \quad (2.27)$$

where  $f$  is defined as before,  $d_{ij}$  is the distance between point  $j$  and cluster  $i$ , and  $\eta_i$  determines the distance at which the membership value for a point in a cluster becomes 0.5. In the current implementation,  $\eta$  has been arbitrarily set to equal 1.

To utilise the cluster validity criterion, the attribute space is clustered using a number of different values of  $K$ , and the final clustering is chosen such that the validity criterion is maximised. This is an expensive procedure, but it reduces the need for performing cluster refinement which is typically also expensive.

An important issue in building the cluster tree is deciding when tree expansion should cease and cluster refinement be employed. Cluster refinement must be considered because tree expansion may never lead to unique clusters, and also reduces the probability of clustering partial data, which is an extremely important consideration with respect to 3-D object recognition. However, cluster refinement leads to a more complex feature space partitioning and a reduction in generalisation ability of the cluster tree—analogous to adding additional hidden units in neural networks. Clearly, a balance between the two techniques must be achieved. Bischof and Caelli [30] perform tree expansion up to a maximum level and non-unique clusters are resolved by splitting previous clusters along attributes which will contribute maximally to decreasing entropy.

### 2.5.1.1 Cluster Refinement

Cluster refinement can be performed in a number of ways. One method is to group the feature space into a larger number of clusters. However, sibling clusters of the unresolved cluster are lost, and must be recomputed, which is inefficient. Another method involves cluster splitting which has the advantage of maintaining the integrity of sibling clusters.

To evaluate the best cluster to split, Bischof and Caelli [30] utilised the cluster entropy measure,  $H$ :

$$H_i = - \sum_j q_{ij} \ln q_{ij} \quad (2.28)$$

where  $q_{ij}$  defines the probability of elements of cluster  $i$  belonging to class  $j$ , and is typically represented by the relative frequency of parts from class  $j$  appearing in cluster  $i$  (each data point is temporarily assigned to a hard cluster for calculating entropies). Using the concept of cluster entropy, partition entropy,  $H_P(T)$ , can be defined as [202]:

$$H_P(T) = (n_1 H(P_1) + n_2 H(P_2))/(n_1 + n_2) \quad (2.29)$$

where  $H(P_1)$  and  $H(P_2)$  are defined by Equation 2.28, and  $n_1$  and  $n_2$  refer to the number of points in clusters  $P_1$  and  $P_2$  respectively. If a leaf cluster is found to be non-unique, then all clusters in the leaf to root cluster path (tree branch) are candidates for splitting. Each of these clusters are reclustered into two distinct clusters using the fuzzy K-means algorithm, and the split entropy of the cluster is calculated according to Equation 2.29. The cluster with the lowest split entropy is chosen as the one to split, and two new clusters are formed. All ancestors of the split cluster must be discarded and the cluster tree must be re-expanded.

A block diagram of the learning stage of the FCRG algorithm is given in Figure 2.11. In summary, FCRG generates part-indexed decision trees where each node is defined by specific parts or relations with associated attribute clusters.

### 2.5.2 The Recognition Phase

Recognition of objects in a scene typically involves locating an object in a scene, which possibly contains multiple objects, and recognising which class that object belongs to. Hence, there are two problems which need to be solved—scene partitioning and object recognition. The FCRG classifier can be used for performing both tasks using the following steps [30]:

1. Unary features are extracted for all scene parts and binary features are extracted for proximate parts.
2. An adjacency graph is extracted from the scene and all non-cyclic paths up to a maximum length,  $maxlength$ , are extracted. The value of  $maxlength$  is equal to the maximum depth of the tree which was generated during the training stage.
3. Each path,  $P = \langle p_i, p_j, \dots, p_n \rangle$ , is classified using the classification tree generated during the learning phase—resulting in a set of classification vectors for each part.
4. The evidence vectors of all paths (evidence paths) starting at  $p_n$  determine the classification of part  $p_n$  and so determine rules for recognition purposes.

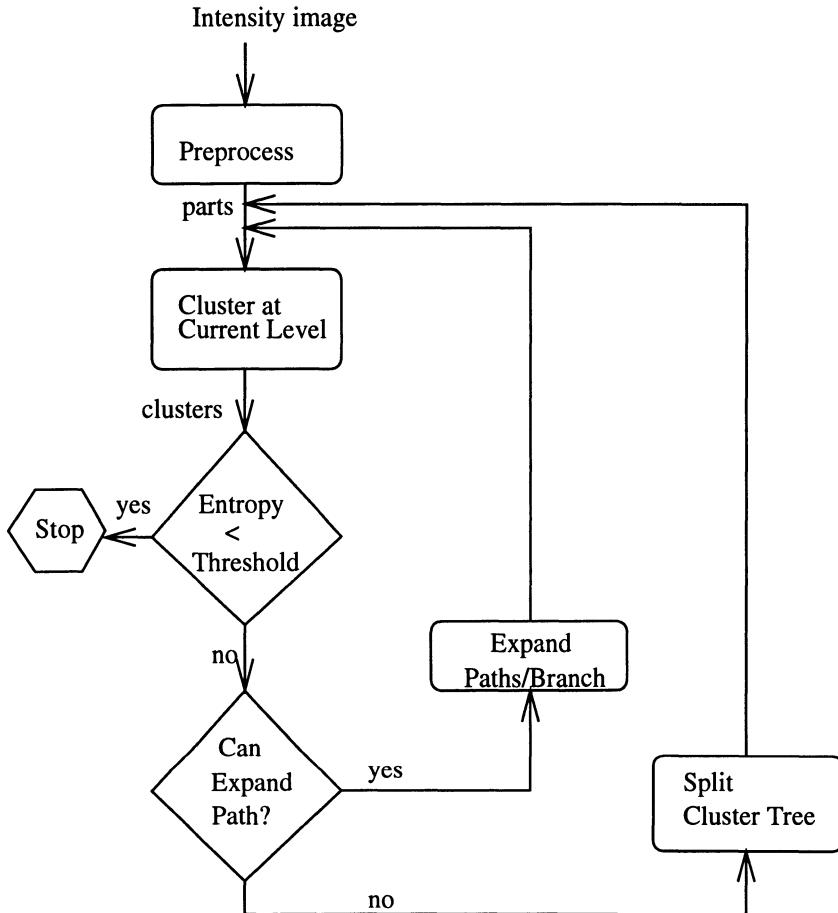


Figure 2.11: The overall learning system.

It is important to note that such rules are “part-indexed” insofar as it is the conjunction of specific parts, relations and their associated attributes, which evidence objects! This is precisely how FCRG and CRG differ from other learning procedures including traditional decision trees and neural networks.

Due to the fuzzy nature of the FCRG tree, each path can be classified in more than one way by choosing the best  $N$  clusters to descend, rather than just one cluster as in the crisp case. In the current implementation, the value of  $N$  given above is 2. This means that at each level of the cluster tree, two clusters are chosen from all the clusters and the path is expanded along both those clusters. The best clusters at a particular level are chosen by evaluating the fuzzy membership function as defined in Equation 2.27.

Finally, not only can we discover the most likely clusters a given path belongs to, we can also extract a measure of how good a given path fits into the cluster tree. At each level in the tree, a path has fuzzy membership associated with each of the clusters at that level in the tree. According to Chang and Pavlidis [60] the decision value of a particular node in a fuzzy decision tree is the product of the decision values of the branches composing the path from the root to the node. Consider a path  $P = \langle p_1, p_2, \dots, p_n \rangle$ , being classified in the decision tree along the cluster branch  $\langle U_i, UB_{ij}, \dots, (UBU\dots)_{ij\dots} \rangle$ , the fuzzy evidence vector for  $P$  is given by:

$$\vec{E}_f(P) = \vec{E}(P) \wedge_{c=U_i}^{(UBU\dots)_{ij\dots}} \mathcal{F}_c(p_c), \quad (2.30)$$

where  $\vec{E}$  is the original evidence vector of path  $P$  characterised by the relative frequencies of each model part in the cluster  $(UBU\dots)_{ij\dots}$ ,  $\wedge$  is the product operator, and  $\mathcal{F}$  is the fuzzy membership function of part  $p_c$  in cluster  $c$  given by Equation 2.27. There are two different interpretations of the product in Equation 2.30—a real number product in the probability model, or as a minimum function in the max-min model [60]. The max-min model has been used here as it is more useful for comparing evidence vectors of instantiated rules, especially if the paths were of different length (using the probability model, shorter paths would typically have higher evidences since there would be less multiplications by numbers less than 1.0 involved in arriving at the final evidence vectors).

Step 4 (see above) deserves further consideration as evidence vectors for a given part may be incompatible, or non-unique. Incompatibility occurs when two evidence vectors indicate two separate class labels for the given part, while non-uniqueness indicates that a rule has been partially instantiated (for example, a path is of type  $UB$ , while rules are of type  $UBU$ ). Further, the problem of how best to utilise such evidence vectors of the paths to provide an optimal and consistent labelling of scene parts is non-trivial. Approaches to this problem fall into two categories. In the first approach, candidate “objects” are selected using perceptual grouping principles and the like (e.g., Mohan and Nevatia [216]). The second approach is the one adopted here which was initially described by Bischof and Caelli [31] and is extended here. In this approach evidence is propagated from parts and relations via compatibility measures, as described below.

### 2.5.3 Compatibility Analysis

Bischof and Caelli [31] have described four rules for updating part evidence vectors by eliminating “crossing” part chains. These are the chain permutation constraint, the single classification constraint, inter-chain compati-

bility analysis, and intra-chain compatibility analysis. The first two rules are deterministic rules, and the last two are probabilistic.

### 2.5.4 Chain Permutation Constraint

Each CRG rule has a set of model or learning parts which are associated with it. The chain permutation constraint is based on the fact that permutations of the same scene chain should instantiate the same model parts. For example, if we consider two chains  $S_1 = \langle p_1 p_2 \rangle$  and  $S_2 = \langle p_2 p_1 \rangle$ , then in chain  $S_1$ , part  $p_1$  will instantiate model parts at the  $U$  level ( $M(U|p_1)$ ), and part  $p_2$  will instantiate parts at the  $UBU$  level ( $M(UBU|p_2)$ ). Similarly, in chain  $S_2$ , part  $p_2$  will instantiate model parts at the  $U$  level, ( $M(U|p_2)$ ), and part  $p_1$  will instantiate model parts at the  $UBU$  level, ( $M(UBU|p_1)$ ). In the chain permutation constraint, the chains  $S_1$  and  $S_2$  are considered valid iff  $M(U|p_1) \cap M(UBU|p_1) \neq \phi$  and  $M(U|p_2) \cap M(UBU|p_2) \neq \phi$ , where  $\phi$  is the empty set. If either of the intersections are empty, then both of the chains are invalid.

Although theoretically sound, the practical applications of such a constraint are limited especially in the domain of 3D object recognition from 2D intensity views. The major reason for this is that it is difficult to tell if the same model parts have been instantiated unless an explicit correspondence is given between model parts in different views. In practice, it will only be possible to determine if the same model part from the same training view has been instantiated. Hence, if two chain permutations instantiate the same model parts, but from different training views (which is quite likely in the current domain), then these chains will be wrongly rejected, and this will adversely affect the recognition performance of the system.

### 2.5.5 Single Classification Constraint

The single classification constraint is based on the assumption that at least one chain starting at a scene part does not cross an object boundary and that at least one instantiated rule indexes the correct model parts. This constraint is somewhat difficult to apply given the possibly large number of chains starting at each part. However, if there exists a part that initiates only a single chain,  $S_i$ , and this chain instantiates a single rule, then the constraint can be applied to all chains which touch  $S_i$ . The constraint is applied in a similar manner to the chain permutation constraint. That is, all chains touching  $S_i$  must instantiate the same set of model parts as  $S_i$ , or it is considered to be inconsistent with  $S_i$  and is removed.

Since this constraint is applied in the same manner as the chain permutation constraint, it suffers from the same problem as described in Section 2.5.4. However, it also suffers from a further limitation. If the assumption upon which the constraint is based is not true, then the constraint cannot

be applied. In other words, if the single classification chain is incorrectly classified, then the recognition performance will be adversely affected. In fairly complicated scenes, however, it is unlikely that a part will initiate a single chain only, and hence this constraint seems to have little effect on the overall recognition performance.

### 2.5.6 Inter-chain Compatibility Analysis

The inter-chain compatibility analysis is based on the idea that the less compatible the evidence vector of a chain,  $S_i$ , is with the evidence vectors of all touching chains, the more likely it is that  $S_i$  crosses an object boundary. A measure can again be formulated based on the model parts which are instantiated by the chains. Let  $S_i = \langle p_{i1}, p_{i2}, \dots, p_{in_i} \rangle$  and  $S_j = \langle p_{j1}, p_{j2}, \dots, p_{jn_j} \rangle$  be touching chains (i.e.,  $p_{ik} \equiv p_{jl}$ , for some  $k, l$ ), and let  $T_{ij}$  be the set of scene parts common to  $S_i$  and  $S_j$ , i.e.,  $T_{ij} = \{p | \exists k, p \equiv p_{ik} \text{ and } \exists l, p \equiv p_{jl}\}$ . The compatibility of  $S_i$  and  $S_j$  is defined as:

$$C(S_i, S_j) = \frac{1}{\#(T_{ij})} \sum_{p \in T_{ij}} \frac{\#(\mathcal{M}(p|S_i) \cap \mathcal{M}(p|S_j))}{\#(\mathcal{M}(p|S_i) \cup \mathcal{M}(p|S_j))} \quad (2.31)$$

Where the function  $\#$  defines the cardinality of a set and  $\#(T_{ij}) > 0$ . The overall inter-chain compatibility of chain  $S_i$  is then defined with respect to the set of all chains touching  $S_i$ ,  $S_T = \{S_j | \#(T_{ij}) > 0\}$ , as follows:

$$w_{inter}(S_i) = \frac{1}{\#(S_T)} \sum_{S \in S_T} C(S_i, S). \quad (2.32)$$

Equation 2.32 can then be used to form the initial average evidence vectors of parts in the following way:

$$\vec{E}(p) = \frac{\sum_{s \in S_p} w_{inter}(S) \vec{E}(S)}{\sum_{s \in S_p} w_{inter}(S)}, \quad (2.33)$$

where  $S_p$  is defined as the set of all chains starting at part  $p$ .

Again, this measure is theoretically sound, but since it, too, uses the model parts which are instantiated, it is also limited by the same problems as described above. Namely, it is difficult to tell if two given model parts are the same part unless they are in fact the same part from the same training view. Also, since it is likely that a significant amount of chains will cross object boundaries in reasonably complex and cluttered scenes, then it is also likely that the value calculated from Equation 2.32 will be low regardless of whether the current chain crosses an object boundary or not.

### 2.5.7 Intra-chain Compatibility Analysis

The final rule for detecting chain crossings is based on the idea that for chains which do not cross object boundaries, the evidence vectors of the parts belonging to the chain should be similar, and dissimilar for chains which cross object boundaries. More formally, given a chain  $S_i = \langle p_{i1}, p_{i2}, \dots, p_{in} \rangle$ , the similarity of all intra-chain evidence vectors can be given by:

$$w_{intra}(S) = \frac{1}{n(n-1)} \sum_{k=1}^n \sum_{\substack{l=1 \\ l \neq k}}^n \vec{E}(p_{ik}) \cdot \vec{E}(p_{il}). \quad (2.34)$$

Since this measure does not make use of compatibilities between instantiated model parts, it does not suffer from the same problems as the other three compatibility rules. However, this rule is rather weak for finding a consistent set of labelled parts.

### 2.5.8 General Comments and a New Compatibility Measure

As has been mentioned, the first three constraints (Sections 2.5.4, 2.5.5, 2.5.6) described above have theoretical merit, but are limited in their practical application due to the problem of finding correspondence between the same model parts in different training views. Hence, model part correspondence is limited to the trivial correspondence between a model part in a particular training view, and itself. The rules may still be useful if part extraction is particularly stable and especially if a model-based recognition system was used. In attempting to apply these rules to non-synthetic data in the current learning system, it was found that very few chains remained after all inconsistent chains were deleted. In fact, many parts could not be recognised at all because of this factor. Hence, these rules were not used in the current object recognition system.

The fourth constraint (Section 2.5.7) for chain compatibility analysis does not suffer from the same problem as it does not rely on correspondence between instantiated model parts, and hence is more applicable to the problem domain. However, the usefulness of this compatibility measure is somewhat limited. This can be attributed to the fact that each chain evidence vector is weighted by a single multiplicative factor which depends only on the compatibility of the average evidence vectors of each part in the chain. It does not depend on the compatibility of the chain's evidence vector with the average evidence vectors of the parts, resulting in the weighting of chain evidence vectors independently of how well the chain represents the overall evidence for those parts. This is best described by an example. Consider the chain  $S = \langle A, B, C \rangle$  in a two class problem,

with average evidence vectors of each of the parts:

$$\vec{E}_{av}(A) = (0.5, 0.5)^T, \quad (2.35)$$

$$\vec{E}_{av}(B) = (0.8, 0.2)^T, \quad (2.36)$$

$$\vec{E}_{av}(C) = (0.9, 0.1)^T, \quad (2.37)$$

and the evidence vector for  $S$ :

$$\vec{E}(S) = (0.2, 0.8)^T. \quad (2.38)$$

The intra-chain compatibility measure for  $S$ , according to Equation 2.34 is:

$$w_{intra}(S) = \frac{1}{3(3-1)} \left[ \begin{array}{c} \vec{E}_{av}(A) \cdot \vec{E}_{av}(B) \\ \vec{E}_{av}(A) \cdot \vec{E}_{av}(C) \\ \vec{E}_{av}(B) \cdot \vec{E}_{av}(C) \end{array} + \right] = 0.29. \quad (2.39)$$

This compatibility measure is quite high, and it should be obvious that the average evidence vectors for each of the parts are compatible. However, the compatibility is used for weighting the evidence vector of  $S$  which is clearly incompatible with the evidence vectors of the parts  $B$  and  $C$ . The result of this is that the updated evidence vector for part  $A$  will probably tend to favour class 2, whereas it is more likely to belong to class 1.

So the current intra-chain compatibility measure does not capture enough information about the compatibility between the chain's evidence vector and the evidence vector's of its constituent parts. A more salient description of the compatibility between parts of a chain would involve a measure of the compatibility of each class of the evidence vector's of the constituent parts with the evidence vector of the chain. In other words, the measure would be a vector quantity and not simply a scalar quantity. More formally, this measure can be characterised by the following equation, for a chain  $S_i = \langle p_{i1}, p_{i2}, \dots, p_{in} \rangle$ :

$$\vec{w}_{intra}(S_i) = \frac{1}{n} \sum_{k=1}^n \vec{E}(p_{ik}) \quad (2.40)$$

This compatibility measure can be used directly with a relaxation labelling scheme:

$$\vec{E}^{(t+1)}(p) = \Phi \left[ \frac{1}{Z} \sum_{S \in S_p} \left( \vec{w}_{intra}^{(t)}(S) \otimes \vec{E}(S) \right) \right], \quad (2.41)$$

where  $Z$  is a normalising factor:

$$Z = \sum_{S \in S_p} w_{intra}^{(t)}(S), \quad (2.42)$$

and the binary operator  $\otimes$  is defined as a component-wise vector multiplication in the following way:

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ bd \end{pmatrix}. \quad (2.43)$$

The function  $\Phi$  is a function mapping the interval  $[0..1]$  to itself and is often sigmoidal in nature to accelerate convergence of the relaxation labelling process. However, such acceleration typically worsened recognition, and a linear identity function was used in its stead.

Redoing the above example gives the following compatibility measure:

$$\vec{w}_{intra} = \frac{1}{3}[\vec{E}_{av}(A) + \vec{E}_{av}(B) + \vec{E}_{av}(C)] = (.73, .26)^T. \quad (2.44)$$

As can be seen from the resultant compatibility measure, chain  $S$  is more likely to belong to class 1 than class 2, even though the initial evidence vector suggests otherwise. So, when the component-wise multiplication is performed in Equation 2.41, the evidence used due to chain  $S$  is minimised because of the apparent incompatibilities and becomes less significant as more iterations are completed.

This measure for calculating compatibilities has proved very successful for finding a more consistent set of part labels for each of the scene parts. It should be noted that the measure can be used in conjunction with the other three compatibility rules if desired. However, due to the unsuccessful application of these rules, it was found to work very well with the relaxation labelling scheme alone.

### 2.5.9 The Overall Recognition Algorithm

A block diagram of the recognition stage of the FCRG algorithm is displayed in Figure 2.12. FCRG and other machine learning approaches to rule generation and recognition can be viewed as ways of “pre-compiling” search strategies for the verification of models in data. For this reason we have also considered a final hypothesis verification stage to resolve the types of ambiguities remaining. In all, then, the result of the FCRG classifier is a set of most likely labels for each part of the scene where the labels refer to a given object, its sample and sample part from the training data which can be used to infer the pose of the object!

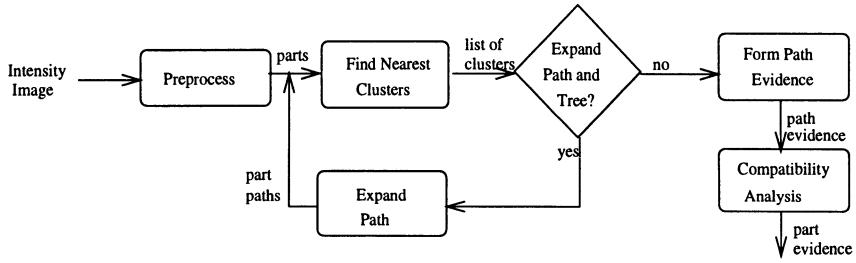


Figure 2.12: The overall recognition system.

## 2.6 Hypothesis Verification

A simple hypothesis verification scheme has been used here to demonstrate the usefulness of the FCRG algorithm for this task. It should be noted that this scheme is not intended to be a practical system, it is intended only for demonstrating the applicability of the FCRG algorithm as an hypothesis generation method.

The hypothesis verification system is basically a correlation based template matching approach. For each of the parts in the scene, the most likely classes are found using the FCRG algorithm. In this case the number of classes used was set to two. These classes are then matched against the parts using a correlation based scheme, and the class which returns the best match for that part is selected. It was found that the classical measure of correlation in the computer vision literature—Pearson’s correlation coefficient [123] - was particularly sensitive to noise in the images. For this reason a more robust measure of correlation was used, which exhibits a lower sensitivity to noise, based on the  $L_1$  norm [44]:

$$g(x, y) = 1 - \frac{\sum_i |x_i - y_i|}{\sum_i |x_i| + |y_i|} \quad (2.45)$$

where  $x$  and  $y$  can be interpreted as the intensity values for the candidate model pose and scene image respectively, when the regions in question are placed on top of one another.

The question remains about how to determine the most likely pose of the scene object to maximise the correlation between model and scene, and how to perform this actual matching. Theoretically, the results from the FCRG algorithm give pose information about the scene objects. Each cluster in the FCRG tree contains information about the training parts that belong to the cluster. From this data, it is possible to determine the most likely object and pose of a particular scene part. For CRG and FCRG, each scene part can be involved in a number of different clusters at many different levels of the tree—due to the likelihood that scene parts are involved in

more than one rule and at different parts of each path. This problem is further compounded by the fuzziness of the FCRG tree which allows a single path to belong to numerous clusters—possibly two child clusters for each parent cluster in the current implementation. Because of the large number of clusters each scene part can be involved in, significant confusion can arise over which model pose is the most likely, making it difficult to accurately perform correlation matching based on this information. Since the number of model poses is small in the current application (typically 6 poses with corresponding left and right views), and utilising the fact that left and right views of a given pose are very similar, the correlation matching for a scene was performed using only the left model views of the two most likely classes, and classifying each part based on this correlation. This was performed for the two most likely classes according to the FCRG recognition algorithm. The correlation is calculated over the entire model view, affecting the recognition of all scene parts which touch the model view.

Obviously, a more efficient and robust hypothesis verification system, such as Bunke and Messmer's graph matching technique [48], would be required. Nevertheless, the current system is adequate for displaying the usefulness of the FCRG learning technique.

## 2.7 Results

A number of experiments have been performed and these are described here. Many of the experiments utilise the single object scenes to evaluate different aspects of the FCRG classifier and were performed using either a cross-validation or train/test set methodology.

### 2.7.1 Comparison of CRG and FCRG

Firstly, it is useful to establish if FCRG does in fact perform better than CRG. To test the relative performance of each, we use a simplified version of the data set described in Section 2.3. The general shape descriptors and mean and variance of intensity were used as the unary attributes, and the binary attributes without depth were used. No hypothesis verification is performed with either classifier, as we just want to test how each classifier performs relative to the other. The CRG classifier achieved a recognition performance of 56%, compared to 72% for the FCRG classifier. The performance of the CRG classifier used here was the best performance found whilst optimising over the input parameters. The parameters used were to set the minimum entropy per cluster to 0.1 (i.e., don't split a cluster if the entropy is less than 0.1), and the minimum size of a cluster to 0.3 along all attribute axes (i.e., don't split a cluster along an attribute axis, if the size of the cluster along that axis is less than 0.3 of the total range). The

parameters for the FCRG classifier were a minimum entropy threshold of 0.1, cluster fuzziness of 1.5, and a maximum tree depth of 7. For each test scene the proportion of correct scene regions was calculated (the number of correctly recognised regions divided by the total number of regions), and this was averaged across all test scenes to produce the overall performance. A cross-validation procedure was used to generate these results. The results were evaluated using a statistical *t*-test [203] and were significant at the 1% level.

Apart from recognition performance, it is also useful to discuss other relative advantages and disadvantages between the CRG and FCRG classifier. In general, the CRG classifier is faster than the FCRG classifier in learning and recognition times, although this is dependent on the parameters used for both classifiers. The CRG classifier learns quickly using default parameters, but often does not perform well (only 39% recognition with the above experiment) with these parameters, depending on the input data. The FCRG classifier, on the other hand, tends to perform fairly well using default parameters across a greater range of problems, since it estimates many of the inherent parameters of the problem by itself (e.g. the number of clusters per level, size of clusters etc). So, although CRG is quicker in terms of running time, the user typically has to adjust the CRG parameters and repeat the experiment a number of times, to find a good set of parameters. So, in effect, the CRG classifier has no real advantage over the FCRG classifier in terms of execution time.

### 2.7.2 Evaluating the Unary Attributes

The unary attributes described in Section 2.4 have been evaluated by performing cross-validation tests, again using a simplified version of the data set described in Section 2.3. The binary attributes which were used are irrelevant except to say that the same binary attributes were used for all tests involving the unary attributes. For the results reported here, the binary relationships were calculated between parts which were within 10 pixels (on a 256 by 256 image) of each other, and no hypothesis verification was performed.

Figure 2.13 displays a graph comparing the performance of each of the attribute sets. For the relevant attribute sets, Set 1 in the graph uses the first 16 descriptors, Set 2 the first 8 and Set 3 the first 4, except for the Fourier descriptors which do not make use of the imaginary part of the first descriptor (since its always 0). In the case of the Fourier descriptors, Set 1 makes use of the 15 non-zero attributes and likewise for Set's 2 and 3.

As can be seen from the figure, the Fourier, Hadamard, Sinusoid, and General descriptors all perform approximately the same for the optimal number of descriptors, with the Hadamard descriptors performing the best overall (marginally). It is unclear from this data which would be the best

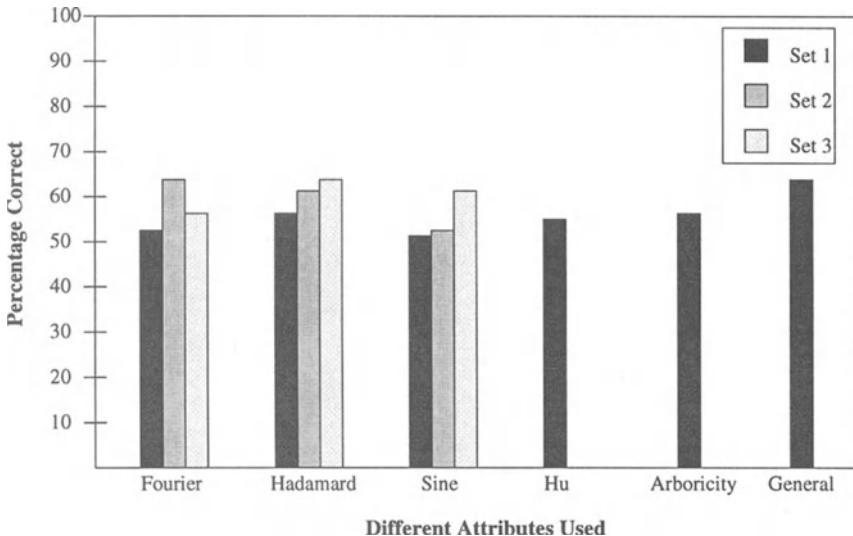


Figure 2.13: Comparison of the performance of a number of attribute sets: For the first 3 distinct attribute sets, Set 1 involves 16 descriptors, Set 2 involves 8, and Set 3 involves 4 descriptors.

set of attributes to use. The Hadamard descriptors have somewhat limited applicability, since they are inefficient at representing shapes with smooth curves. Contrary to popular belief, this experiment shows that Hu invariants are not particularly good at shape representation and recognition, and in fact the invariants are not linearly independent which reduces their usefulness [203].

Taking these points into account, it was decided to use the general shape descriptors since they are the most intuitive, and to augment them with the mean and variance of the intensity inside the region. For the experiments described in the following sections, the full data set described in Section 2.3 was used .

### 2.7.3 Depth Attribute Evaluation

In these experiments we use a full cross-validation procedure to test the effectiveness of the depth attributes. That is, the classifier is trained on all training objects except for one pose (both left and right views) of each class. The accuracy of the classifier is determined by using the pose left out for testing. This is repeated across all poses of each class, except for the first and last pose as these poses are degenerate views ( $0^\circ$  and  $90^\circ$ ), and it is unreasonable to expect the classifier to recognise views which are not

within the learning pose range.

Figure 2.14 shows the results of the depth attribute evaluation experiments both with and without hypothesis verification. As can be seen from the figure there is a small improvement on performance when depth attributes are used without hypothesis verification, but the hypothesis verification procedure nearly equalises out all performance. The results from this experiment prove to be inconclusive in regards to the usefulness of sparse depth attributes. Certainly, the experimental images do not display a great range of depth, but this is typical of the normal operating environment of human object recognition: objects viewed from more than a few metres display very little depth from stereo cues, yet we have little difficulty recognising them. Although the results from this experiment are inconclusive, it does appear that sparse depth attributes obtained from stereo data can be useful for recognition in certain circumstances, but cannot be generally applied in a natural environment. It is likely to be more useful for other applications such as navigation and object position determination.

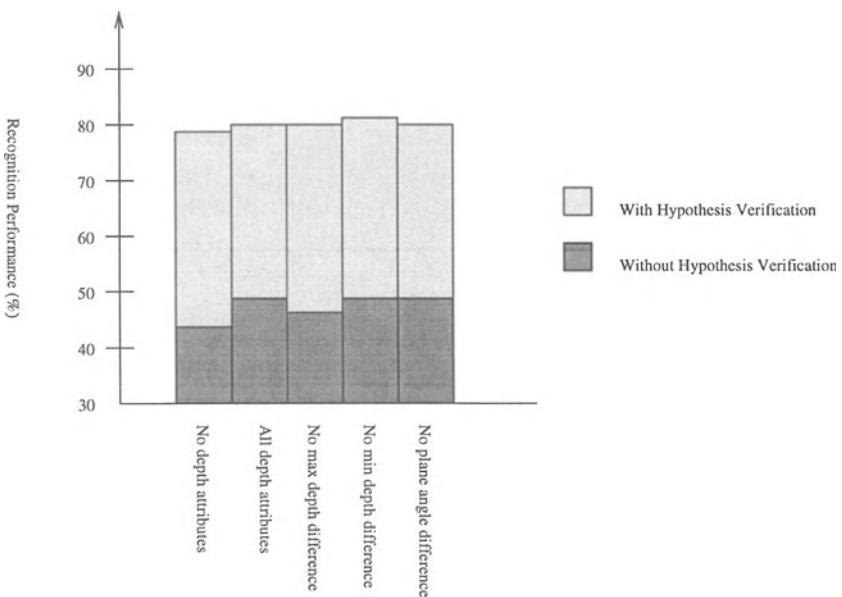


Figure 2.14: Results of the Depth Evaluation Experiments: both hypothesis testing and non-hypothesis testing results are shown

### 2.7.4 Tree Depth Experiments

In this experiment, the maximum depth of the FCRG tree is varied to determine the optimal maximum depth. The experimental methodology involves training the FCRG classifier using three out of the possible six different poses as training examples, and the remaining three poses as test examples, resulting in six training images (a left and right stereo image for each pose), and six test images. The classifier then attempted to recognise each of the poses (training and test) at the recognition stage. The recognition performance on the training examples was performed to test the accuracy of the classifier and the test examples are used to evaluate the generalisation ability of the classifier. Obviously, the ideal situation involves maximising both quantities, but these can be conflicting goals (it is possible to get near perfect accuracy by considering each attribute point as a single cluster, but the generalisation in this case would be poor). These experiments are only indicative of the power of the FCRG system as only 6 examples per class were used for training. Hence, it is the trends in the results reported in this section which are important, and not the overall results.

Three important factors which characterise the performance of any classifier are the recognition performance, the execution time, and the compactness of the final representation. The FCRG classifier is evaluated in relation to all three performance metrics. Also, we show results with and without cluster splitting (i.e., cluster refinement, see Section 2.5.1.1), and with and without cluster searching (see Section 2.5.1). Without cluster refinement, the FCRG tree is only expanded to the maximum allowable depth of the tree without the provision for backtracking and refinement. Without cluster searching, the maximum allowable number of clusters per attribute space is used to initially cluster each space (this may increase if cluster splitting/refinement is turned on) rather than searching for the optimal number of clusters. Here, performance was measured in terms of the percentage of correctly identified parts. This is a conservative measure as correct model projection can be accomplished with quite incomplete part identification.

Figure 2.15 and 2.16 displays the accuracy and generalisation performance of the classifier respectively, whilst varying the (maximum) depth of the tree, and turning cluster splitting and cluster searching on or off. Concentrating firstly on cluster splitting and cluster searching, it is clear that cluster splitting is more important than cluster searching, especially in terms of the accuracy of the classifier. This is not unexpected since cluster splitting is tuned to splitting based on misclassification (actually entropy), whereas cluster searching is performed based on the geometry of the data (good clusters in terms of the shape rather than the contents of the clusters). As can be seen from Figure 2.16, cluster searching or splitting is less important for the generalisation abilities of the classifier, although cluster

splitting still provides an improvement.

In terms of the depth of the tree, it is clear from the figures that the best performance can generally be obtained at depths of two or three. Seemingly, this indicates that most objects in the current data set can be distinguished based on mostly two part paths (unary-binary-unary). We do not claim that this can be generalised to all data sets as more work needs to be done on this problem. However, if this observation is true generally it has quite far-reaching consequences, for example, in object graph-matching schemes. If only two part paths are required to distinguish between most objects, then presumably, the efficiency of graph-matching schemes can be improved using this observation.

Time and space efficiency of the experiments are shown in Figures 2.17 and 2.18. As expected, both time and space requirements increase as the depth of the tree increases. For cluster searching, the time increases rapidly (possibly  $O(N^2)$ ), but for the other experiments, the time increases at a reasonable rate (probably  $O(N)$ ). The size of the tree on the other hand increases most slowly for cluster searching, which is to be expected, since if cluster searching isn't performed, the maximum number of clusters is used. Hence, cluster searching should be used if the size of the tree is a critical

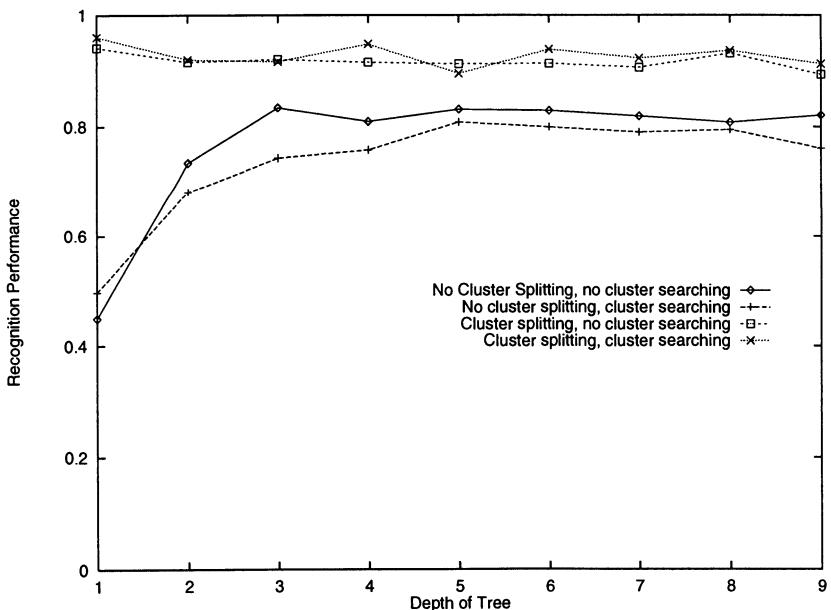


Figure 2.15: Varying the depth of the tree—accuracy performance. The x-axis indicates the depth of the tree ( $1=U$ ,  $2=UB$ ,  $3=UBU$ ) and the y-axis indicates the recognition performance.

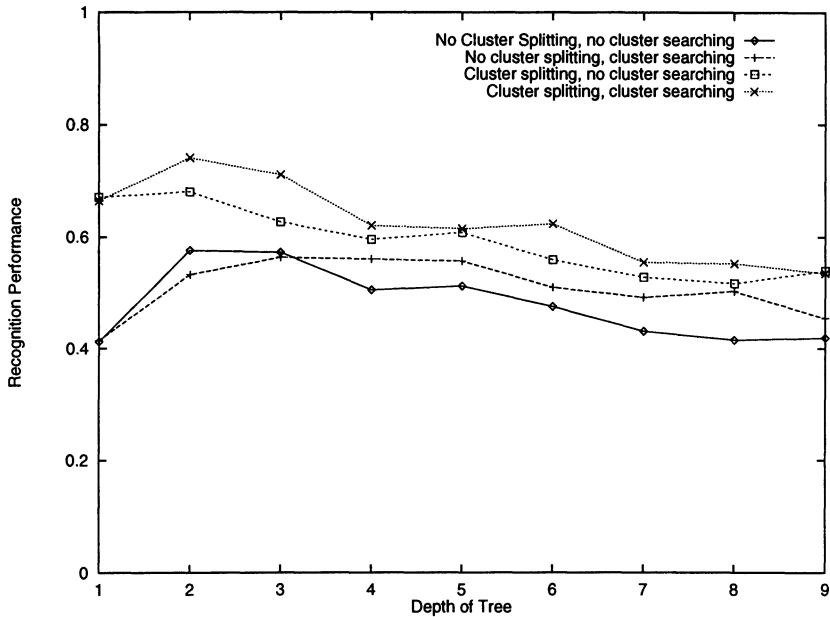


Figure 2.16: Effect of depth of cluster tree on generalisation performance. The x-axis indicates the depth of the tree (1=U, 2=UB, 3=UBU) and the y-axis indicates the recognition performance.

factor, but not if learning time is important.

### 2.7.5 Occlusion Experiments

In these experiments, we demonstrate the effectiveness of FCRG when only partial data is available. The FCRG classifier is trained on the set of all 18 objects, and each training view is progressively occluded in an artificial way and used to test the classifier. This occlusion is performed by blanking out part of the view going from the left to right of the object (i.e., if 50% of the object is occluded, then the left half is blanked out). So the experiment is not entirely realistic, but should give a reasonable indication of the performance in the presence of occlusion.

Figure 2.19 shows the results of the experiments, with between 10% and 90% occlusion of the test views. The system has performed exceptionally well. Even with 60% occlusion of objects, a recognition rate of better than 70% is achieved indicating that the FCRG classifier degrades gracefully as the quality of the input degrades.

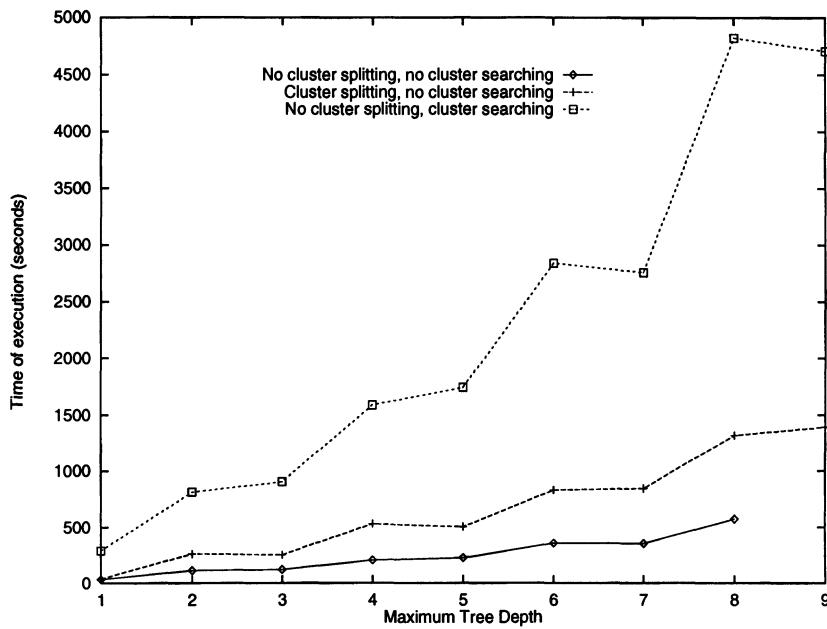


Figure 2.17: Effect of depth of cluster tree on execution time (CPU time).

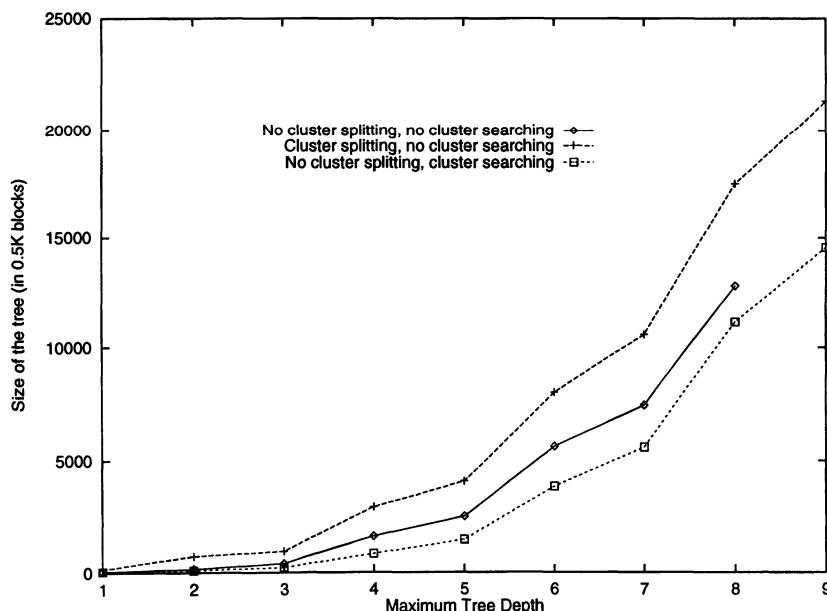


Figure 2.18: Effect of maximum depth of cluster tree on size of tree

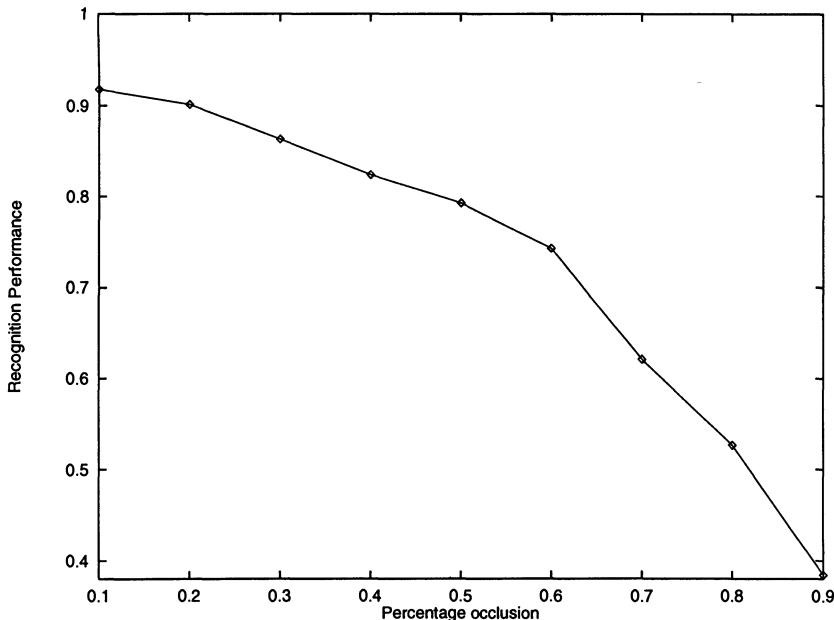


Figure 2.19: Results of the Occlusion Tests: The x-axis indicates the amount of occlusion and the y-axis indicates the recognition performance.

### 2.7.6 Scaling Experiments

In this set of experiments, the number of classes used for training and testing was varied to study how well the system scales as the number of classes increases. The methodology used for this experiment was a type of bootstrap procedure. The number of classes to use was varied between 10 and 17. For each of these tests, the classes were randomly chosen from the set of 18 objects, and learning and recognition was performed using both training (accuracy) and test data (generalisation). This process was repeated 5 times for each number of classes thereby randomising the classes used, and removing any bias obtained by choosing “good” classes. Again, this experiment was exploratory only, so it is the trends that are important and not the overall results.

Figure 2.20 shows that the generalisation of the classifier decreases as the number of classes increases, but the accuracy remains approximately constant. It is likely that the slow degradation in generalisation performance would continue as the number of objects increases, but it would be interesting to see if the accuracy performance would degrade.

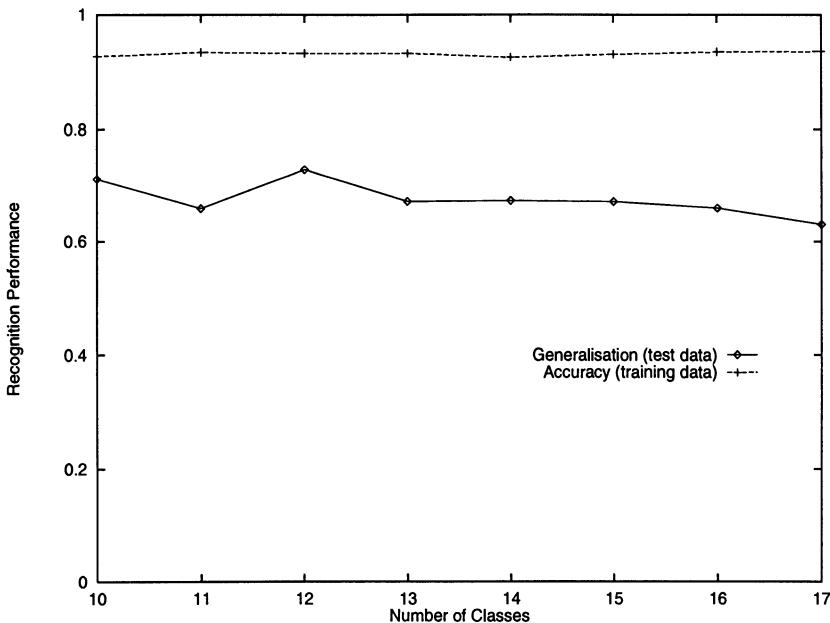


Figure 2.20: Results of the Scaling Tests

### 2.7.7 Multiple Object Scenes

The final experiment involved the use of multi-object scenes. In this experiment, a single scene involving a number of objects was presented for classification. The image was pre-processed in the same way as described in Section 2.3 above. The objects were presented in similar, but not identical, pose to the training data. This problem is considerably more difficult than the single object scenes for a number of reasons:

1. In scenes with overlapping objects, occlusion effects make recognition more difficult.
2. The segmentation of multiple object scenes can lead to single regions which cross object boundaries making it impossible to correctly recognise the whole region.
3. Paths are often formed which cross object boundaries and these cause confusing evidence.
4. There are generally more paths in a multiple object scene which makes recognition more difficult.
5. Two similar objects in a scene can create extra confusion especially if paths cross over these objects boundaries.

6. Specifically related to the current experiments—often objects in a multiple object scene are of different size than the training views. This can cause problems with segmentation (the same object at different scales may be segmented differently) and, without normalisation, can confuse the hypothesis verification stage of the system (which is based on correlation).

As expected, it became increasingly difficult to recognise objects as the number of objects in a scene increased. This was especially true for scenes where the objects were at a greater distance from the camera than for the training data (see problem 6 above). Since the attributes used were scale invariant this is not a problem with the learning system but a limitation with the current hypothesis verification system. For those images which did not display problems of scale, the classifier and hypothesis verification system have worked quite well. Figures 2.21 to 2.24 display the recognition results for a number of different scenes containing various objects. In the images, a flecked pattern indicates a correctly recognised region, while a lattice pattern indicates an incorrectly recognised region.

## 2.8 Conclusion

In this chapter, a system which can learn to recognise 3D objects from 2D intensity images was presented. The system uses a recognition-by-parts approach and encodes 3D objects in terms of rules which describe the properties of object parts and their relations. The FCRG learning algorithm is an extension of work by Bischof and Caelli [30] and, like that system, is focused on the automatic generation of part-indexed decision trees to solve the problem of induction over labelled and attributed graphs. This recognition-by-parts approach is particularly useful for multi-object scenes

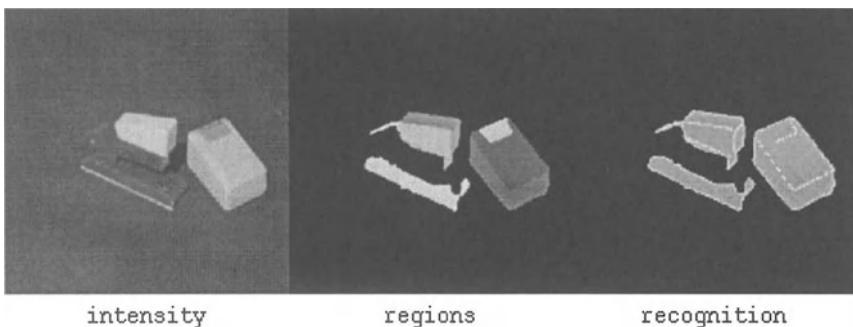


Figure 2.21: The mouse and the stapler

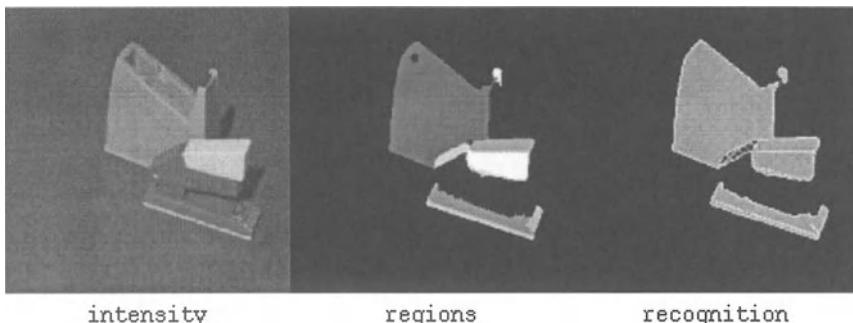


Figure 2.22: The stapler and the sticky tape holder. The back end of the stapler was recognised as part of the sticky tape holder due to the hypothesis verification algorithm

and for recognition problems where recognition is required with partial data, where there is considerable variation of model instances and where recognition-under-transformations (invariance) is required. The system addresses many of the problems inherent in the recognition problem as a whole, rather than ignoring these problems as has been the prevalent approach in the literature. The important features of the system include:

1. The use of real 2D intensity images as input to the system.
2. The input images were segmented using an automatic adaptive multi-scale segmentation procedure. That is, training and test scenes were

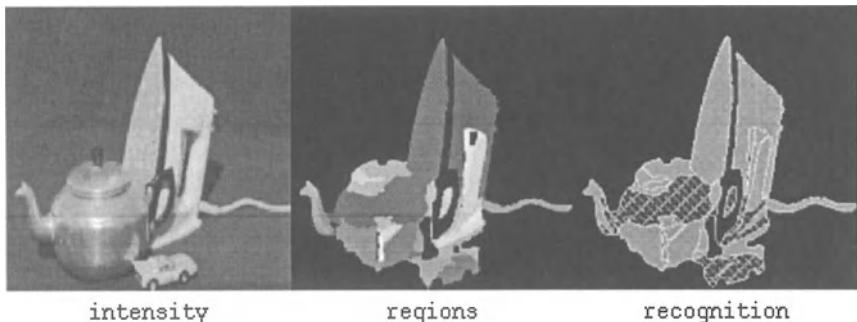


Figure 2.23: The teapot, iron and the model car. The middle section of the teapot was misclassified because of incorrect segmentation and the model car was misclassified due to lack of detail.

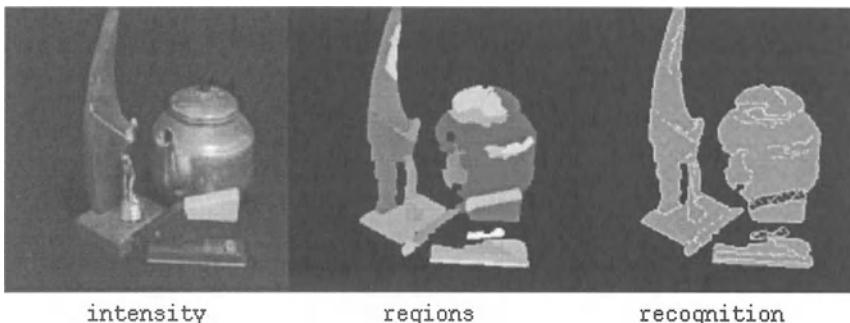


Figure 2.24: The stapler, trophy and the teapot. The top of the stapler was misclassified as belonging to the teapot.

not human or computer generated.

3. A reasonably sized database of 18 objects was used to evaluate the system.
4. Sparse depth data obtained from a depth-from-stereo algorithm was used to aid recognition.
5. Multiple object scenes are handled in exactly the same manner as single object scenes. Hence, it is not necessary to locate separate objects in a scene prior to recognition.
6. The system is a machine learning system, which learns to recognise objects from examples. Hence, objects which are difficult to model can be recognised.

The results obtained by the system are very promising. A recognition rate of 80% was achieved for recognising single-object parts not previously seen. Initial success for the multi-object scenes indicates that the approach used here is valid for recognising 3D objects from 2D intensity images. We have evaluated numerous unary attribute sets and have found that general shape descriptors perform at least as well as other descriptors but have the advantage of being intuitive. We have also shown that the use of sparse depth attributes have little effect on improving recognition performance, and we suggest that perhaps such information is more useful in position determination and navigation problems. We have shown that for this application, only part-paths of up to two parts are useful for recognition. As more parts are used, recognition performance degrades. This result has possible implications for any recognition algorithm which relies on relational data involving part-paths (e.g. graph matching techniques).

There are still areas of the system that can be improved. Most notably, a more comprehensive hypothesis verification sub-system needs to be implemented. The FCRG algorithm could also be improved to include non-uniform shaped clusters (e.g., elliptical) and the minimum message length (MML—Wallace and Freeman [352]) criterion could be used to aid in the tree building algorithm. Also, although it is not necessary to isolate objects in a scene prior to recognition, it would probably improve the performance of the system if this were attempted (although it would not be necessary for the isolation procedure to be perfect). Perhaps the depth data obtained from the depth-from-stereo module could be used to aid in isolation of some objects.

Finally, it has been shown in this paper that recognising complex 3D objects from 2D intensity images is an attainable goal. The problem has been approached from an holistic point of view—that is many of the problems inherent in such a domain have been addressed, rather than just focusing on a small subset of problems, and assuming that all other problems have been solved (or will be in the future). We believe that better results can be obtained by addressing the problem in its entirety.

## Chapter 3

# Relational Evidence Theory and Interpreting Schematics

Adrian R. Pearce and Terry Caelli

### Abstract

A new relational learning algorithm, the Consolidated Learning Algorithm based on Relational Evidence Theory (CLARET) is presented. Here, two different approaches to evidential learning are consolidated in how they apply to generalising within relational data structures. Attribute-based discrimination (decision trees) is integrated with part-based interpretation (graph matching) for evaluating and updating representations in spatial domains. This allows an interpretation stage to be incorporated into the generalisation process. These components of the system are demonstrated in an on-line system for the recognition of hand drawn, schematic diagrams and spatial symbols. The approach uses an adaptive representational bias and search strategy during learning by efficiently grounding the learning procedures in the relational spatial constraints of their application. It demonstrates how inductive rule generation can be constrained by domain knowledge.

### 3.1 Introduction

Schematic interpretation essentially involves labelling planar drawings with respect to domain knowledge and this involves building systems which learn to bind spatial patterns with such knowledge. The efficiency of schematic interpretation systems is measured not only in terms of their ability to learn to classify patterns, but also their computational complexity and their capacity to accommodate different and new patterns.

The focus of this work is on the theoretical and procedural issues involved in applying machine learning to the problems of efficient schematic interpretation. Of specific focus is what we term “Relational Evidence Theory” which integrates information theoretic methods from decision trees with graph matching methods from constraint interpretation. It offers an evidence-based framework for evaluating and updating relational representations suitable for spatial applications. A consolidated learning algorithm based on relational evidence theory (CLARET) is presented which integrates graph matching techniques with rule generation from inductive logic programming. The approach utilises the relational constraints in spatial data to optimise the representational hierarchies and search strategies used during learning.

Interpretation of montages or scenes of different patterns rely on the relative layout and positioning of parts as an important component of the interpretation process. Interpretation hierarchies must include both a notion of scale resolution (what a *part* is) combined with relative location to other parts (*part relationships*). For example, in the identification of mathematical equations, formulae are defined by sets of strokes at different locations and orientations in space (see Figure 3.1).

Schematic interpretation is representative of many real-world image interpretation tasks which involve spatial patterns arranged in different ways. Although the feasibility of schematic interpretation has been proven, current techniques are yet to be optimised with respect to their outcomes and the human factors involved. Surveys which consider handwritten recognition can be found in Mori, Suen and Yamamoto [218]; Pavlidis and Mori [246]; and Plamondon [258].

Recognition of mathematical equations, schematic diagrams, Chinese characters and robust text recognition require feature attributes and rules which are invariant to rotation, scale and shift. Current systems do not typically have high degrees of invariance and, in general, do not fully exploit machine learning models [218, 246, 258].

In this work, an on-line schematic and symbol recognition application is demonstrated for learning to recognise symbols and patterns invariant to rotation, scale and shift. The classification performance, computational efficiency and the human factors involved in incrementally training the system are empirically compared with other inductive logic programming techniques.

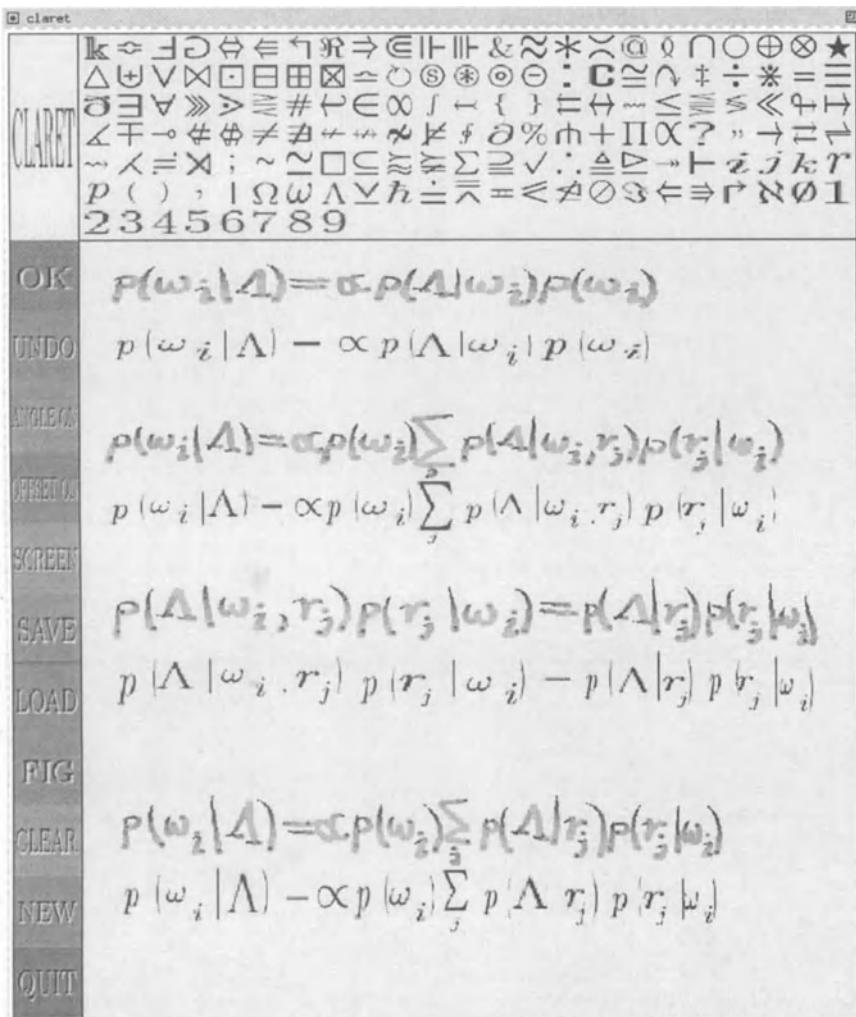


Figure 3.1: “Scientific Symbols” application: The application window is shown comprising of 128 scientific symbols from the American Mathematical Society package for Latex [124]. Symbols are drawn on the central canvas and matched interpretations are back-projected by calculating orientation parameters (shown here using vertical offset). The system works in recognition mode, and the user inputs the identity of unseen and misclassified examples by selecting the symbols tiled at the top of the application window.

## 3.2 Recognition by Parts

In schematic interpretation, images are typically segmented into region- or edge-based descriptions [241]. These descriptions are then indexed using either attribute-indexed representations, such as decision trees and neural networks, or using part-indexed representations, such as relational structures. These two representational schemes are described below.

### 3.2.1 Attribute-indexed representations

Attribute-indexing relies on measures such as information statistics [252] to partition (split) data based on expressions involving feature attribute values. For example, decision tree representations can be generated through a process of attribute learning and conjunctive addition of feature constraints [270, 273] resulting in propositions of the form:

$$\begin{aligned} \text{feature\_state}_i \leftarrow \text{feature\_attribute}_1 &\geq \text{attribute\_value}_1, \\ \text{feature\_attribute}_2 &\leq \text{attribute\_value}_2, \dots \end{aligned}$$

where  $\text{feature\_state}_i$  implies the existence of  $\text{pattern}_i$  if  $\text{feature\_attribute}_i$  is within  $\text{attribute\_value}_i$  ranges. For example, the rule,

$$\begin{aligned} \text{equals\_state} \leftarrow \text{num\_strokes} &\geq 2, \text{num\_strokes} \leq 2, \\ \text{aspect\_ratio} &\geq 1.0, \text{aspect\_ratio} \leq 2.0 \end{aligned}$$

is evidence for the symbol ‘=’.

Pure attribute-indexed learning systems are limited in efficiency with respect to their classification ability. The propositional representation fails to encompass the dependency of the different spatial components when there are multiple parts contained in patterns [250].

### 3.2.2 Part-indexed representations

Part-indexing relies on the associations between different components or *parts* which are captured by *labelling* specific instantiations of feature states. Relational structures thus refer to the representation of patterns or shapes in terms of indexing over attributed, labelled parts and their relationships [311]. Both unary (vertices in the graph representation) and binary (edge) features are typically included in the feature lists.

**Definition 1** A relational structure is defined by labelled parts (vertices)  $\{\lambda_1, \lambda_2, \dots, \lambda_k\}$  which have unary feature states,  $u(\lambda_i)$  and binary feature states (edges),  $b(\lambda_i \lambda_j)$ .

These associations can be represented as labelled first order rules,

**Definition 2** Labelled first order rules are defined by unary  $u_i(\lambda_I)$  and binary  $b_j(\lambda_I \lambda_J)$  terms,

$$r_j \leftarrow u_1(\lambda_I), b_1(\lambda_I \lambda_J), u_2(\lambda_J), b_2(\lambda_J \lambda_K), \dots$$

where pattern<sub>i</sub> is implied by the existence of  $r_j$ ,  $\lambda_I, \lambda_J, \lambda_K, \dots$  are label variables and  $u_1, b_5, \dots$  refer to attribute bounds.

Local information or feature attributes (unary features such as length) need to be extracted as well as relationships (binary features such as angles). For example, the equals sign '=' could be represented by the rule,

$$\text{equals} \leftarrow \text{long(stroke}_1\text{)}, \text{parallel\_to(stroke}_1, \text{stroke}_2\text{)}, \text{long(stroke}_2\text{)}.$$

In three-dimensional object recognition, for example, an object can be described by features characterising surface parts (unary features) such as average curvatures or boundary shape descriptors and by features describing part relationships (binary features) such as centroid distance or mean normal-angle differences. However, these part and part-relation features have to be linked together into a relational structure in order to define patterns uniquely.

Recognition takes place via the instantiation of rules which involves binding of label variables from unary and binary terms with labelled feature states. In doing so it is necessary to check the compatibility between the instantiations of unary and binary rules (see Figure 3.2(a)).

This representation, and the associated graph matching, has been the preferred architecture for object recognition—in the form of interpretation trees and feature indexing [109]. Other methods for generating relational structures for visual recognition include constraint-based decision trees [126], pre-compiled tree generation [148], heuristic search techniques

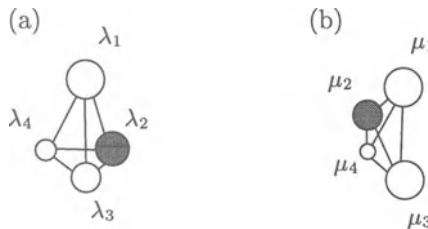


Figure 3.2: Isomorphism example: In (a) a labelled and attributed graph (relational structure) is shown. Global label compatibility checks (subgraph isomorphism) are required to differentiate between this graph and the graph in (b) as their unary and binary feature states, in isolation, are identical (unary features are size and colours, binary features are distance and colour difference).

[35], dynamic programming [107], relaxation labelling [101, 215] and hierarchical model fitting [186]. However, methods for *generalisation* of relational structures have only been addressed sporadically in the literature (such as by Michalski and Stepp [209]) within the framework of inductive learning of symbolic structural descriptions. To date, little attention has been devoted to the problem of developing techniques for the automated learning of such relational descriptions, and to the associated problem of generalising over spatial structures.

### 3.3 Relational Learning

Relational learning has been implemented within inductive logic programming systems such as GOLEM or CIGOL [221] and first order inductive logic systems such as the FOIL system of Quinlan [272]. This is achieved by utilising relational constraints in the data to construct new clauses. Systems based on relational evidence theory [250] have also been successfully used for learning relationships in spatial image domains. These include the conditional rule generation (CRG) system of Bischof and Caelli [30] and the graph matching system of Pearce, Caelli and Bischof [251] which have been used for pattern and object recognition. However, relational learning procedures and the evidential frameworks used, must be tailored for efficient schematic interpretation. The learning approach must utilise the relational constraints present in the application to optimise the representational hierarchies and search strategies used during learning.

The efficiency of schematic interpretation systems is measured not only in terms of the ability to learn to classify patterns, but also by their computational complexity and capacity to accommodate large numbers of different patterns. Another factor is the number of training examples required. This is typically reflected in the ease of interactive inputting and manipulating training patterns. For example, applications in handwritten character recognition may require large numbers of training examples for each different symbol or character [218, 258].

In an incremental learning approach, representations which discriminate between input patterns are built up from training examples incrementally through a process of hypothesis-and-test and the man-machine interaction and decision processes are thus also improved incrementally.

In the context of incremental evidence-based relational learning, the process of generalising not only requires a way to generate new rules, a kind of generalising engine, but also a way to evaluate rules to address the process of searching for sets or hierarchies of rules. These two aspects are reflected in different approaches to the problem, learning by inductive logic programming and learning using graph models and graph matching.

**Inductive logic programming** has addressed the generation of rules in spatial domains. It has been demonstrated in several applications [40,

242] and can be used for part-based recognition. In these systems, search is addressed using some form of heuristic strategy [325]. Inductive logic programming systems have utilised methods such as entropy for term selection during rule conjunction (see the FOIL system [272]), minimum description length [225, 287] for evaluating and updating representations (see the FOCL [247, 248] and MOBAL [369] systems) and probabilistic stochastic methods (see stochastic inductive logic programs [222]).

For example, a Chinese character recognition system based on FOIL [271, 272] has been developed by Amin, Sammut and Sum [9] and is capable of recognising large numbers of different characters. The Conditional Rule Generation system (CRG) (see below) has been used for learning to recognise three-dimensional objects from two-dimensional grey scale views [203].

**Graph models and graph matching** methods have been used to build relational representations for describing spatial patterns. Interpretation techniques use labelled attributed graphs to describe patterns and use graph matching techniques to establish explicit correspondence between unseen patterns and known patterns. Interpretation trees have been used by Grimson [126] for three-dimensional object recognition. Invariant feature indexing of interpretation tables have also been used [112, 113] for object recognition based on computer aided design (CAD) based models.

A polynomial technique for detecting subgraph isomorphism by pre-compiling the representation is described in [207]. The system has been applied to handwritten symbol and diagram recognition [208].

The proposed CLARET algorithm integrates relational learning (in the form of the Conditional Rule Generation method) with graph matching for the incremental learning of hierarchical relational representations. CRG is used for generating relational rules over spatial constraints. Rulegraph matching is subsequently used for checking the compatibility between sets of rules and optimising the representation and search involved.

### 3.3.1 Conditional Rule Generation

Conditional Rule Generation (CRG) is an inductive learning and rule generation procedure which takes into account the spatial relationships inherent in relational structures or patterns.

The CRG algorithm generalises over labelled continuous (numerical) data by substituting labels with variables [30]. This allows for the analysis of paths or chains through instantiation of unary ( $u$ ) and binary ( $ub$ ) features states. Rules are generated of the form,

$$r_i(\lambda_I) \leftarrow u(\lambda_I), ub(\lambda_I \lambda_J), ubu(\lambda_J), ubub(\lambda_J \lambda_K) - \dots \quad (3.1)$$

where  $u, ub, ubu, \dots$  correspond to feature states which are conditional upon adjacent, labelled states and  $\lambda_I, \lambda_J, \lambda_K, \dots$  are label variables.

The CRG technique creates trees of hierarchically organised rules for classifying structural pattern parts. The algorithm searches for the occur-

rence of unary and/or binary feature states between connected components of the training patterns. Here, the occurrence of more than one pattern sample, as measured by a non-zero entropy (information,  $H$ ) statistic, indicates that higher order rules (extension of the right-hand side of Equation 3.1) are required. A backtracking, rule-splitting procedure resolves ambiguity while achieving greatest generalisation.

Rules are generated through a depth-first decision tree expansion and cluster refinement similar to attribute-induced systems such as C4.5 [273]. Each pattern or object is composed of a number of labelled parts where, in turn, each part  $\lambda_I$  (where  $I = 1, \dots, p$ ) is described by a set of unary features  $u(\lambda_I)$  and directed pairs of parts  $(\overrightarrow{\lambda_I \lambda_J})$  belonging to the same pattern (but not necessarily all possible label pairs) are described by a set of binary feature vectors  $b(\overrightarrow{\lambda_I \lambda_J})$ .

The simple two-dimensional example shown in Figure 3.3(a) contains two different patterns defined by circles with different arrangements of parts with unary features of size and colour and binary features of length and colour difference.

An initial unary feature space is constructed for all parts over all samples and patterns,

$$U_I = \{u(\lambda_I) : I = 1, \dots, p\}$$

where feature attributes from all labelled parts  $\lambda_I$  are combined (see Figure 3.3(b)).

The feature space is then partitioned into two clusters  $u_1$  and  $u_2$  by minimising an information measure  $H$ , or cluster entropy statistic,

$$H_i = - \sum_j p_{ij} \ln p_{ij}$$

where  $p_{ij}$  defines the conditional probability of elements of cluster  $i$  belonging to pattern class  $j$ .

At this stage the initial clustering procedure is not critical since recursive splitting is used to refine clusters. Clusters that are unique with respect to pattern membership (with entropy  $H = 0$ ) provide a simple classification rule for some patterns. Each non-unique cluster  $u$  is extended with respect to binary features by constructing the (conditional) binary feature space,

$$U_I B_{IJ} = \{b(\overrightarrow{\lambda_I \lambda_J}) \mid u(\lambda_I) \in U_I\}.$$

This feature space is clustered with respect to binary features into clusters  $ub$ . Again, clusters that are unique with respect to class membership provide classification rules for some objects. Each non-unique cluster  $ub$  is then analysed with respect to unary features of the second part and the resulting feature space,

$$U_I B_{IJ} U_J = \{u(\lambda_J) \mid b(\overrightarrow{\lambda_I \lambda_J}) \in U_I B_{IJ}\}$$

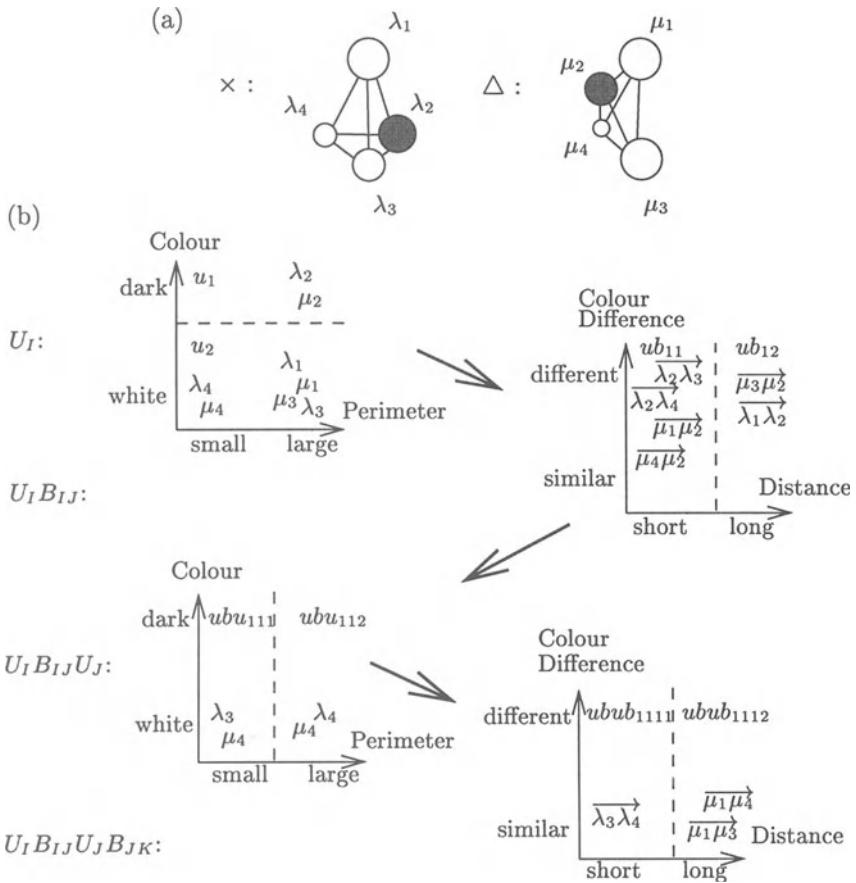


Figure 3.3: Conditional rule generation: In (a) two patterns are shown which contain circles with unary features of size and colour and binary features of length and colour difference. In (b) the conditional rule generation procedure (CRG) relies on label-compatible paths ( $u(\lambda_I) \rightarrow ub(\lambda_I \lambda_J) \rightarrow ubu(\lambda_J) \rightarrow ubub(\lambda_J \lambda_K) \rightarrow \dots$ ). The unresolved clusters are expanded until either all rules are resolved or the pre-determined maximum rule length is reached, in which case rule splitting occurs.

is clustered into clusters  $ubu$ .

In general, unique clusters provide pattern classification rules for some objects. The other clusters have to be further analysed, either by repeated conditional clustering involving additional parts at levels  $ubub$ ,  $ububu$ , etc., or through cluster refinement which is based on a partition entropy measure (see Bischof and Caelli [30] for more details).

Classification is enacted by activating rules that descend the conditional rule tree and link compatible parts together into paths or *evidence chains*

according to the *IJ-connectivity* constraint which constrains the way conjunction of unary and binary terms should occur.

There is no guarantee that the rules are optimal for classifying patterns. Conflicting evidence can arise in cases where either parts are occluded or missing in patterns, or when rules for different patterns may be activated. Further, the rules may not necessarily be of minimum length with respect to classification as zero entropy ( $H = 0$ ) is not needed.

### 3.3.2 Rulegraph Matching

Rulegraphs allow the representation of compatibilities between sets of rules (inter-rule compatibilities). The compatibilities can be represented by considering the unary rules as vertices and the binary rules as edges in a graph using the labels to determine the connections.

A *Rulegraph* is a graph of rules in which vertices correspond to rules and edges correspond to the relationships between rules. For the case where separate unary  $u_i$  and binary  $b_j$  rules are present, the following connection criterion is used.

**Criterion 1 Connection criterion:** Two unary rules  $u_i$  and  $u_j$  are connected by a binary rule  $b_k$  if there exists labels  $\lambda_I, \lambda_J$  which produce instantiations  $u_i(\lambda_I)$  and  $u_j(\lambda_J)$  and there exists  $b_k$  such that  $b_k(\lambda_I \lambda_J)$  holds.

A *Rulegraph* for a training pattern corresponds to a graph where rules replace pattern parts and their relationships. Rulegraphs, therefore, explicitly represent the rules and their interrelations via shared label instances and they capture compatibility information with respect to the structural aspects of the pattern description.

The two-dimensional circles patterns in Figure 3.4 are used to label the unary and binary rules according to the mapping of parts and part relationships into each feature space. The generation of Rulegraphs is shown where unary rules are labelled with single labels and binary rules are labelled with label pairs. Rulegraphs may then be formed according to the connection criterion.

The Rulegraph is a convenient representation since a lower cardinality than the original graph can be achieved by selecting fewer rules than labels; each unary rule may contain multiple labels. Further, multiple binary rules may connect two unary rules consistent with this connection criterion. The adjacency information is reflected by the connectivity in the Rulegraph according to the labelling of each rule vertex and edge.

The likelihood of a Rulegraph corresponding to each pattern in the training set may be determined by the evidence weights for each rule vertex and edge. For each unary and binary rule the probabilities are determined from the pattern frequencies within a given rule's bounds in feature

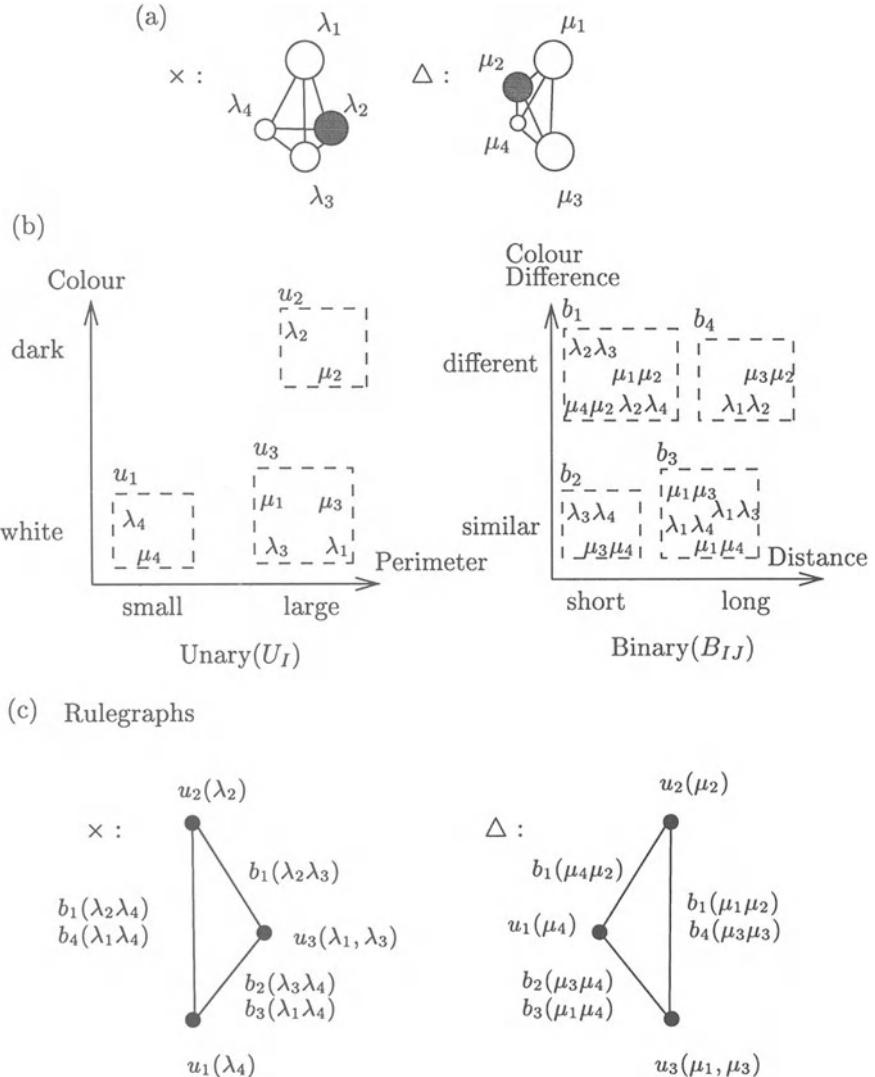


Figure 3.4: Rulegraph matching: The two-dimensional circles patterns are shown in (a) containing two different patterns with differently arranged parts. Training patterns are used in (b) to label the unary and binary rules according to the mapping of the parts and their relationships into each feature space. Unary rules have single labels and binary rules have label pairs. Rulegraphs may then be formed according to shared labels and these are shown in (c).

space. For example, for rule  $u_3$  from Figure 3.4(b), conditional probabilities  $p(\times | u_3) = 0.5$  and  $p(\Delta | u_3) = 0.5$  are derived.

At run time, parts in the sample pattern activate unary and binary rules based on their feature states. Dynamic programming principles and evidence weights are used to prune the search using an A\* search for label-compatible rules. Label compatible sets of rules are found by checking the existence of compatible labels between unknown patterns with respect to Rulegraphs and solving label mappings (for more details see Pearce, Caelli and Bischof [251]).

The main limitation of the Rulegraph system is the absence of any rule induction stage. Rules must be provided to the system a priori. Conditional rule generation, on the other hand, can induce rules but does not provide a finite set of rules for classification. There is no *evaluation* of the rule tree or method of managing multiple, possibly conflicting, rule instantiations. This is relevant to inductive logic programming systems in general, which do not necessarily order rules or manage conflict between multiple rules. Under these conditions, the search for solutions can be substantial or even intractable.

Entropy and minimum message length measures may be used to guide the search process during rule generation. These measures assume singular evidential sources (one part per class) and lists of attributes are often used. Under these conditions it is reasonable to assume independence between different feature attribute variables with respect to their contribution to pattern membership [353]. However, when there are *multiple* parts, the feature states of parts and relationships from the same class *are* dependent [180]. Minimum message length and entropy measures are based on the same independence of evidence assumption and both rely on a universal prior probability assumption about the existence of parts and their relationships [180].

For schematic interpretation, the treatment of labelled patterns with multiple parts necessarily involves the instantiation of multiple rules. This means that the probabilistic dependency between rules really needs to be considered *explicitly*.

### 3.4 The Consolidated Relational Learning Algorithm (CLARET)

In this section, the two different approaches to evidential learning of relational data structures are presented. Relational evidence theory integrates decision tree methods with inductive logic programming and graph matching, for evaluating and updating representations in spatial domains. A learning algorithm, the Consolidated Learning Algorithm based on Relational Evidence Theory (CLARET) is presented, which combines the rule

induction of the Conditional Rule Generation (CRG) system with the explicit rule evaluation of the Rulegraph system, resulting in a consolidated approach to generalisation.

CRG is used for generating relational rules over spatial constraints. Rulegraph matching is subsequently used for checking the compatibility between sets of rules and optimising the representation and search involved. A Bayesian conditioning framework is used to define label-compatibility measures into rule evaluation which allows an interpretation stage to be incorporated into the generalisation process. The system is applied to schematic diagram and symbol recognition. The approach uses an adaptive representational bias and search strategy by efficiently grounding the learning procedures in the relational constraints inherent in the spatial data.

The algorithm is diagrammatically presented in Figure 3.5 for recognising patterns comprising hand-drawn symbols. An unknown pattern is presented to the system together with  $n$  known patterns.

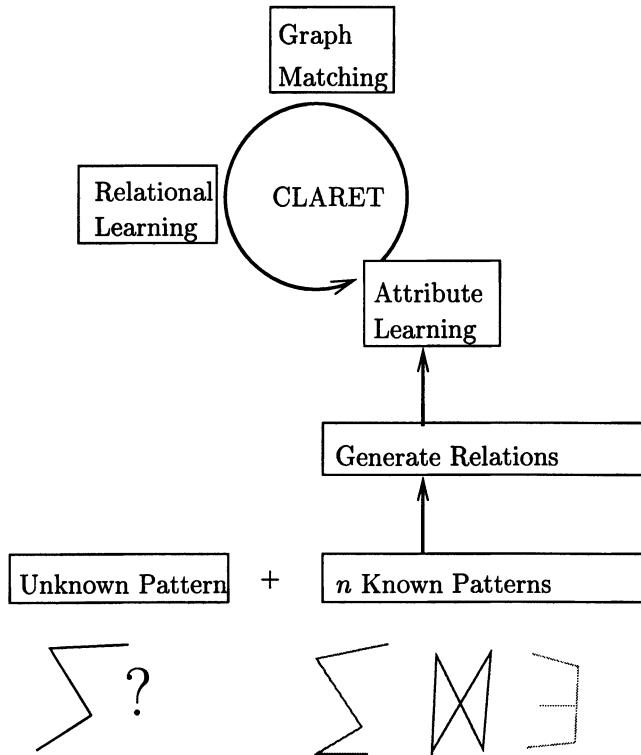
The CLARET algorithm utilises dynamic generalisation in the sense that there is no separate learning phase and no rules are stored (other than relations) for each pattern. This allows for the incremental addition of new patterns, and rules are dynamically generated each time an unknown pattern is recognised. Representations are adaptively generated as required by the data.

First, relational structures are generated by extracting relationships between pattern segments. Second, the patterns are matched using the feature attributes and labels from relational descriptions. This involves the repetition of the following steps: **attribute learning**, **graph matching** and **relational learning**:

- The attribute learner utilises the numeric feature attributes of relations to form rules (feature states) by specialisation.
- Graph matching is used to form interpretations of unknown patterns with respect to known patterns and to solve the mappings of parts (using Rulegraph matching), by finding compatible sets of labelled rules.
- Relational learning allows for the generation of conditional rules which create paths through the relational structures (based on Conditional Rule Generation).

### 3.4.1 Step 1: Generate Relations

Relational descriptions are generated by extracting relationships between the segments of input patterns. From all known patterns, attributed and directed binary relations are generated using labelled pattern parts.



**Figure 3.5:** The Consolidated Learning Algorithm based on Relational Evidence Theory (CLARET) for matching an unknown pattern to one of  $n$  known patterns, shown here matching segmented symbols. Relational descriptions are generated by extracting relationships between pattern segments. The attribute learner utilises the numeric feature attributes of relations to form rules (feature states) by specialisation. Graph matching is used to form interpretations of unknown patterns with respect to known patterns and to solve the mappings of parts by finding compatible sets of labelled rules (using Rulegraph matching). Relational learning allows for the generation of conditional rules, which form paths through the relational structures (using a form of conditional rule generation).

**Definition 3** A relation  $(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots, F_n)$  represents the directed binary relationship from labelled parts  $\lambda_I$  to  $\lambda_J$  in terms of feature attributes  $F_1, F_2, \dots, F_n$ .

Relations from all patterns are combined to form a feature space  $B_{IJ}$ .

**Definition 4** A binary feature space represents the set of all relations from all patterns  $B_{IJ} = \{(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots, F_n) : I = 1, \dots, p; J = 1, \dots, q\}$ .

Directed relations are extracted between pairs of line segments, in this case for relative rotation and scale. For the example shown in Figure 3.6,

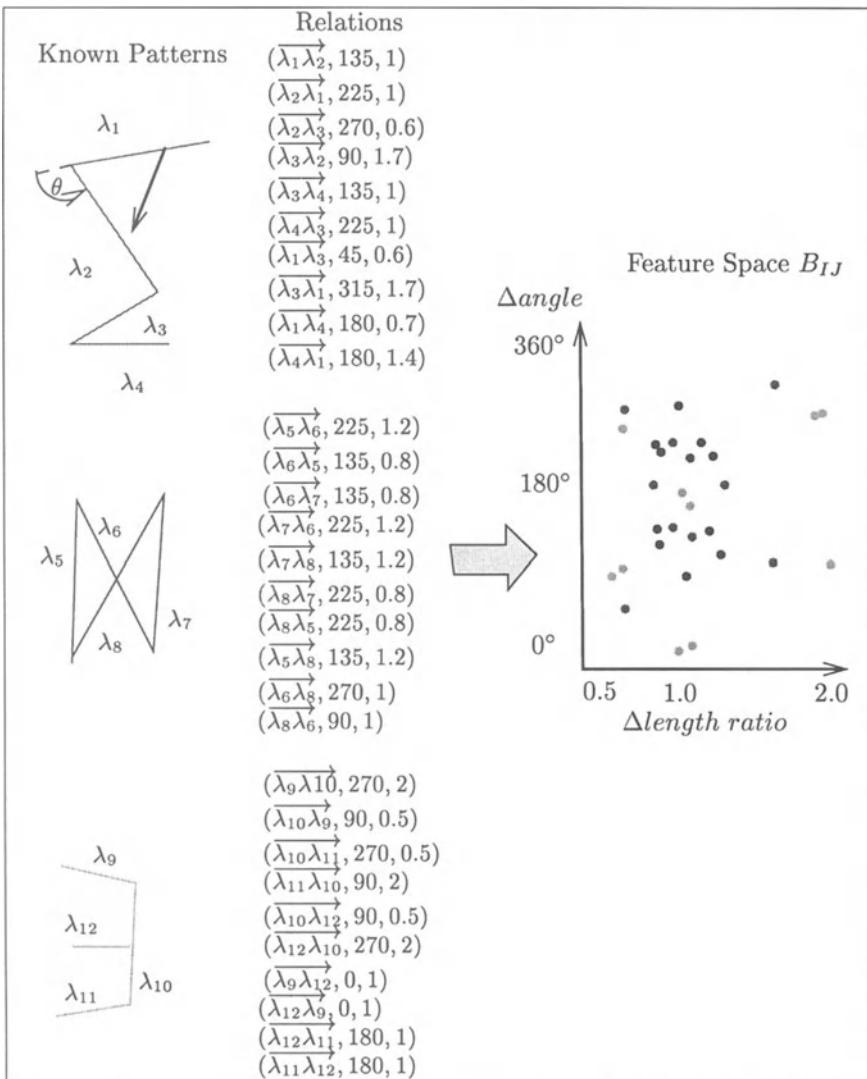


Figure 3.6: Generating binary relations: For all known patterns, relations are generated which form directed, planar relational structures and form a feature space  $B_{IJ}$ . The relations  $(\overrightarrow{\lambda_I \lambda_J}, \Delta ANG, \Delta LEN)$  represent difference in angle ( $\Delta ANG$ ) and length ratio ( $\Delta LEN$ ). In this example, the segmented lines are assumed to be directed, obtained from an on-line input device such as a digitiser or pen.

binary relations are generated of the form  $(\overrightarrow{\lambda_I \lambda_J}, \Delta ANG, \Delta LEN)$  where  $\Delta ANG$  is angle difference and  $\Delta LEN$  is length difference. In this example the lines are assumed to be directed, obtained from an on-line input device such as a digitiser or pen. Relations are generated so that only non-overlapping relational structures are formed.

**Definition 5** *In a planar relational structure, the relations, when considered as edges in a two-dimensional plan, do not overlap.*

Relations forming planar graphs are generated, reasonably efficiently, by sorting all relationships based on distance and allocating pairs one by one in order of distance, while checking that edges do not overlap at each addition.

Next, relations are generated from the unknown pattern and included in feature space  $B_{IJ}$  (see Figure 3.7).

### 3.4.2 Step 2: Attribute Learning

The method used for partitioning these (binary) attributes is based on the feature selection and splitting technique used in Conditional Rule Generation (see Section 3.3.1), except that the information metric is replaced by a variance criterion. Feature attributes are partitioned into regions and each

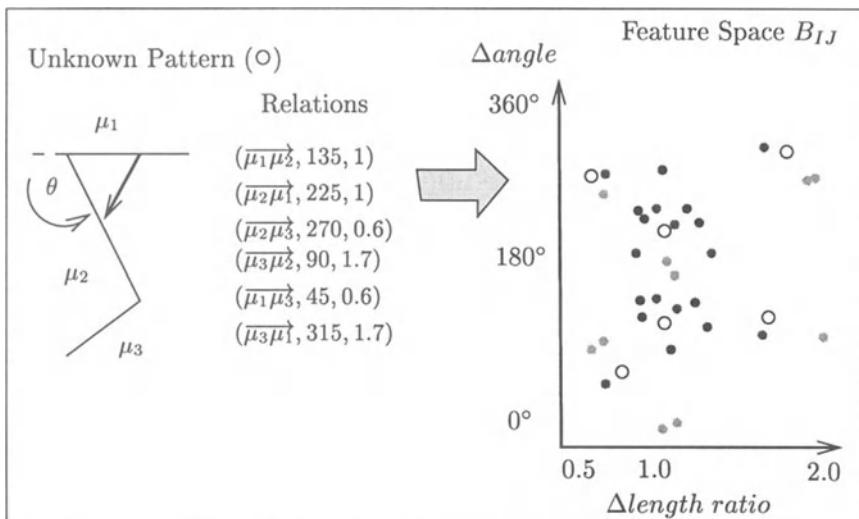


Figure 3.7: Binary relations for unknown pattern: Relations are generated from the unknown pattern ( $\circ$ ) and are included into feature space  $B_{IJ}$ . Relations are also of the form  $(\overrightarrow{\mu_I \mu_J}, \Delta ANG, \Delta LEN)$  representing the same difference in angle ( $\Delta ANG$ ) and length ratio ( $\Delta LEN$ ) used for the known patterns.

split results in two new regions defined by the conjunctions of attribute bounds: rules ( $r_i$ ), see Figure 3.8.

A least general technique is used which approximates K-means splitting [152] and minimises the attribute range of each new rule by maximising the split between rules.

**Criterion 1** *The least generalisation  $\gamma_{i,j,k}(t)$  at iteration  $t$ , defines the best split over all attributes ( $f_i, i = 1, \dots, n$ ), rules ( $r_k, k = 1, \dots, R_{t-1}$ ) existing up to and including iteration  $t - 1$ , and discrete attribute values  $j$  (for each feature). It is determined by computing, at each iteration (new splitting procedure):*

- $I_j^{i,k}$ : The intervals between adjacent discrete attribute ( $i$ ) values ( $j$ ) for all relations (both existing and current) for each rule  $k$  (region, see Figure 3.8).
- $\bar{I}^{i,k}$ : The average interval length between adjacent discrete attribute ( $i$ ) values for each rule ( $k$ ).

The actual splitting interval chosen is then defined by

$$\gamma_{i,j,k}(t) = \operatorname{argmax}_{i,j,k} \left( \frac{1}{n_{t-1}^i} \left[ \frac{I_j^{i,k}}{\bar{I}^{i,k}} \right] \right) \quad (3.2)$$

where  $n_{t-1}^i$  corresponds to the total number of previous splits on attribute  $i$  current at iteration  $t$ . This weights for ‘novelty’: The bias towards selecting new attributes is used to avoid the process falling into local minima and focusing on only one feature. This is similar to what is known to occur in clustering [152].

An initial rule,  $r_0$ , is generated by forming the least generalisation ( $\gamma_{i,j,k}$ ) with respect to all the relations in attribute space  $B_{IJ}$ . Rules are therefore of labelled first-order form,

$$r_i(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots) \leftarrow F_1 \geqslant \text{attribute\_value}_1, F_1 \leqslant \text{attribute\_value}_2, \dots, \\ F_2 \geqslant \text{attribute\_value}_3, F_2 \leqslant \text{attribute\_value}_4, \dots$$

Since the process is recursive, each such rule can be repartitioned by calculating the least generalisation and maximising Equation 1 in each attribute. The best attribute to partition is selected based on the least generalisation  $\gamma_{i,j,k}$ . The partitioned rule is replaced by two new least general rules (see Figure 3.8, rule  $r_0$  is replaced with rules  $r_1$  and  $r_2$ ). Such hashing of the observed relational attributes provides initial hypothesis about the resolution of parts required to recognise and discriminate between the known patterns.

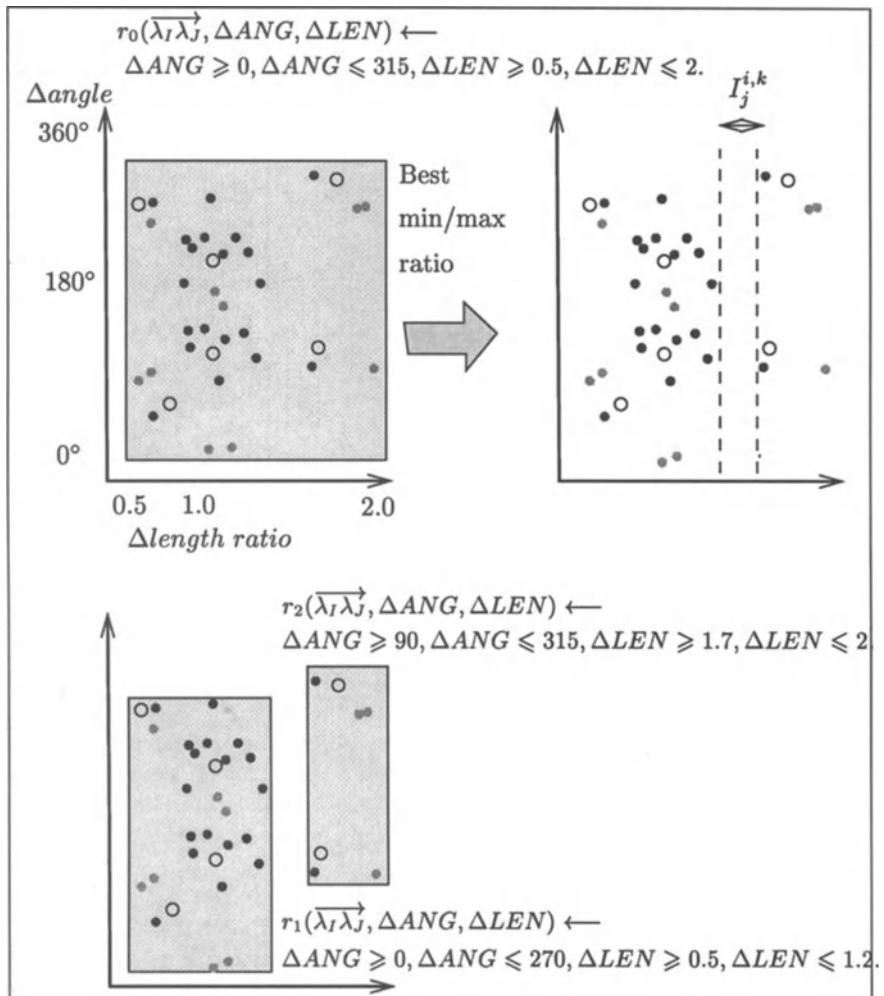


Figure 3.8: Attribute learning: An initial rule  $r_0$ , is constructed by forming the least generalisation with respect to the relations in feature space  $B_{IJ}$  (top left). A rule,  $r_k$  is selected from a feature space  $B_{IJ}$  along attribute  $i$  at interval  $I_j^{i,k}$  with the largest least generalisation  $\gamma_{i,j,k}$ , as determined in Equation 1 (top right). This corresponds to minimising the attribute range of each new rule by maximising the range between rules. After partitioning, two new least general rules  $r_1$  and  $r_2$  replace rule  $r_0$  (bottom).

### 3.4.3 Step 3: Graph Matching

Where CLARET differs from Conditional Rule Generation in so far as the essence of CLARET is to check that the set of rules are sufficient to uniquely

differentiate patterns before further attribute splitting or rule extension. Graph matching is used to solve the mapping of unknown pattern parts (labels) with respect to known pattern parts, by finding compatible sets of labelled rules using Rulegraph matching. Inter-rule dependencies are determined for subgraphs of rules.

Rulegraphs are formed when labelled relations instantiate rules according to their feature attribute values. The following connection criterion is used.

**Criterion 2** *Connection criterion:* Two rules  $r_i$  and  $r_j$  are connected if there exists instantiations  $r_i(\lambda_I \lambda_J)$  and  $r_j(\lambda_J \lambda_K)$ , such that label  $\lambda_J$  is shared.

For example, in Figure 3.9(a), relation  $\overrightarrow{\lambda_1 \lambda_2}$  instantiates rule  $r_1$  to form  $r_1(\overrightarrow{\lambda_1 \lambda_2})$ . A Rulegraph for the known pattern is shown in Figure 3.9(b) according to connection Criterion 2.

Label mappings between unknown and known patterns  $\mu_J \mapsto \lambda_J$ , are solved by checking the existence of common labels in rules. This relies on checking the compatibility between rules based on the consistency of the mapping states in the unknown pattern with respect to known pattern, according to the following existence criterion.

**Criterion 3** *Existence criterion:* Given mapping states  $\mu_I \mapsto \lambda_I$  and  $\mu_J \mapsto \lambda_J$ , if instantiation  $r_c(\overrightarrow{\lambda_J \lambda_K})$  exists for the known pattern in candidate rule  $r_c$ , then there must exist a relation  $\mu_J \mu_K$  in the unknown pattern such that  $r_c(\overrightarrow{\mu_J \mu_K})$  holds.

The Label Compatibility Checking Method for CLARET checks the compatibility between rules based on existence Criterion 3 and solves label mappings between unknown and known patterns (see Figure 3.9(c)). The steps used to determine compatibility between candidate rule  $r_c$  and interpretation rule  $r_i$  are shown in Algorithm 1.

The algorithm is similar to that of the Rulegraph algorithm (for details see Pearce, Caelli and Bischof [251]) in that it represents a generalisation of the traditional subgraph isomorphism algorithm [21, 341, 366]. However, in CLARET, checks are made over binary rules only (pairs of labels) and unary rules (single labels) are not used. Steps 1–4 of the label compatibility checking method describe a technique for checking compatibility between two rules. The matching process may now proceed by finding subgraphs of rules which are all pair-wise compatible, or *interpretations* [21]. Interpretations are also known as cliques.

**Definition 6** An interpretation of compatible rules  $r_i$  corresponds to a subgraph of rules where each rule is label compatible with every other rule.

Interpretations are built up by testing compatibility of candidate rules  $r_c$  for possible addition to interpretation rules  $r_i$ . The order of assignment is significant in terms of the different label mapping states which arise.

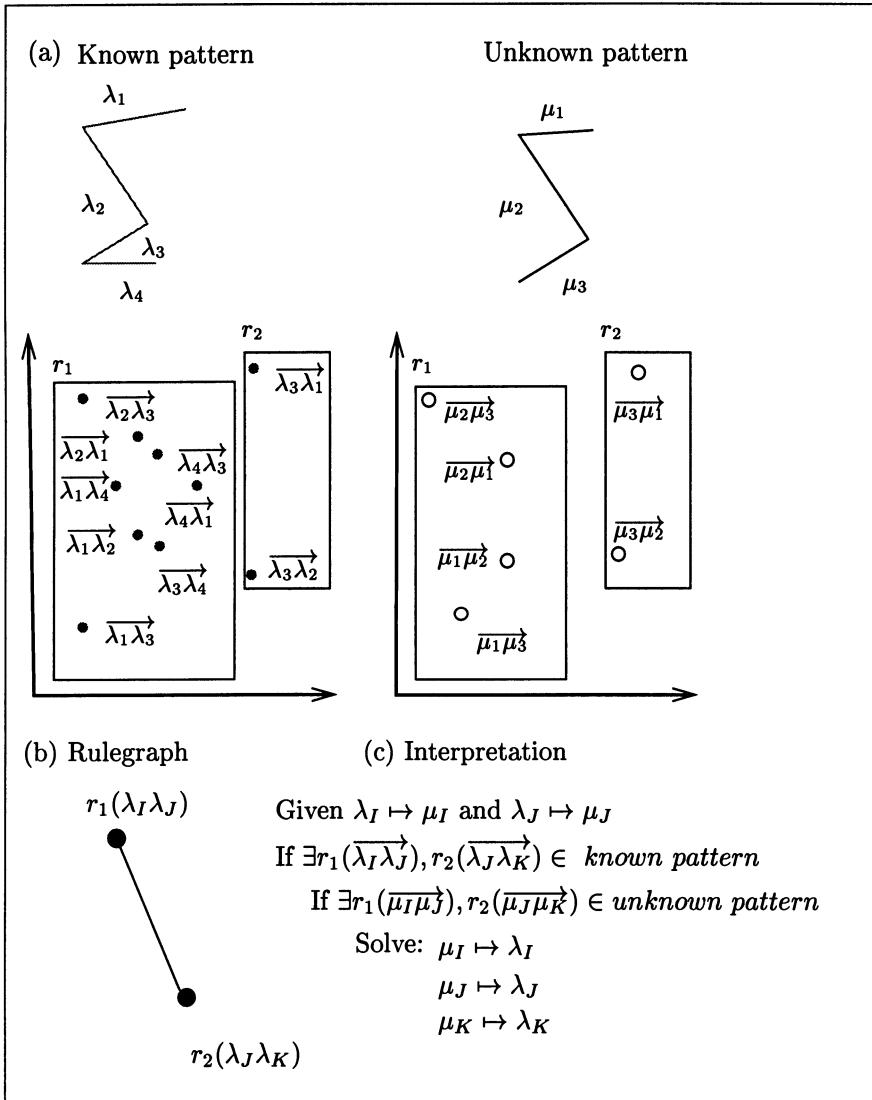


Figure 3.9: Graph matching: Instantiation of the partitioned rules with relation labels is shown in (a). A Rulegraph is shown in (b) based on the connection criterion (Criterion 2). Interpretations are formed in (c) by checking the existence of common labels between known and unknown patterns in rules and solving label mappings (Algorithm 1).

**Algorithm 1** Label Compatibility Checking Method for relations: This algorithm checks the compatibility of a candidate rule  $r_c$  for adding to the set of compatible rules in the current interpretation and is shown here solving label mappings for just one such candidate-compatible rule pair. In the algorithm,  $\lambda_I, \lambda_J, \lambda_K$  are variables for labels in the known pattern and  $\mu_I, \mu_J, \mu_K$  are variables for labels in the unknown pattern.

**Label Compatibility Checking Method for relations:**

1. From interpretation rule  $r_i$  we have instantiations  $r_i(\overrightarrow{\lambda_I \lambda_J})$  and  $r_i(\overrightarrow{\mu_I \mu_J})$ . Possible mapping states are  $\mu_I \mapsto \lambda_I$  and  $\mu_J \mapsto \lambda_J$ .
2. Existence criterion: Given mapping states  $\mu_I \mapsto \lambda_I$  and  $\mu_J \mapsto \lambda_J$ , if instantiation  $r_c(\overrightarrow{\lambda_J \lambda_K})$  exists for the known pattern in candidate rule  $r_c$ , then there must exist relation  $\mu_J \mu_K$  in the unknown pattern such that  $r_c(\overrightarrow{\mu_J \mu_K})$  holds.
3. For valid existence checks, solve the mapping states by either instantiation (if the label is *not* yet mapped) or elimination (if the label *is* mapped). The new mapping states are:  $\mu_I \mapsto \lambda_I$ ,  $\mu_J \mapsto \lambda_J$  and  $\mu_K \mapsto \lambda_K$ .
4. There must exist at least one shared label mapping  $\mu_K \mapsto \lambda_K$  for the candidate rule  $r_c$  to be compatible with interpretation rule  $r_i$ .

Interpretations are made by mapping labels which instantiate rules for the unknown pattern with respect to each known pattern. Initially, labels are mapped from unknown patterns to known patterns for labels only in the root rule  $r_0$ . The mapping will initially be many to many, in the sense that rule  $r_0$  is completely general and has not yet been specialised, partitioned by the attribute learner.

An interpretation of the unknown pattern is generated for each known pattern and added to a queue, which is maintained in order of relational evidence which is presented in Section 3.5. (see Figure 3.10(a)). At each invocation of the graph matcher, an interpretation is removed from the head of the queue and the attribute learner is called which also selects the best rule to split. New interpretations are formed by generating subgraphs of newly partitioned rules and put back onto the queue. In the process, the mapping between labels of unknown and known patterns is solved. Multiple mapping states, denoted by

$$\mu_I \mapsto \lambda_I \vee \lambda_J \vee \lambda_K \dots$$

are updated either by instantiation (if the label is *not* yet mapped) or by elimination (if the label *is* mapped) using the mappings generated from the existence checks (see Figure 3.10(b)).

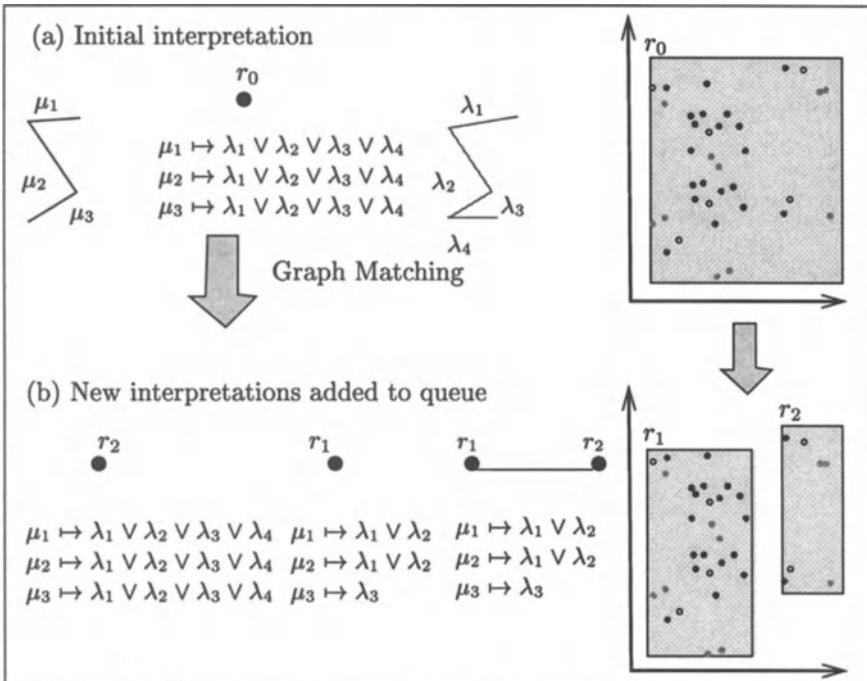


Figure 3.10: Interpretations: In (a) an initial interpretation is formed for a known pattern. The label mappings are initially many to many. New interpretations are formed from all subgraphs of new rules and added back on to the interpretation queue in (b). In the process, the mapping between labels of unknown and known patterns is solved. Mapping states are updated by either instantiation (if the label is *not* yet mapped) or elimination (if the label *is* mapped).

### 3.4.4 Step 4: Relational Learning

The purpose of relational learning is to generate *conditional* rules by creating paths through the relational structures. The method is based on Conditional Rule Generation and allows for (intra-rule) dependencies between feature states to be represented by creating paths through the relational descriptions.

Conditional feature spaces  $B_{JK}$  are formed by traversing paths via relations in feature space  $B_{IJ}$  to relations in feature space  $B_{JK}$  and extending rules relationally.

**Definition 7** *Relational learning forms conditional feature spaces*

$$B_{JK} = \{r_j(\overrightarrow{\lambda_J \lambda_K}) \mid r_i(\overrightarrow{\lambda_I \lambda_J}) \in B_{IJ}\}$$

for each of the rules  $r_i \in B_{IJ}$ .  $IJ$ -connected paths are traversed from in-

stantiations  $r_i(\overrightarrow{\lambda_I \lambda_J})$  to relations  $\overrightarrow{\lambda_J \lambda_K}$  (via label  $\lambda_J$ ).

A new conditional rule  $r_j$  is formed by calculating the least generalisation  $\gamma$  in feature space  $B_{JK}$  and including term  $r_i$  on the right-hand side of rule  $r_j$ . For example, the rule,

$$\begin{aligned} r_j(\overrightarrow{\lambda_J \lambda_K}, F_1, F_2, \dots) \leftarrow \\ r_i(\overrightarrow{\lambda_I \lambda_J}, \dots), F_2 \geqslant \text{attribute\_value}_3, F_2 \leqslant \text{attribute\_value}_4, \dots \end{aligned}$$

is a relational extension of rule,

$$r_i(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots) \leftarrow F_1 \geqslant \text{attribute\_value}_1, F_1 \leqslant \text{attribute\_value}_2, \dots$$

This results in a hierarchy of rules where extended (more relational) rules exist in conditional feature spaces (see Figure 3.11).

After relational learning, interpretations which include the newly extended (compatible) rules, are added back to the interpretation queue. The best rule to partition is calculated for each interpretation, based on the least generalisation magnitude ( $|\gamma|$ ) for each feature space  $B_{IJ}$ ,  $B_{JK}$ , etc. The algorithm then returns back to the attribute learning step.

A best-first relational learning strategy for rule generation allows for subgraphs of rules to be established relative to the known patterns and the unknown pattern, as required. This is in contrast to the Conditional Rule Generation (CRG) system which uses a depth first strategy and for partitioning.

Attribute learning, graph matching and relational learning steps are repeated until there exists an interpretation with a unique label mapping for one known pattern, or no more compatible rules are formed, as determined by the label compatibility checking method. The interpretation queue contains interpretations of the unknown pattern in terms of all known patterns. Successive applications of attribute learning, graph matching and relational learning results in the specialisation of rules, while solving label mappings in interpretations (see Figure 3.12). The evaluation (ordering) of rule subgraphs is required to determine a *finite* (best) interpretation and this requires an evidential measure.

To determine the best interpretation an evidential measure is required. A relational evidence network is used to impose an ordering of interpretations during the search, which maximally prunes the search space and thus determines the best interpretation in optimal time.

## 3.5 Relational Evidence Network

In order to represent uncertainty in the CLARET hierarchy of rules, a model must be used which captures the dependency between specific instantiations

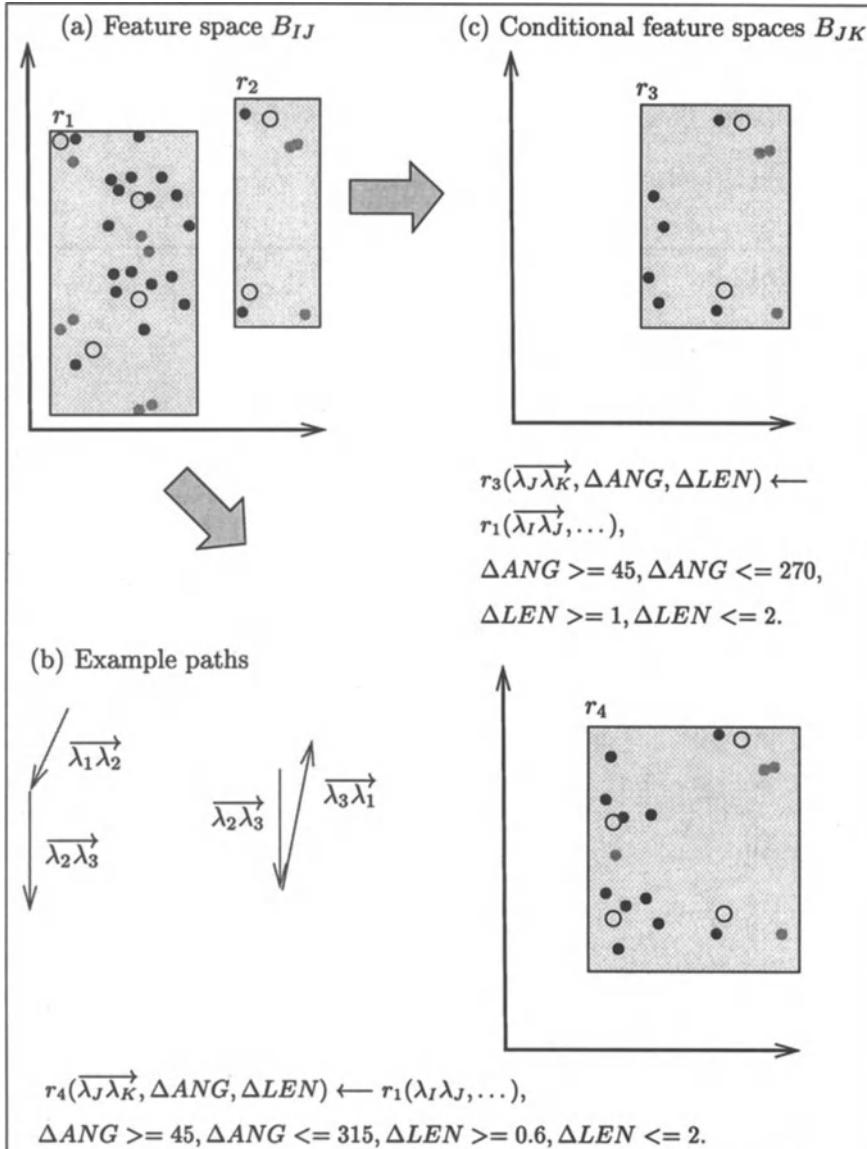


Figure 3.11: Relational learning: A feature space  $B_{IJ}$  is shown in (a). Paths are traversed via relations  $\overrightarrow{\lambda_I \lambda_J} \rightarrow \overrightarrow{\lambda_J \lambda_K}$  shown in (b). The conditional feature spaces  $B_{JK}$  formed from each of the rules  $r_i$  and  $r_2$  in feature space  $B_{IJ}$  are shown in (c).

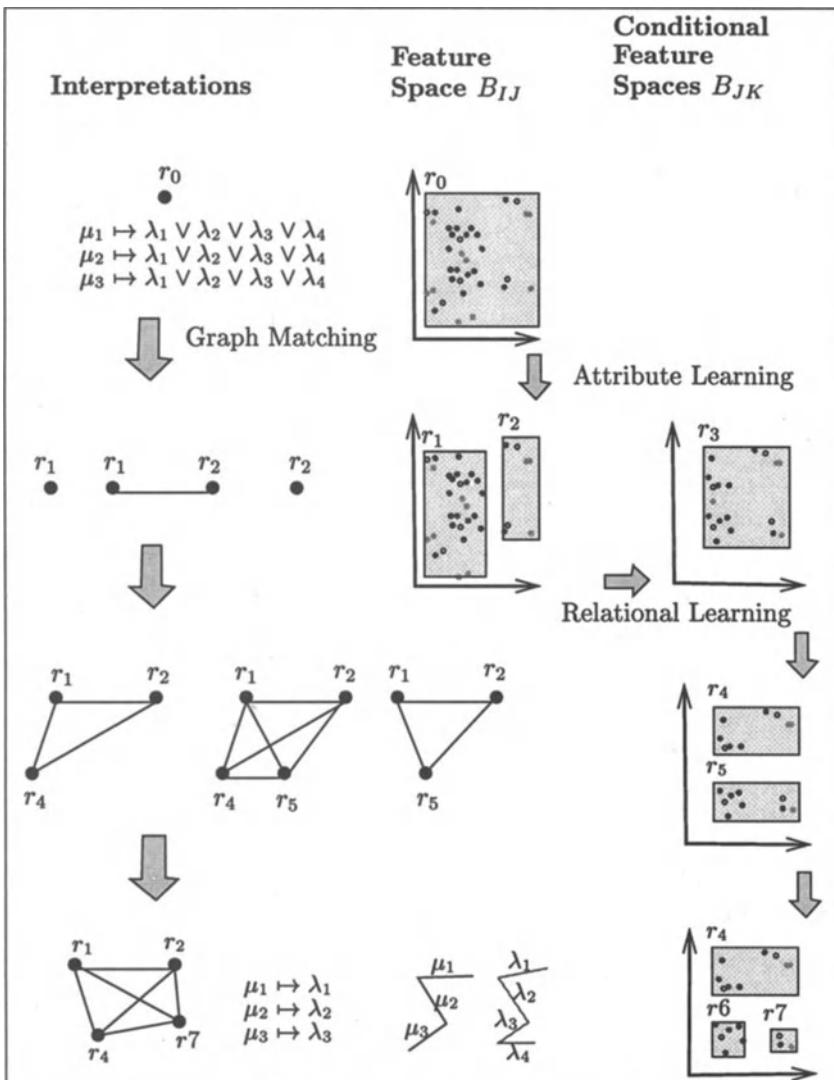


Figure 3.12: Search for the finite Interpretation: Successive applications of attribute learning, graph matching and relational learning results in best-first search. During the search, mappings between labels in the unknown pattern are solved with respect to the known pattern. The interpretation queue contains interpretations of the unknown pattern in term of all known patterns. Relational evidence measures are used to prune the search space while guaranteeing a finite interpretation.

(labeled parts) and rules (feature states). Bayesian techniques have been used to model structural descriptions in the form of matching metrics [312] or random parametric structural descriptions [308]. However, these models typically assume independence between parts and do not consider the trade off between attribute generalization and part specialization. More complex probabilistic independence networks (PINs) [317] have been used to model networks over dual probability distributions.

In CLARET there is clear dependency of parts via the instantiation of conditional rules with labeled relations, and so a Bayesian network model can be formulated to include both attribute and part indexing. Each pattern  $\omega_p$  is from an existing (though updatable) closed world of patterns  $\omega_p \in \Omega$ . Patterns give rise to relations  $(\overrightarrow{\lambda_I \lambda_J} : F_1, \dots, F_n) \in \Lambda$ , where  $\Lambda$  corresponds to all relations, with labelled parts  $\lambda_I$  and  $\lambda_J$  and feature attribute values,  $F_1, \dots, F_n$ . Attribute values are partitioned into rules  $r_i$  which represent all possible attribute states  $r_i \in \Pi$ . This allows for hierarchical modeling of such processes, shown in Figure 3.13 (see Chapter 4, pp. 143–237 of Pearl [252]).

$$\begin{array}{ccc} \Omega & \longrightarrow & \Lambda \\ \text{Patterns} & & \text{Relations} \end{array} \longrightarrow \Pi$$

Figure 3.13: Hierarchical modelling over patterns, relations and rules. Patterns give rise to labelled relations, which in turn give rise to observable features which are partitioned into different rules (feature states).

### 3.5.1 Conditioning over Relational Rules

The conditional probability of pattern  $\omega_p$  ( $\omega_p \in \Omega$ ), given relations  $\overrightarrow{\lambda_K \lambda_L} \in \Lambda$ , is

$$p(\omega_p | \overrightarrow{\lambda_K \lambda_L}) = \alpha p(\overrightarrow{\lambda_K \lambda_L} | \omega_p) p(\omega_p) \quad (3.3)$$

where  $\alpha$  is a normalizing term,  $p(\omega_p)$  being the prior pattern probability. The relational evidence for subgraphs of compatible rules is determined analytically by conditioning over both labeled relations and relations which instantiate rules. The following two equations are derived.

Conditioning over mutually exclusive rules  $r_j$  for Equation 3.3 results in

$$p(\omega_p | \overrightarrow{\lambda_K \lambda_L}) = \alpha p(\omega_p) \sum_j p(\overrightarrow{\lambda_K \lambda_L} | r_j(\overrightarrow{\lambda_K \lambda_L}), \omega_p) p(r_j(\overrightarrow{\lambda_K \lambda_L}) | \omega_p). \quad (3.4)$$

Conditioning over labeled parts from relations  $\overrightarrow{\lambda_I \lambda_J}$  for Equation 3.3 results in

$$p(\omega_p | r_j(\overrightarrow{\lambda_J \lambda_K})) = \alpha p(\omega_p) \sum_{I,J} p(r_j(\overrightarrow{\lambda_J \lambda_K}) | \overrightarrow{\lambda_I \lambda_J}, \omega_p) p(\overrightarrow{\lambda_I \lambda_J} | \omega_p). \quad (3.5)$$

The conditional probability  $p(\overrightarrow{\lambda_I \lambda_J} | \omega_p)$  can be based on the (posterior) likelihood ratio  $p(\omega_p | \overrightarrow{\lambda_I \lambda_J})$  and  $p(r_j(\overrightarrow{\lambda_I \lambda_J}) | \omega_p)$  can be based on  $p(\omega_p | r_j(\overrightarrow{\lambda_I \lambda_J}))$ . Successive applications of Equations 3.4 and 3.5 allows for conditioning over the relational rule hierarchy, by instantiating rules  $r_i(\overrightarrow{\lambda_I \lambda_J})$  and following paths to conditional rule instantiations  $r_j(\overrightarrow{\lambda_J \lambda_K})$  etc. The relational evidence for pattern  $\omega_p$  given relations  $\Lambda$ , is defined by substitution of Equation 3.5 into Equation 3.4 and so by recursive Bayesian updating [252]. The final relational evidence form is,

$$p(\omega_p | \Lambda) = \sum_{i,j} \sum_{I,J,K} \prod_{I,J,K} p(r_j(\overrightarrow{\lambda_J \lambda_K}) | \overrightarrow{\lambda_I \lambda_J}, \omega_p) p(\overrightarrow{\lambda_I \lambda_J} | r_i(\overrightarrow{\lambda_I \lambda_J}), \omega_p) \quad (3.6)$$

for relations  $\overrightarrow{\lambda_I \lambda_J}, \overrightarrow{\lambda_J \lambda_K} \in \Lambda$  and where rules  $r_i$  and  $r_j$  are from conditional feature spaces  $r_i \in B_{IJ}$  and  $r_j \in B_{JK}$ . Note that the dependency between relations and instantiated rules is captured. That is, independence is *not* assumed between parts and relational evidence is based on probability distributions of attributes and their dependencies. The CLARET algorithm is designed to maximize  $p(\omega_p | \Lambda)$  in Equation 3.6 while minimizing the right hand side with respect to least generalization of attributes. The formulation relates to the class of exact solutions to Bayesian Networks [175] and this allows for the following definitions to be made:

### 3.5.2 Attribute Probabilities

The attribute probabilities correspond to the conditional probability of relations from different patterns instantiating rules, corresponding to the term  $p(\overrightarrow{\lambda_J \lambda_K} | r_j(\overrightarrow{\lambda_J \lambda_K}), \omega_p)$ . These are determined from the likelihood of relations instantiating rules with respect to different patterns using

$$p\left(\overrightarrow{\lambda_J \lambda_K} | r_j(\overrightarrow{\lambda_J \lambda_K}), \omega_p\right) = \frac{\sum_{\overrightarrow{\lambda_J \lambda_K} \in \omega_p} r_j(\overrightarrow{\lambda_J \lambda_K})}{\sum_{\overrightarrow{\lambda_J \lambda_K} \in \Omega} r_j(\overrightarrow{\lambda_J \lambda_K})} \quad (3.7)$$

where  $\omega_p$  is one pattern out of all patterns  $\omega_p \in \Omega$ .

### 3.5.3 Relational Probabilities

The relational probabilities correspond to the conditional probability of relations from the current pattern instantiating rules with respect to the

existing pattern, corresponding to the term  $p(r_j(\overrightarrow{\lambda_J \lambda_K}) | \overrightarrow{\lambda_I \lambda_J}, \omega_p)$  in Equation 3.4. These are determined from the likelihood of relations instantiating rules from the current pattern with respect to relations in the existing pattern. From instantiations  $r_k(\overrightarrow{\mu_K \mu_L})$  in the current pattern and instantiations  $r_k(\overrightarrow{\lambda_K \lambda_L})$  in the existing pattern the relational probabilities are calculated using,

$$p\left(r_j(\overrightarrow{\lambda_J \lambda_K}) | \overrightarrow{\lambda_I \lambda_J}, \omega_p\right) = \frac{\sum_{\mu_J \rightarrow \lambda_J, \mu_K \rightarrow \lambda_K} r_j(\overrightarrow{\mu_J \mu_K})}{\sum_{\overrightarrow{\lambda_J \lambda_K} \in \omega_p} r_j(\overrightarrow{\lambda_J \lambda_K})} \quad (3.8)$$

where  $\sum_{J,K} r_j(\overrightarrow{\lambda_J \lambda_K})$  denotes the number of instantiations of rule  $r_j$  by the set of relations  $\{\overrightarrow{\lambda_J \lambda_L} : J = 1, \dots, n; K = 1, \dots, m\}$ . Only mapped parts in the current pattern are counted, as in  $\mu_J \rightarrow \lambda_J$  and  $\mu_K \rightarrow \lambda_K$ .

The underlying Bayesian network model is a tree of part indexed rules as defined by Equation 3.6. By conditioning over both relations and rules in the tree, the relational evidence network formulation allows for the analytical determination of relational probabilities and attribute probabilities.

## 3.6 Finite Interpretation

The determination of relational evidence allows for interpretations to be *ordered*. The search for different sets of compatible rules gives rise to different label mappings and different relational evidence. A *finite interpretation* can be determined by searching for the interpretation with the highest relational evidence by generating rules dynamically upon demand.

**Definition 8** *A finite interpretation corresponds to the finite subgraph of compatible rules with the highest relational evidence, as determined by Equation 3.6, which solves part mappings between the current pattern and an existing pattern.*

The relational evidence measure is used to deterministically order the search for the finite interpretation. CLARET relies on ordering of subgraphs of rules (generalisations), which allows for an explicit and admissible search strategy. The complexity of the search is reduced to the number of subgraphs of rules explored (as opposed to subgraphs of parts, see Section 3.3.2)

$$O(n^2 2^{r/3})$$

where  $n$  is the number of relations and  $r$  is the number of rules required for a unique label mapping.

### 3.6.1 Dynamic Programming

The search is further reduced using *dynamic programming* in combination with the relational evidence in Equation 3.6 to allow probabilistic pruning of the search tree.

**Principle 1** *The dynamic programming principle uses a best-first strategy. At each stage, the relational evidence for the current interpretation is evaluated and compared to a heuristic estimate of the upper bound on how good the relational evidence could become.*

The aim of dynamic programming is to find the finite interpretation of unknown pattern rules in terms of known pattern rules, i.e. to find the subgraph of compatible rules with the largest evidence weight. It is used to maximally prune the search by eliminating solutions which can not possibly end up with higher evidence. An A\* search is used, which is essentially a branch-and-bound search utilising current evidence with a heuristic estimate of the remaining evidence [230, 285, 364]. The heuristic is based on the evidence of the current match plus an upper-bound estimate of the potential evidence remaining and this results in optimal pruning of the search. The current evidence is obtained by evaluating the relational evidence weight for the rules in the current interpretation of compatible rules, plus the heuristic estimate.

The heuristic estimate is based on how much the current match could be improved through addition of extended rules. The heuristic (best possible finite interpretation) is calculated by assuming that the best possible label mapping will result from all subsequent rule extensions.

**Heuristic 1** *The heuristic estimate makes the (optimistic) assumption that known labels in rules in the current interpretation will ultimately map one-to-one with unknown labels for all subsequent subgraphs of rules.*

Since the heuristic estimate is an upper bound on how good the relational evidence could become, the search is admissible in the sense that the finite interpretation is guaranteed to have a subgraph of rules with the highest relational evidence.

A summary of the complete CLARET algorithm using A\* search for determining the finite interpretation is presented in Algorithm 2. It relies on an upper-bound heuristic estimate to order the search and branch-and-bound to prune interpretations which could not possibly lead to a better result. The search is admissible in the sense that the finite interpretation found is guaranteed to have the subgraph of rules with the highest relational evidence.

The analysis of relational evidence during the A\* search for the finite interpretation can be made as follows. Initially, the relational probabilities are 1.0 and the average number of mappings is n (every label is mapped

---

**Algorithm 2** CLARET A\* Search for the finite interpretation based on the dynamic programming principle.

---

**CLARET A\* Search for finite interpretation:**

1. Present unknown pattern together with  $n$  known patterns:
    - (a) Generate binary relations from known patterns (which form planar graphs) and form a combined feature space.
    - (b) Generate binary relations from the unknown patterns and add to the feature space  $B_{IJ}$ .
    - (c) Create an initial rule  $r_0$  by forming the least generalisation ( $\gamma$ ) for all relations for known patterns.
    - (d) Create interpretations of unknown patterns in terms of all known patterns by matching parts in  $r_0$ .
    - (e) Form a queue of the interpretations and maintain the queue in decreasing sorted order with respect to relational evidence.
    - (f) Initially, all matchings are many:many. Interpretations may be extended by adding compatible (newly partitioned or extended) candidate rules.
  2. Repeat until finite interpretation is found:
    - (a) Attribute learning:
      - i. Select an interpretation from the head of the queue.
      - ii. Partition rule  $r_i$  in the feature space  $B_{IJ}$  along the feature with the highest least generalisation magnitude ( $|\gamma|$ ).
      - iii. Generate two new least general rules.
    - (b) Graph matching:
      - i. Select the best interpretation from the queue and instantiate rules  $r_i$  with relation labels.
      - ii. Solve label mappings to form subgraphs of compatible rules (Rule-graphs) by checking the existence of common labels between known and unknown patterns in rules.
      - iii. New interpretations are formed for label-compatible subgraphs of new rules and added back to the interpretation queue using the Label Compatibility Checking Method.
    - (c) Relational learning:
      - i. Form conditional feature spaces  $B_{JK}$  by traversing paths via relations in feature space  $B_{IJ}$  and extend rules relationally (conjunction terms).
      - ii. Select the best rule to partition from any feature space  $B_{IJ}$ ,  $B_{JK} \dots$  by calculating least general generalisation magnitude  $|\gamma|$ .
  3. A finite interpretation has been found when either the interpretation can no longer be extended (no more compatible rules), or the label mapping is solved (all labels mapped one-to-one).
-

to every other label in the root rule, where  $n$  is the maximum number of labels in the scene or the memory pattern). This will result in an initial relational evidence of less than 1 provided that the evidence weights are normalised ( $\sum_i w(r_i) = 1$ ).

As matching proceeds and rules are added to the interpretation, the relational probabilities (as determined from the ratio of labels in rules from known patterns with respect to unknown patterns) either stay constant for compatible patterns or decline for incompatible patterns. The observable feature probabilities decline on average, as rules become more specific through partitioning and through relational learning. However, for patterns which instantiate rules with high evidence weights, the relational evidence improves relative to other (poorer) interpretations. If a particular rule has a relational probability of unity, then there can be no further improvement by subsequent partitioning or extending, as all labels are mapped one-to-one.

### 3.6.2 Comparison with other Algorithms

In summary, two different relational learning techniques have been adapted for generalising within spatial domains. Generalisation is addressed from two different perspectives, first, least generalisation from attribute learning and relational extension of rules, and second, finite interpretation from dynamically determining subgraphs of rules describing unknown patterns in terms of known pattern examples.

In Conditional Rule Generation (CRG), the search strategy is depth first. The Rulegraph System, on the other hand, is able to check for the compatibility of pattern data with respect to rules via efficient, best-first dynamic programming principles. The consolidated learning algorithm (CLARET) offers an analytical way to improve the classification performance and the computational effort involved in learning relational structures.

A number of systems have been developed which utilise stochastic and minimum message length methods for building and evaluating relational representations. Examples include FOIL [271], FOCL [247, 248] and MOBAL [369]. Probabilistic stochastic methods have also been developed [222]. The CHILL system [222] combines the relative least general generalisation methods of GOLEM with the information entropy methods of FOIL.

In contrast to the assumption of independence made by entropy and minimum message length methods, CLARET incorporates an interpretation stage into the generalisation process. Interpretation utilises the compatibilities between rules and takes into account the dependencies between different rule instantiations (label compatibilities). In addition, CLARET uses a relational evidence measure to deterministically order the search resulting in an admissible strategy. This is in contrast to the stochastic search strategies used by other methods.

Relational Evidence Theory extends the Bayesian conditioning framework to incorporate checks for label compatibility between rules. Search for compatible subgraphs of rules or interpretations is used rather than assuming independence between rules. The search is made tractable by reducing the cardinality of the search space with multi-labelled Rulegraphs and by applying dynamic programming principles using relational evidence measures to prune the search.

CLARET allows for intentionally defined input data in the sense that the labels from input relations are treated as identifier variables allowing for the generation of hierarchical rules but not as significant attribute variables. The labels in predicates are not treated as extensionally defined attribute values as is the case in FOIL. The similarity between the rule generation of CLARET and the predicate invention of the CIGOL system of Muggleton and Buntine [220] may also be worth investigating.

In conclusion, relational evidence theory methods have been used for checking and evaluating both the inter- and intra-rule compatibility by combining inductive logic programming with graph matching.

### 3.7 Schematic Interpretation

In this section, the Consolidated Learning Algorithm based on Relational Evidence Theory (CLARET) is tested in the domain of schematic drawings and compared with FOIL [271, 272], a related inductive logic programming system. An on-line schematic and symbol recognition application is demonstrated for learning to recognise symbols and patterns invariant to rotation, scale and shift. Performance comparisons measure the classification rate, computational efficiency and the human factors involved in incrementally training the system.

The approach used for the interpretation of spatial patterns and designs is based on the dynamic generalisation and incremental learning capabilities of CLARET and is shown in Figure 3.14. Elements of the schematic input are hierarchically processed by the same procedures according to their progression around a central search loop. The elements include individual symbols (scenery) and designs or plans (scenarios).

At the data level, a sensing device collects data and inputs it to the system according to the application domain. Sets of known patterns are collected for use in matching unknown patterns. Data is then segmented into region or edge based descriptions For a comprehensive survey of segmentation techniques see Pal and Pal [241].

At the matching level, relational features are extracted from pairs of labelled segments in continuous numerical form. The CLARET algorithm proceeds while successive application of attribute learning, graph matching and relational learning stages using dynamic programming methods. A

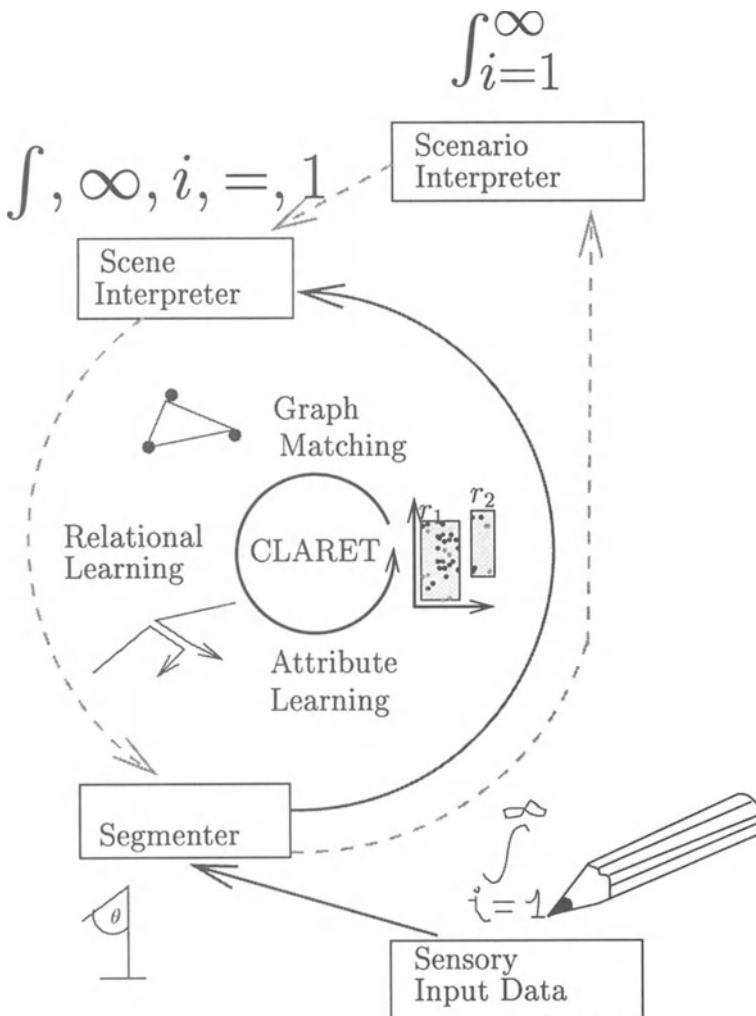


Figure 3.14: Schematic Interpretation: Sensory data is input in the form of  $x, y$ , (position) and  $z$  (pressure) sequences from the digitiser pen. Line segments are generated using polygonal approximation methods and the CLARET algorithm matches on the basis of rotation, scale and shift relationships. Successive applications of the matching algorithm result in both scene interpretations (patterns) and hierarchical scenario interpretations (sets of patterns). For the integration equation shown, there are only finite sets of symbols which one would expect as subscripts ( $i, j, =, [0-1], \dots$ ) and superscripts ( $n, m, \infty, \dots$ ) to the integration symbol  $\int$ .

finite interpretation is returned by solving the mappings between unknown and known pattern parts (see Section 3.4).

At the interpretation level, the recognised patterns are processed at two sub-levels—the scene interpretation and the scenario interpretation. Scene interpretation corresponds to the interpretation of patterns at one level, while scenario interpretation corresponds to the interpretation of hierarchical patterns and designs. Both forms of interpretation are carried out using the CLARET algorithm. In the first pass, different candidate scene symbols are identified and in the second pass, different scenarios are identified according to the relationship between symbols.

At the scene level, sets of recognised patterns are maintained and their orientations are calculated based on the label mappings relative to the spatial domain. The relationship between patterns is then established relative to their orientations and hierarchical scenarios may be processed.

In the current system, the analysis of scenario information is confined to recognising patterns from montages (sets) of patterns in scenes using a connected region strategy based on an efficient union find algorithm [332–334]. This algorithm uses a maximum independent set algorithm to find connected sets of relations and utilises efficient path compression and union by rank heuristics (for a simple description see Sedgewick [305]).

In this application, the interpretation of unconstrained hand-drawn schematic data invariant to rotation, scale and shift is investigated. Figure 3.14 shows an example of on-line hand-drawn mathematical symbol recognition.

The interpretation of different scenarios (equations) constrains the process of search for different scene patterns (symbols) by defining sets of possible symbol types at each location. For example, for the integration equation shown, there are only finite sets of symbols which one would expect as subscripts ( $i, j, =, [0-1], \dots$ ) and superscripts, ( $n, m, \infty, \dots$ ) to the integration symbol  $\int$ .

In handwritten text and symbol recognition, the classification of different types of strokes or symbols and the ability to assign higher-level descriptions to collections of such labelled regions is critical to efficient interpretation. This is particularly so when orientation and positioning information defines different scenarios, as is the case in mathematical equations.

There is a need for objective evaluation of the efficiency of development tools, especially for more difficult image interpretation problems. The use of pattern and object recognition techniques for schematic interpretation not only needs to be evaluated in terms of correctness of output, but also in the context of the interaction which occurs between the user and the system. The efficiency of these systems is critical to their usability and a number of limitations often emerge.

Systems have used attributed relational structures for recognition to

improve the robustness of classification performance and tolerance to noise and deformation (see Rocha and Pavlidis [289]). The need to introduce attributed relational attributes (such as scale) was identified by systems such as MIRABELLE [197]. The facility of integrating attribute graphs into the process of defining pattern descriptions has been demonstrated for grey scale schematic diagram interpretation [47] for rotation, scale and shift invariant symbol recognition [178] and for handwritten Chinese character recognition [188]. However, for relational structures, the input of symbols typically requires hand-coded structural descriptions (the relational structure is not learned). The classification ability is limited in the number of patterns the systems can recognise. The complexity of matching is often high, which degrades the on-line possibilities or the ability to include new symbols and patterns incrementally. For example, many systems have exponential training time and systems based on neural network approaches involve substantial search spaces [87].

Relational learning techniques have been adapted to the schematic domain and promise to automate the process of integrating new patterns and building relational representations for efficient recognition. A system described in Messmer and Bunke [208] utilises an efficient polynomial graph matching technique to define and match symbols and schematic diagrams. Amin, Sammut and Sum [9] present a system for recognising Chinese characters based on the FOIL program [271] which is capable of differentiating very large numbers of symbols.

In the CLARET application, an environment is developed in which the programmer constructs interpretation hierarchies by manipulating the input and defining the context, of different hand-drawn schematic symbols and diagrams.

### **3.7.1 Operational Schema**

An operational schema has been devised in which the input of hand-drawn schematic data is interpreted interactively during the process of defining patterns and concepts. Interpretations are formed based on the context of particular applications. Schematic applications include the editing of mathematical equations, the annotation of photographic images, the production of multimedia, or the definition and manipulation of architectural designs.

The user inputs different symbols and diagrams using an on-line drawing device which records position and pressure information. The interpretations found using CLARET are then back-projected onto the display in its relative position, scale and rotation to where it was drawn. The interpretation of symbols at different orientations is shown in Figure 3.16. The nature of the information which can be back-projected is quite general and can be in the form of mathematical equations, architectural plans, photographic images, or multiple forms of media relative to the application domain. The

interpretation of mathematical equations is shown in Figure 3.1. Note that the way that users refer to different symbols and images may change quite dramatically. The notion of back projection is related to the process of augmenting the reality (domain) or environment in which the user is operating [266].

The system uses an interactive correction and new schema entry method, and there is no learning phase distinct from the run phase. The recognition mode is always active and the user simply inputs the identity of new symbols when the system fails to identify an unseen or mis-classified example. New symbols are incrementally learned and the system does not require the exhaustive annotation of each (different) symbol used. The vocabulary of the system corresponds to the kinds of hierarchical concepts defined by the user. Many spatial patterns and designs can be synthesised by simply drawing examples of them.

Generalisations of symbols are built dynamically from stored examples during the process of interpretation. This is in contrast to systems in which the designer has to hand-code explicit descriptions of patterns and designs, in an intermediate language.

Unconstrained schematic drawing is allowed, that is, symbols can be input with no constraints on rotation, scale, or positioning. Handwritten “components” are defined from pen-down condition to pen-up. Input of contextual information such as menu selection and the completion of drawing, is enacted through either the use of the pen buttons or the icons. Pattern identity is made by selecting one of the images tiled at the top of the application window.

In this application a Kurta® XGT digitising tablet with a cordless pressure pen [171] is used to collect data on both positional and pressure information. This offers more information for matching and gives better visual feedback cues to the user. The data comprises x,y (position) and z (pressure) sequences defined by pen-down and pen-up occurrences (see Figure 3.15(a)).

A hierarchical segmentation technique is used to segment the on-line data based on the polygonal approximation techniques of Lowe [186] and Rocha and Pavlidis [289]. The approach amounts to determining successive approximations to the line integral by splitting lines into segments which minimise the path length difference. The algorithm has been shown to be straightforward to implement and offers good approximations to curves in terms of integral analysis (see Rosin and West [294]). The depth (level) in the hierarchical segmentation tree corresponds to scale (see Figure 3.15(b)). The generation of new tree nodes is ordered and an A\* search [230, 285, 364] proceeds in terms of greatest reduction in least mean squared error to the given line from the approximating segments. A threshold of approximation to the line integral or maximum number of segments is used. In these experiments, an approximation of either 95% difference in path length or a

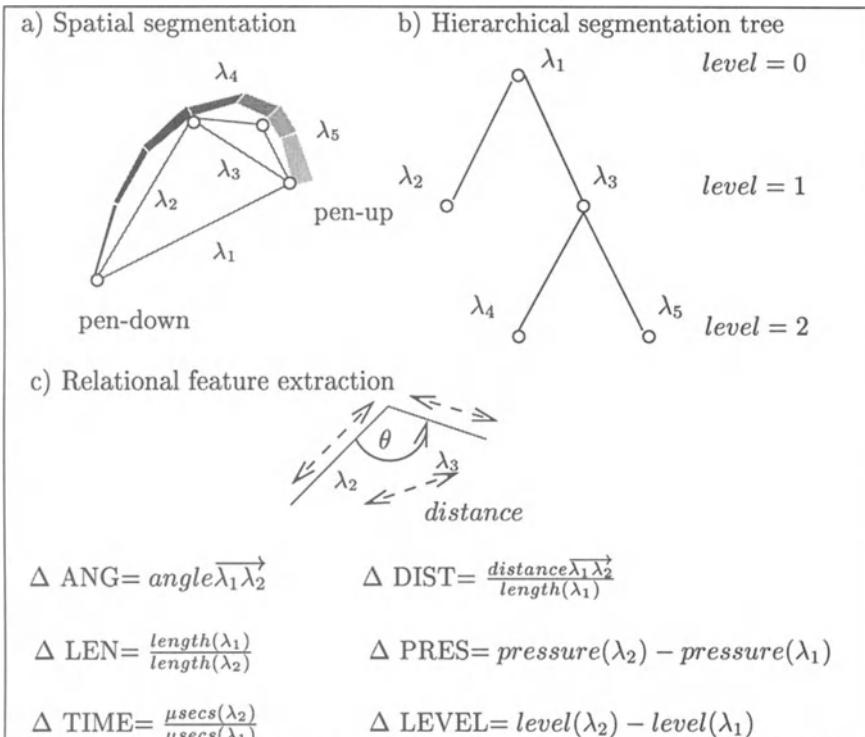


Figure 3.15: Spatial segmentation and relational feature extraction: Successive polygonal approximations are made to the curve in (a) resulting in a labelled hierarchical segmentation tree shown in (b). Features are then extracted from segments in (c) which are invariant to rotation, scale and shift and relations are generated of the form  $(\overrightarrow{\lambda_1 \lambda_2}, \Delta \text{LEVEL}, \Delta \text{TIME}, \Delta \text{DIST}, \Delta \text{ANG}, \Delta \text{LEN}, \Delta \text{PRES})$ .

maximum of 20 segments was accepted.

Segmentation techniques can also utilise more sophisticated features such as curvature, scale-space or multiple scale-space to improve classification performance (see Mokhtarian and Mackworth [217] or Rosin [295]). The Lowe algorithm was chosen for its simplicity and it is sufficient for demonstrating and empirically comparing relational learning techniques based on the *available* features.

The features extracted from relationships between patterns have been chosen so that they are invariant to rotation, scale and shift as much as possible. They include relative angle, relative distance, relative scale and relative pressure as shown in Figure 3.15(c). In addition, the difference in hierarchical level of line segments, and relative time difference is extracted. Features are normalised with respect to the characteristic of the segments



Figure 3.16: Rotation, scale and shift invariance: Symbols are shown drawn at different orientations. Matched interpretations are projected back by calculating rotation, scale and shift orientation parameters.

themselves to allow for scale invariance, relative distance is normalised by the length of the line segments.

After an input symbol is received, a search is initiated over all the known patterns using the CLARET interpretation algorithm. The finite interpretation is returned for one of the known patterns including a solution

for the mappings of segments in the known pattern to segments in the new pattern (see Figure 3.17). Labelled rules are generated in the form:

$$r_i(\overrightarrow{\lambda_J \lambda_K}, F_1, F_2, \dots) \leftarrow \\ r_j(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots), F_1 \geq \text{attribute\_value}_1, F_1 \leq \text{attribute\_value}_2 \dots$$

where  $\lambda_I$  are intensionally defined label variables and  $F_i$  are features with values  $\text{attribute\_value}_i$ .

(a) A finite interpretation

$$r_2(\overrightarrow{\lambda_J \lambda_K}, \Delta LEVEL, \Delta TIME, \Delta DIST, \Delta ANG, \Delta LEN, \Delta PRES) \leftarrow \\ r_1(\overrightarrow{\lambda_I \lambda_J}, \dots), \\ \Delta LEVEL \geq -2.00, \Delta LEVEL \leq -1.00, \\ \Delta TIME \geq 1.00, \Delta TIME \leq 6.99, \\ \Delta DIST \geq 0.06, \Delta DIST \leq 10.00, \\ \Delta ANG \geq -6.07, \Delta ANG \leq 3.42, \\ \Delta LEN \geq 0.05, \Delta LEN \leq 6.00, \\ \Delta PRES \geq 3.40, \Delta PRES \leq 254.00.$$

(b)

<u>Rule Statistics:</u>	<u>Path distribution:</u>	<u>Label Solution:</u>
Total rules: 100	length, number	$\mu_0 \mapsto \lambda_0$
Path nodes: 1942	1, 546	$\mu_1 \mapsto \lambda_1 \parallel \lambda_2$
Relational levels: 4	2, 637	$\mu_2 \mapsto \lambda_3$
Rule entropy: 4.06	3, 488	$\mu_3 \mapsto \lambda_4$
Search expansions: 83	4, 253	$\mu_4 \mapsto \lambda_5$

Relational Evidence: Evidence=0.77

Interpretation Statistics: Position=(338,176) Angle=0.16 (radians)  
Xscale=45 Yscale=51 Angle=0.16 (radians)

Figure 3.17: A rule from a finite interpretation output by CLARET is shown for matching the handwritten symbol,  $\sum$ , from the “Scientific Symbols” application. A first order rule is shown in (a). Statistical information is shown in (b) for rule generation, path distribution, label mapping solution and interpretation orientation. The input (unknown) symbol has less labelled parts ( $\mu_I$ ) than the known symbol ( $\lambda_I$ ), and as a result of unresolved label mapping  $\mu_1 \mapsto \lambda_1 \parallel \lambda_2$ , the evidence weight for the interpretation is 0.77 (subgraphs of rules with highest evidence).

## 3.8 Performance Comparison

The CLARET system is demonstrated and empirically tested using schematic applications involving mathematical symbols and equations or chemical symbols and formulae. These applications involve recognising both individual patterns (symbols) and hierarchies of patterns (mathematical symbols and chemical formulae). The comparison of the systems is made in two ways, first, the ability of the system to operate using a difficult symbols example and second, the utilisation of orientation information for the task of recognising different chemical formulae.

### 3.8.1 Scientific Symbols

The efficiency of CLARET is compared with the first order inductive logic learning system FOIL [271] using the set of 128 scientific symbols from the American Mathematical society extension to Latex [124]. The results for three different users were averaged for the experiment and the task was to use the system to learn symbols for each user separately.

The aim of the experiment was to evaluate the capabilities of FOIL and CLARET for the incremental learning of spatial symbols from unprocessed input data. User independence was not tested here, because the system aims to learn a variety of usage patterns by users for quite different applications.

The FOIL algorithm is based on similar inductive logic techniques as CLARET. Two loops are used for learning sets of rules which can discriminate between positive and negative pattern examples. The outer loop learns rules one at a time, until all patterns (positive examples) are instantiated (covered). The inner loop builds rules by adding terms or numeric expressions (literals) to the body of rules (the right-hand side) until no negative examples are instantiated. An entropy-based gain is used to select literals to add to new rules (see Algorithm 3).

The FOIL system does not employ domain constraints and in this sense is general in that there is no specific rule expansion strategy. FOIL relies primarily on its entropy gain measure to select the next term or numeric expression for rule expansion. CLARET, on the other hand, has a specific rule expansion strategy which treats attribute indexing and part indexing separately during the rule generation process.

FOIL was unable to produce relational rules from the labelled and attributed data used in our experiments. It failed in two ways. First, it produced rules of zero length due to the fact that entropy gain heuristic failed when all candidate literals had zero gain. Second, it produced rules containing numerical expressions for attribute values that contained grounded label terms (values instead of variables). This is unsatisfactory, because unlike attribute values which are significant in their value, label

**Algorithm 3** The FOIL algorithm consists of two loops for learning sets of rules which can discriminate between positive and negative pattern examples. The outer loop learns rules one at a time, until all patterns (positive examples) are instantiated (covered). The inner loop builds rules by adding terms or numeric expressions (literals) to the body of rules (the right-hand side) until no negative examples are instantiated. An entropy-based gain is used to select literals to add to new rules.

**FOIL algorithm:**

```

while there exist positive examples not covered by learned rules
    • create new rule with nil body.
    • while new rule is instantiated by negative examples
        – generate new candidate terms and numeric expressions (literals)
          based on examples
        – select best literal using foil entropy gain
        – add best literal to body (right hand side) of new rule
    • add new rule to learned rules

```

identifiers are important in how they connect relational structures.

In order for FOIL to learn the symbols, sets of features needed to be pre-compiled into propositions which combined two relations each, and only contained attribute values, but no labels. We attempted to generate propositions with combinations of more than two relations but the computational complexity of induction over the large number of resulting propositions (tertiary or quaternary combinations) was intractable. It should be pointed out that FOIL was invoked in propositional form only and it's performance is expected to be limited under these conditions.

**Definition 9** A pre-compiled proposition consists of a combination of unlabelled features,

$$(F_1, F_2, \dots, F_n, G_1, G_2, \dots, G_n)$$

where  $F_i$  is obtained from labelled relation  $(\overrightarrow{\lambda_I \lambda_J}, F_1, F_2, \dots, F_n)$  and  $G_i$  is from labelled relation  $(\overrightarrow{\lambda_J \lambda_K}, G_1, G_2, \dots, G_n)$  and the two relations share label  $\lambda_J$ .

From the pre-compiled propositions, FOIL generates unlabelled first order rules of the form

$$\begin{aligned}
 r_i(F_1, F_2, \dots, F_n, G_1, G_2, \dots, G_n) \leftarrow \\
 F_1 \geq \text{attribute\_value}_1, F_1 \leq \text{attribute\_value}_2 \dots \quad (3.9)
 \end{aligned}$$

```

 $r_1(\Delta LEVEL1, \Delta TIME1, \Delta DIST1, \Delta ANG1, \Delta LEN1, \Delta PRES1,$ 
 $\Delta LEVEL2, \Delta TIME2, \Delta DIST2, \Delta ANG2, \Delta LEN2, \Delta PRES2) \leftarrow$ 
 $\Delta LEVEL \geq -2.00, \Delta LEVEL \leq -1.00,$ 
 $\Delta TIME \geq 1.00, \Delta TIME \leq 6.99,$ 
 $\Delta DIST \geq 0.06, \Delta DIST \leq 10.00,$ 
 $\Delta ANG \geq -6.07, \Delta ANG \leq 3.42,$ 
 $\Delta LEN \geq 0.05, \Delta LEN \leq 6.00,$ 
 $\Delta PRES \geq 3.40, \Delta PRES \leq 254.00.$ 

```

Figure 3.18: An example of the unlabelled first order rules output by FOIL are shown for matching the handwritten symbol,  $\Sigma$ , from the “Scientific Symbols” application. Rules are generated in the form shown in Equation 3.9 from pre-compiled propositions containing feature attributes from two relations.

An example of the rules produced by FOIL are shown in Figure 3.18.

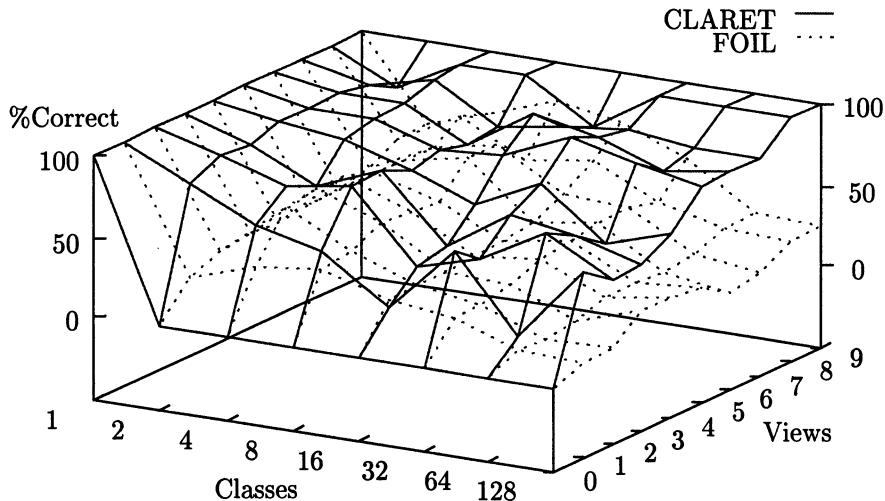
For both FOIL and CLARET, two performance characteristics were tested, classification performance and computational complexity. For classification performance, the systems were measured for different numbers of symbols in logarithmically increasing numbers up to 128. The subjects were asked to input new symbols incrementally. Initially, three known examples of a symbol were input as known patterns in an upright orientation. Then the matching of new, unseen symbols was tested at different orientations and scales, and miss-classified symbols were added as known patterns until six had been correctly matched.

The user input different numbers of views for different symbols, in order for the system to learn to differentiate symbols according to their similarity. On average, 1536 input symbols were input for each subject and the experiment was conducted over approximately two hours. The classification performance was measured as a function of the number of symbols (patterns) and the numbers of input examples of each symbol (views) that were required. This creates a classification *surface* and the slope of this surface correspond to the ability of the systems to incrementally learn new symbols. As the number of patterns increases, the difficulty of generalisation increases and more views of each symbol are required to reliably differentiate unknown symbols. (see Figure 3.19(a)). Trials were conducted by reducing the number of views required for reliable classification, in single steps down to zero views, and the classification performance was tested using the reduced views. The number of views shown in Figure 3.19 represents the average number of views for each symbol.

For each subject, the tiling of symbols at the top of the application window was changed. This normalised the effect of symbol order on incremental learning. All the segmented input data for symbols was saved during the experiments for use in comparison trials.

The segmented input data from the trials was partitioned into known

(a) “Scientific Symbols” classification; difficulty(entropy=4.06 rules=100)



(b) “Scientific symbols” computation; difficulty(entropy=4.06 rules=100)

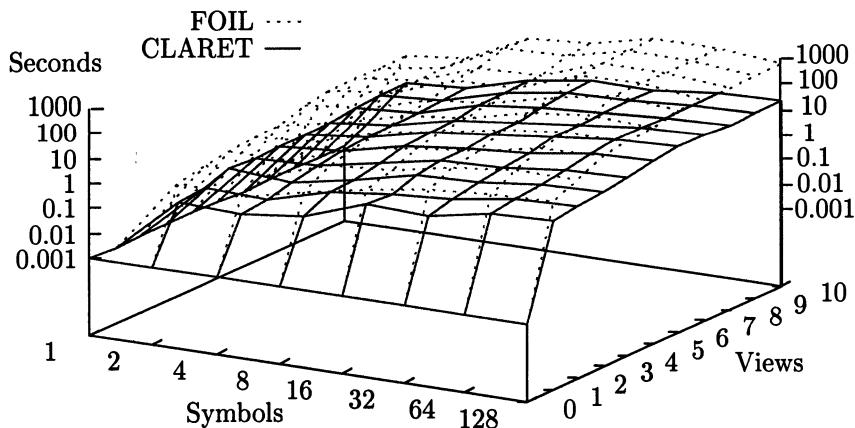


Figure 3.19: “Scientific symbols” performance: Classification results are shown in (a) for CLARET and FOIL for different numbers of symbols (patterns) and input training views (examples). The slope of the classification surface corresponds to the ability of the systems to incrementally learn new symbols. The logarithm of elapsed time in seconds is shown in (b) for the CLARET and FOIL systems for different numbers of symbols (patterns) and input training views (examples).

(training) patterns and unknown (new unseen example) patterns for testing, for each set of symbols for both CLARET and FOIL. The classification performance for FOIL was measured by the ratio of correctly classified pre-compiled propositions. It can be seen that CLARET achieves a high performance level with substantially smaller numbers of known pattern views than FOIL, that is, CLARET learns *faster*. It should be pointed out that the incremental scheme used in this application defines the number of symbols required for classification by adding new symbols until reliable classification is achieved (defined by a pre-determined number of successful matches in a row). This means that the system adapts to different numbers of symbols relative to the current state of the system. This is in contrast to random sample partitioning, which does not satisfy the sufficiency of known patterns criterion and therefore typically results in lower classification performance.

The computational complexity of the learning systems was compared by measuring the elapsed time required to match each input symbol. The incremental learning scheme requires that for each symbol matched, both the rule generation (induction over the current set of input patterns) and the rule instantiation involved are measured (see Figure 3.19(b)).

The experiments were conducted using a Silicon Graphics® workstation utilising the MIPS® R4400 Processor Chip (with MIPS R4010 Floating Point Chip) operating at a clock speed of 150MHz and using 96 Megabytes of main memory. Both CLARET and FOIL are written in the C programming language and were compiled under the Gcc compiler using full optimisation.

CLARET was able to classify symbols in seconds, whereas the time required by FOIL was much greater; minutes or hours when large numbers of symbols were used. The faster performance of the CLARET system can be explained in terms of two factors. First, CLARET generates rules with increasing arity on the basis of need. On the other hand, FOIL required relations to be pre-compiled into combinations of relations which results in a quadratic increase in the size of input data. Second, CLARET uses a dynamic programming strategy to prune search and the worst-case computational complexity is governed by the number of rules, not the number of segmented pattern parts.

The difficulty of a classification task is proportional to the number of rules needed for the generation of interpretations (generalisations) as well as rule entropy. Larger numbers of rules correspond to the necessity for more partitions and relational extensions. If the number of rules are the same for two matching problems and the average entropy is higher in one problem, then matching is harder in the problem with higher entropy because parts have features which are less segregated in feature space. CLARET had difficulty matching some symbols where the segmentation results were very similar. For example the symbols  $\oplus$  and  $\boxplus$  result in very similar polygonal



Figure 3.20: Distortion and missing parts: **Top:** Symbols are shown drawn with increasing distortion and missing components. **Bottom:** Symbols are drawn with distortion, together with different stroke orders.

approximations. However, this was somewhat compensated for by the way subjects drew the symbols using different stroke orders, speed and pressure. The performance of the system under distortion and missing parts can be seen in Figure 3.20.

### 3.8.2 Interpreting Chemical Symbols

In this application, the utilisation of orientation information is evaluated for the tasks of recognising different chemical equations including scenarios of different symbols. An example of such a scenario is the series of chemical equations comprising different organic carbon bases with attached groups, as shown in Figure 3.21.

The aim of this experiment was to demonstrate the ability of the system to utilise orientation information in analysing scenarios. Here, the CLARET system was not used to attach labels to the entire scenarios (chemical compounds), but to differentiate between multiple symbols via the utilisation of expected orientation information.

Because of the more complex classification task required when sets of symbols are involved, the classification performance of symbols was measured in the context of relationships to other symbols. If the expected orientation of symbols is known in chemical compounds, an improvement is possible by checking the orientation of candidate matches to unknown symbols.

For this task, subjects were asked to construct different chemical compound equations using the allowed organic chemical constructor symbols (see bottom of Figure 3.21). Trials were conducted, using the same strategy as in the “Scientific Symbols” experiment, except that the orientation of symbols had to be within a certain range (28 degrees was chosen, based on experiments with different values). The results represent the classification improvement obtained by rejecting matches which are outside this range (see Figure 3.22). The “Chemicals” symbols are less difficult to classify than the “Scientific Symbols”, as is reflected in the number of rules required, and the average rule entropy. This is seen for both CLARET and FOIL classification performance. In total, 32 symbols were used for the chemical application.

The facility of recognising symbols out of collections of symbols can also be demonstrated. The CLARET system is asked to look for a pre-determined number of symbols in the segmented input (2–4 in this experiment). The system then attempts to match one of the known symbols to the segmented input drawing and identifies the first symbol, calculates orientation and back-projects it on to the canvas. Next, the matched segments are removed and the remaining segments are re-issued to the CLARET search to look for the next symbol and so on (see Figure 3.23).

The use of the union-find algorithm to find connected sets of matched patterns is an important component of this capability. This will only accept matchings which have segments that are connected together, in some way, via paths. One of the side effects of this is that classification performance degrades when symbols overlap because difference relationships are included in the planar graphs generated.

The ways in which CLARET can utilise orientation information has

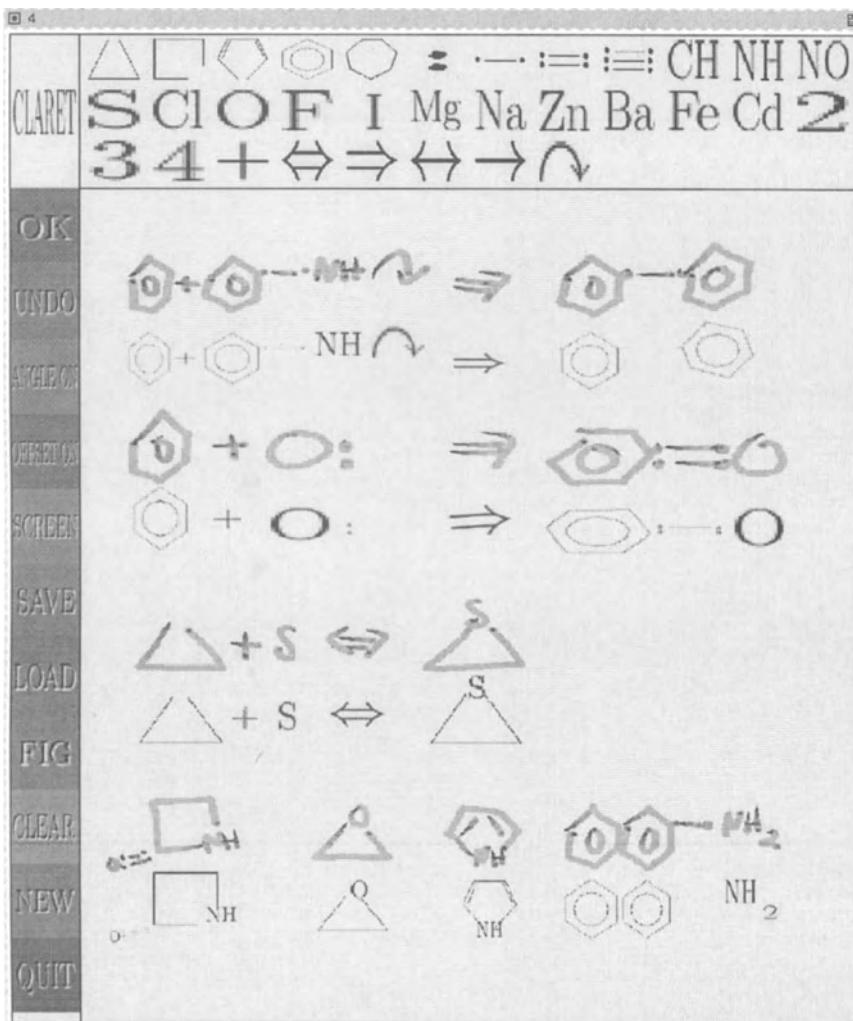
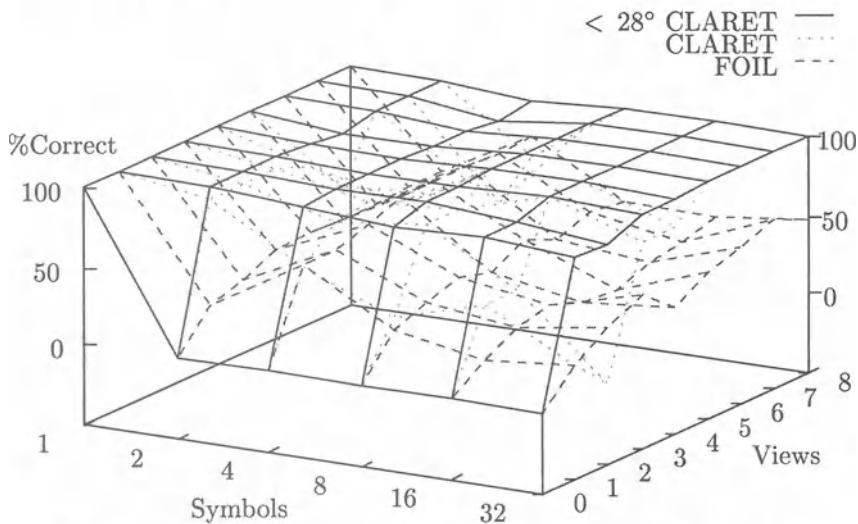


Figure 3.21: “Chemicals” task: The “Chemicals” application is shown for different organic constructor symbols including carbon bases of different structure, end groups, bonds and operators. **Top:** Several chemical equations are shown, where symbols are entered one at a time. CLARET successively matches each symbol and back projects the result. **Bottom:** The task of drawing chemical formulae is shown for azetidin-2-one, oxirane, pyrrole and  $\beta$ -aminonaphthalene (left to right).

(a) "Chemicals" classification; difficulty(entropy=2.68 rules=100)



(b) "Chemicals" computation; difficulty(entropy=2.68 rules=100)

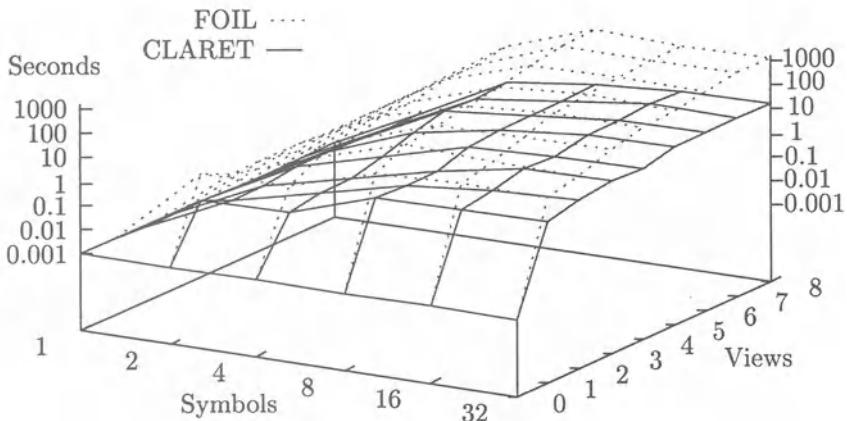


Figure 3.22: "Chemicals" performance: Classification performance is shown in (a) for CLARET and FOIL for the "Chemicals" application for different numbers of chemical symbols. The performance of CLARET is improved using the expected orientation constraint to reject matching candidates outside a threshold of 28 degrees. The logarithm of elapsed time in seconds is shown in (b) for the CLARET and FOIL systems, for different numbers of symbols (patterns) and input training views (examples).

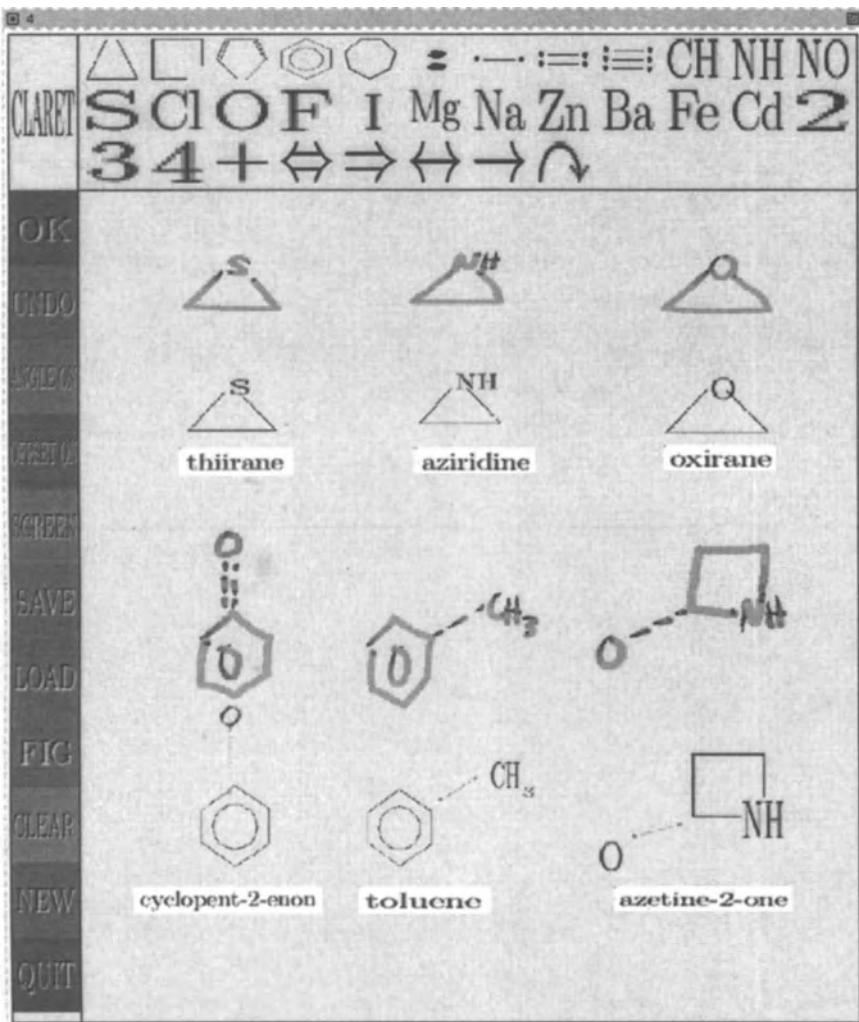


Figure 3.23: “Chemicals” scene task: The task of interpreting chemical formulae for organic compounds is shown, where CLARET is used to interpret symbols from multiple symbol input. Scenarios are shown for which CLARET successively matches each symbol and back projects the result. In these cases, the orientation returned can be used to improve the classification performance based on expected relationships between symbols.

been demonstrated and the ability to differentiate between multiple input symbols. In this implementation, full scenario interpretation has not yet been invoked and the relationship between the individual symbols is not efficiently utilised. However, the hierarchical recognition of sets of symbols remains a promising area for further work (see Section 3.9).

### 3.9 Conclusion

This work has consolidated theoretical and procedural issues involved in applying machine learning to computer vision for efficient spatial interpretation. Two different approaches to evidential learning have been combined for generalising relational data structures.

Relational evidence theory integrates information theoretic methods from decision trees with graph matching methods from constraint interpretation. An evidence-based framework for evaluating and updating relational representations has been presented, which is suitable for spatial applications.

A consolidated learning algorithm based on relational evidence theory (CLARET), has been presented which integrates Rulegraph matching with rule generation techniques from inductive logic programming. The approach utilises the relational constraints in spatial applications to optimise the representational hierarchies and search strategies used during learning. CLARET incorporates an interpretation stage into the generalisation process where compatibilities between rules and dependencies between different rule instantiations (label compatibilities) are taken into account.

An on-line schematic and symbol recognition application has been demonstrated for learning to recognise symbols and patterns invariant to rotation, scale and shift. Performance comparisons with other inductive logic programming techniques are given, which measure classification performance, computational efficiency and the human factors involved in incrementally training the system.

A systematic approach to schematic interpretation has been demonstrated for the on-line schematic diagram and symbol recognition system based on the consolidated learning algorithm (CLARET). It has demonstrated the benefit of utilising inductive logic programming and graph matching methods in this domain and the importance of adapting these techniques to utilise spatial constraints.

There have been a number of systems developed for analysing relational representations during learning. Minimum message length techniques are utilised by the FOCL [247, 248] and MOBAL [369] systems. Probabilistic stochastic methods have also been developed (see stochastic inductive logic programs of Muggleton [222]). Text recognition has also been performed using Hidden Markov Models to encode relational structures, but only of

single arity. In optical character recognition, the use of transition probabilities is used to limit the number of character candidates during line-by-line recognition [37].

In contrast to the assumption of independence made by entropy and minimum message length methods, CLARET incorporates an interpretation stage into the generalisation process. This is based on the compatibilities between rules and the dependencies between different rule instantiations (label compatibilities) are taken into account (see Section 3.4).

Improved classification performance has been demonstrated for the interpretation of unspecific, on-line handwritten schematic symbols. This is explained by the explicit representation of compatibilities between rules at arbitrary (higher) orders, for the multi-labelled patterns encountered in the schematic application. In addition, the search is made tractable by reducing the cardinality of the search space with the use of multi-labelled Rulegraphs and the application of the dynamic programming principle using relational evidence measures to prune the search. The ordering of subgraphs of rules (interpretations) in CLARET allows for an explicit search strategy to be used. A major strength of the CLARET system is that new symbols may be incrementally entered without interruption to the system. On the other hand, the system has not been tested in other application domains and further testing is required to confirm the generality of the techniques.

The comparison with alternative graph matching techniques should also be conducted, such as the polynomial subgraph isomorphism detection technique of Messmer and Bunke [207]. A system described in [208] utilises an efficient polynomial graph matching technique to define and match symbols and schematic diagrams.

There are a number of other ways to improve the performance of spatial interpretation performance. Recognition of very large numbers of patterns, in the absence of any other constraints, is an ill-posed problem in the sense that there are typically restrictions on the possible ways that sets of symbols are arranged (such as orientations). In CLARET, the interpretation of hierarchies of patterns (scenarios) needs further development.

An extension to conditional rule generation (CRG) has been conducted by Bischof and Caelli [31], which investigates the use of constraints in the scenes in the instantiation of conditional rules. The system—Scene Understanding by Rule Evaluation (SURE)—is also suitable for recognising arrangements of multiple patterns or montages. In contrast to the SURE system, relational evidence theory investigates the use of graph matching methods to utilise constraints in the patterns themselves, rather than the scene.

Investigation of the interaction between the matching system and the segmentation system may prove beneficial. Relational learning systems rely on the reliable decomposition of patterns. Using a hierarchical segmentation procedure, the most significant number of parts for differen-

tiating different patterns could be established *for each symbol*. Such an approach would be consistent with the use of more sophisticated curvature, scale-space, or multiple scale-space features (see Mokhtarian and Mackworth [217] or Rosin [295])

The significance of this work is twofold. Firstly, it extends the applicability of machine learning theories and algorithms into new domains. The techniques complement the image query and retrieval tools currently available in computer vision by offering additional ways of recognising and manipulating spatial information. Secondly, the development of a working schematic system allows the evaluation of efficient spatial interpretation techniques and places emphasis on the dialogue between the user and the technology.

# **Chapter 4**

# **Cite—Scene Understanding and Object Recognition**

**Craig Dillon and Terry Caelli**

## **Abstract**

This chapter presents a new approach to image interpretation which can produce hierarchical descriptions of visually sensed scenes based on an incrementally learnt hierarchical knowledge base. Multiple segmentation and labelling hypotheses are generated with local constraint satisfaction being achieved through a hierarchical form of relaxation labelling. The traditionally unidirectional segmentation-matching process is recast into a dynamic closed-loop system where the current interpretation state is used to drive the lower level image processing functions. The theory presented in this chapter is applied to a new object recognition and scene understanding system called *Cite* which is described in detail.

## 4.1 Recent Systems and Proposed Theory

Learning about and understanding the environment is critically important to a number of existing and emerging challenges including autonomous navigation, environmental monitoring, surveillance and content specific image queries. Passively viewing the external world is a consistent and informative way automata can obtain local area data. A common passive modality is through visual sources such as CCD cameras. This chapter presents a theory of object recognition and scene understanding which could enable an autonomous robot or other visual processing systems to obtain a detailed description of complex natural and person-made worlds and environments.

The earliest forms of object recognition dealt with the task of identifying single isolated objects in one image. This remains the most common form of object recognition today, and despite the increase in complexity of the methods, the results have not improved greatly. Early scene understanding systems typically dealt with person-perspective or map views of natural scenes containing objects such as buildings, trees, and roads. Within the object recognition research, considerable effort has been spent on solving the part indexing, part clique and knowledge acquisition problems. By comparison, scene understanding systems tend to be pixel-based and are concerned more with identifying probable labels for small image regions rather than locating and labelling entire objects.

Two recent scene understanding systems which are comparable to *Cite* are SCHEMA [89] and SIGMA [198].

### SCHEMA

The SCHEMA system is a major attempt at a general purpose knowledge based scene interpretation system. The fundamental aspect of SCHEMA that sets it apart from most other scene interpretation systems is that both knowledge and computation are partitioned at a coarse-grained semantic level. The knowledge base contains a set of schemas, each of which is designed to recognise one particular class of object. This differs from most machine vision approaches which usually focus on one representation and one matching strategy.

Each schema has access to a global *blackboard* which contains the current interpretation of the scene, and can post and receive messages to this global space asynchronously. An instance of the appropriate schema is generated for each hypothesised instance of the object in the scene. Each schema instance has access to what are termed *knowledge sources*. The knowledge sources, which operate at a number of different levels, provide analysis of the image and resultant image structures. For example, there are lower level feature extraction and segmentation knowledge sources, and higher level knowledge sources covering initial hypothesis generation, token clustering and knowledge driven re-segmentation.

The knowledge base in SCHEMA is hand coded by a knowledge engineer in declarative language which covers the *endorsement space*, *confidence function* and *control strategies* for each schema. The endorsement space contains structures which can accumulate evidence for an instance of the object represented by the schema, and the confidence function describes how to combine these various endorsements. The control strategy section of each schema determines how the support space is searched by providing flow control over the sequence of knowledge sources accessed by the schema. Although demonstrated to be successful on natural images with a small number (15) of objects in the knowledge base, the requirement for each schema to be *programmed* requires the presence and attention of an expert.

### SIGMA

The SIGMA system is a complete image understanding system designed for aerial image analysis. It contains three modules for low level vision, model selection and geometric reasoning, as well as a query module through which the user interacts. These modules are interconnected such that top-down and bottom-up vision processes are closely integrated. The interpretation within SIGMA is in the form of a *part-of* hierarchy with image features at the leaf nodes and spatial relations describing the higher level structure.

SIGMA's knowledge base is object oriented in the sense that instances of an object are created (instantiated) dynamically from base object classes. Each object has three main knowledge components; unary properties of the object, relationships with other objects, and control information to guide the analysis process. The control structures are triplets of the form (condition, hypothesis, action).

After an initial segmentation, seed hypotheses are generated and stored in the iconic/symbolic database which represents the instantaneous scene interpretation. These initial hypotheses result in bottom-up hypothesis generation of higher level object instances. These, in turn, result in the top-down hypothesis generation of missing parts. The geometric reasoning expert iterates the interpretation cycle through hypothesis generation, evidence accumulation and the resolution of the most reliable composite hypothesis. This cycle continues until no new hypotheses are generated and the resulting global interpretation is stable.

SIGMA contains a sophisticated low level vision module which performs goal directed image segmentation. A *goal* in this system is expressed as a set of constraints and properties that the solution must satisfy, some of which specify the environment in which the solution is to be found. One strength of this system is the ability to search the image at a very low level for expected image features.

SIGMA has been demonstrated on aerial images of new housing develop-

ments and robustness to shadows and poor image contrast has been shown. As with SCHEMA, the knowledge base for SIGMA is hand-coded by an expert, and the system as a whole has not been demonstrated to recognise more than a few different object classes.

#### 4.1.1 Proposed Theory

This chapter describes *Cite* which specifically aims to extend and improve the current research in object recognition, scene understanding systems, and machine learning as applied to computer vision systems. *Cite* extends the conventional flat knowledge base to a fully hierarchical one with no depth limitation. Multiple hypotheses are generated for each scene element, and these are resolved using a hierarchical extension to relaxation labelling. Traditional feed-forward segmentation is augmented with knowledge driven re-segmentation which closes the control loop on low level vision processes. *Cite* also extends the typical “learn-phase then run-phase” operational environment with robust incremental learning algorithms which build and improve the knowledge base after each scene has been fully analysed.

An architectural diagram of *Cite* is shown in Figure 4.1. The numbers beside each function block represent the approximate order of operation for each operator. An initial segmentation (1) of the image causes the unary feature calculator to compute features for each of the low level regions (2). These features are then matched with the knowledge base (3) to provide initial labelling hypotheses which are represented in the indexing structure called the *scene interpretation*. Clique resolving and hierarchical binary matching then occur on these initial hypotheses (5) using binary features calculated from the hierarchical segmentation (4). The higher level scene hypotheses are added into the scene interpretation structure, and hierarchical relaxation labelling begins to resolve the multiple ambiguous labels for each object (6).

As the labels begin to resolve, nodes are individually re-segmented (7) using parameters stored in the knowledge base. These re-segments replace the initial segmentations in the visual interpretation structure, resulting in a repeat of the unary and binary feature extraction and matching (stages (2) through (6)). This cycle continues a number of times until the interpretation becomes stable. If *Cite*’s final interpretation is incorrect, the user may chose to incrementally learn the correct object labelling (8) by selecting the incorrectly labelled nodes and the desired knowledge base node. The updated knowledge base is then available as the next scene is viewed.

The relaxation labelling, knowledge driven re-segmentation and hierarchical clique resolving and matching provide very tight closed-loop feedback within *Cite*. The hierarchical knowledge base provides a rich scene description which can include contextual, taxonomic and deep decomposition information. The use of incremental supervised learning provides *Cite* with

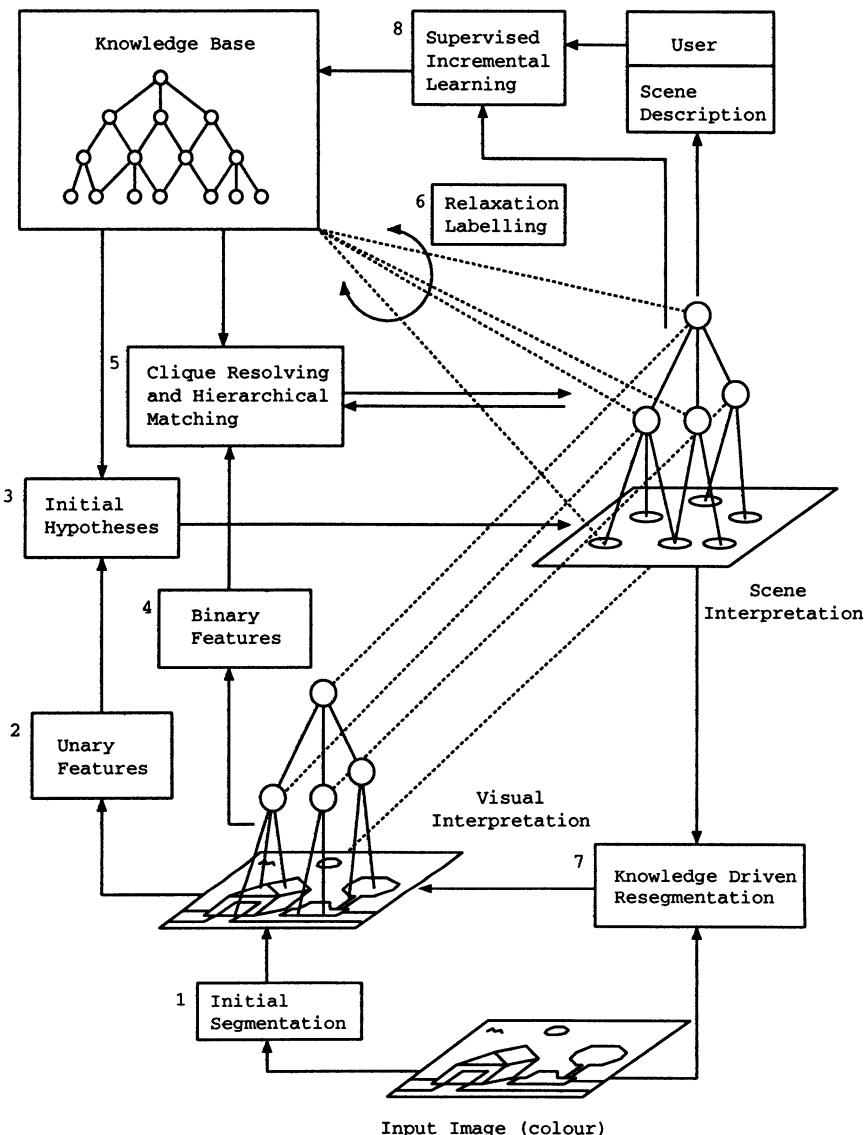


Figure 4.1: Overview of *Cite* Architecture

the ability to increase the descriptive power and accuracy of its analyses, as well as to add new world knowledge as it becomes available, rather than requiring the full set of scene objects to be present during an initial learning phase.

## 4.2 World Knowledge

One of the most important data representations in scene understanding and object recognition systems is the representation of world knowledge. *Cite* represents world knowledge as a semi-restricted graph in which each node represents an object or visual concept which is either a **part-of**, a **view-of** or a **type-of** its parent or parents. Figure 4.2 is a simple example of such a knowledge base. Within each node is stored information about the optimal segmentation, feature extraction and matching algorithms that are used to recognise this object in an image.

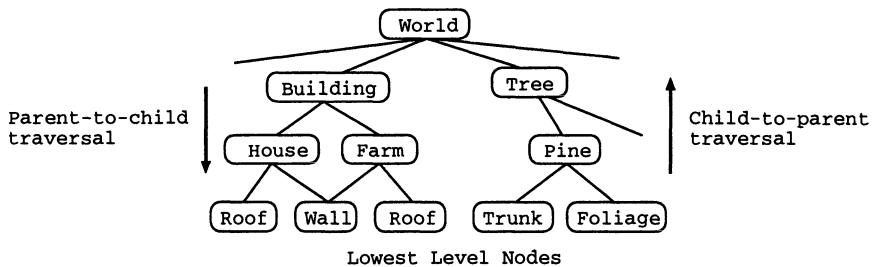


Figure 4.2: Knowledge Base Graph Example

Most previous object recognition systems represent world knowledge as a simple linear list of objects, each of which may be described by specified properties, or as a set of parts each of which can be described. *Cite* can easily represent such a data structure, but because the objective is to use higher level contextual information to describe the scene, the hierarchical knowledge base in *Cite* represents a significant knowledge extension in comparison to conventional object recognition systems.

### 4.2.1 Knowledge Base Structure

The knowledge base in *Cite* is stored as a semi-restricted graph where each node in the graph represents an object or scene element. The objects themselves may be parts of objects or parts of parts of objects, and so on. Each node can have a number of child nodes and a number of parent nodes. The child nodes represent either parts-of, views-of or types-of the parent node. The graph is restricted in that no cycle in the graph can contain only parent-to-child edges or only child-to-parent edges. An important and useful consequence of this is that the graph can be drawn in two dimensions such that for every node its parents are above it and its children are below it.

Any downward traversal of the graph is treating edges as parent-to-child edges, and upward traversals are child-to-parent edges. All knowledge bases

in this chapter are depicted in this manner. An interesting point is that nodes higher up in the knowledge base represent more general knowledge, and nodes lower down in the knowledge base represent more specific knowledge.

The knowledge base is unbounded with the exception that it must have exactly one node with no parents (called “World”). There must also exist a downward traversal path containing only parent-to-child edges connecting the top level “world” node to every other node. Expressed in another way, every node other than the top level node must have at least one parent. A typical knowledge base tends to expand out looking similar to a tree, with the addition of a small number of nodes with multiple parents. Nodes with no children are called leaf nodes and generally represent the smallest atomic scene elements.

The knowledge base in *Cite* decomposes world knowledge in three ways. A scene element or object may be broken into constituent parts, into different views of the object under 3D rotation, or into different specific types of the object. Consequently there are three types of parent-child relationships; part-of, view-of and type-of. Nodes in the knowledge base always contain children with the same parent-child relationship type. A parent whose children have a part-of parent-child relationship is called a part-of node for simplicity, and similarly for view-of and type-of nodes.

In the part-of node, the parent is constructed or composed from instances of the children. For example, “car” would be the parent node, and “wheel”, “door”, “wind-screen” and “side-panel” could be the children. The “wheel” node in turn could have two children “tyre” and “rim”. A forest could be described as being composed of trees, ground, bushes, and maybe a river.

There are two variants of the part-of node. The “constructed” part-of node describes objects which are composed from a strict number of parts where the relationships between those parts is complex but generally rigid. The “consists” part-of node describes scene elements where the parts are collected together less rigidly to form the collective object. A car would be described using a “constructed” part-of node, while a forest would be described using a “consists” part-of node.

Deep knowledge hierarchies are supported by *Cite* so that it is possible to decompose objects over a number of levels. Part-of nodes within such a hierarchy can be a mixture of “constructed” and “consists” nodes. This permits the construction of less well-defined objects such as a “street-scene” which contains well defined objects such as cars, people, and a road.

Each part-of node contains a unary and a binary matching strategy, usually in the form of rules which are evaluated on candidate image regions or groups of image regions. The unary matching strategy is used to recognise the object as a whole, whereas the binary matching strategy is used to recognise the configuration of the constituent parts.

*Cite* represents taxonomies using the type-of node. For complex hierarchical world knowledge, taxonomies represent valuable reductions in data complexity. A “forest” may contain a number of “tree” objects where each “tree” could be of type “gum”, “poplar” or “oak”. In the same knowledge base, “my house” might be represented as a “house” object next to a “gum”. Specifying a simple tree taxonomy in this situation gives the specificity required for “my house” and the generality required for “forest”.

Many objects have a high degree of rotational variance. There is often more similarity between two views of different objects than there is between two different views of the same object. *Cite* represents different views of the same object as children of a common parent. This is achieved through the view-of node.

The view-of representation is not tied explicitly to a geometric viewsphere interpretation, and consequently *Cite* can represent multiple views of objects using only as many views as are required to distinguish the object from other objects in the knowledge base. In most viewsphere representations a predetermined tessellation of the viewsphere is used to calculate the number of views required. These methods all treat the object in isolation to the other objects in the knowledge base, and hence excessive data can be stored.

When dealing with higher level scene elements there are situations where the formal structure of the knowledge base as described in the previous section is too restrictive. In some higher level scene interpretations certain scene elements may be able to exist on their own or as part of another construction. This is represented in *Cite* as an additional parent node with just one child, and is interpreted as implying that the child node may exist either in isolation or as part of a more complex object. These special case nodes are not treated any differently by any of the internal processes in *Cite* and are only occasionally required in the correct representation of higher level scene elements.

Each knowledge base (KB) node contains one list of parents and one list of children. Each element in these lists is another KB node. The parent list is an unweighted list, while the child list is weighted. This is done so that the KB-KB hypothesis weight is stored in only one place. The probability of KB node  $a$  being the parent of KB node  $b$  is the same as the probability of KB node  $b$  being the child of KB node  $a$ .

The part-of node contains three additional components beyond the parent and child lists. The first is a flag indicating whether the type-of node is a “consists” node or a “constructed” node. This flag is used by a number of processes in *Cite* to determine whether all children are expected or required.

The other two additional components are the unary and binary matching strategies used by that node. These matching strategies do not necessarily contain descriptions of the objects, rather they contain processes

which can match the object to a given set of data. Both the unary and binary matching strategies are built by supervised incremental learning. This process is discussed further in Section 4.5.

## 4.3 Interpretation Structures

*Cite* contains a hierarchical segmentation in the form of a semi-restricted graph called the *visual interpretation*. Multiple labelling hypotheses can be generated for each node in the visual interpretation and these are stored in an intermediate structure called the *scene interpretation*.

### 4.3.1 Visual Interpretation

Most object recognition systems work from single level segmentations in which each pixel is assigned to exactly one region from a set of non-overlapping regions. It is generally assumed that by direct (open-loop) segmentation these regions will all represent objects or parts of objects, and the objects can be found from appropriate groupings. In the case of recognising isolated objects this latter grouping is not required as all parts (apart from the background) are assumed to belong to the object in question. *Cite* differs in that it contains a true hierarchical segmentation in which regions are grouped to form larger regions which are grouped to form even larger regions and so on.

There is one type of node in the visual interpretation, called the VI node, which may contain multiple parents and multiple children. Multiple children represent the grouping of smaller regions with common properties or labels into a larger region with a single label. Multiple parents represent the notion of ambiguous set membership, which is important in solving what is called the *clique problem* (grouping problem).

*Cite* uses a process of parallel hypothesis generation to generate multiple likely solutions (local constraints) and then relaxation labelling to propagate these hypotheses to maximise global consistency. This process is discussed in detail in Section 4.6, suffice to say at this point that this approach relies on the ability for the segmentation structure to support multiple ambiguous parent memberships for parts.

There must be at least one child-to-parent and one parent-to-child relationship in every cycle of the VI graph. Similar to the knowledge base graph, the VI graph can be drawn in two dimensions such that for every node its parents are above it and its children are below it. There is exactly one top-level VI node which contains no parents. Each VI node contains an unweighted list of parents and a weighted list of children. This arrangement means that the VI-VI weight is only stored in one place, although for the purposes of analysing the VI structure the hypothesis weight is accessible from either the parent or the child node.

The scene interpretation (SI) structure is an indexing structure which connects the visual interpretation to the knowledge base, and represents the current scene belief state. Each VI node contains a weighted list of hypothesis links to SI nodes and each hypothesis link is interpreted as being the visual support for the given scene interpretation node. Multiple SI hypotheses may be connected to a VI node during top-down hypothesis of the existence of missing or occluded parts. Ambiguous clique memberships may also be represented by multiple SI hypotheses on the given VI node.

Each VI node also contains a list of pixels that the node represents in the original input image. There is no ordering or adjacency limitation placed on these pixels, which means that a VI node can be used to represent multiple non-connected regions. This is particularly useful for representing occluded objects and complex high level scene elements whose constituent parts are not necessarily connected in the image. The final important component of the VI node is that this is data structure where image features are stored when they have been calculated. Unary features are stored in the VI node and binary features are stored in the common parent of the multiple VI nodes between which they are calculated.

### 4.3.2 Scene Interpretation

Most object recognition systems work only from single images of single or small numbers of different objects. The internal representation tends to be a list of parts, each with one label which represents the most common object part. In more sophisticated systems, there may be multiple labels for each part or a first level of grouping parts together to form objects, each of which has a label. In these systems, the knowledge base is shallow and the data representation for hypotheses is a straight forward connection from each part to a node in the knowledge base.

This direct connection of regions to labels is unfortunately not sufficient for more complex scene analysis. For example, we may know that a region exists but it may belong to two possible objects. In this instance, the region requires two part labelling. Without a further indexing scheme it will not be clear to each of the two parent object labels which of the child labels necessarily supports its own hypothesis. This situation is shown in Figure 4.3. When computing the support for the node labelled VP it is not clear how to interpret the multiple VC-KC1 and VC-KC2 hypothesis labelling. Of course, one could search the parent-child relationships to look for ancestry, but in the generalised graph where the hierarchy may be considerably deeper this form of search becomes very costly.

The problem occurring in Figure 4.3 is that there is no relational information between the various hypotheses. The relational information required is in the form of a compatibility set which groups hypotheses into mutually consistent groups. For a hierarchical scene description these com-

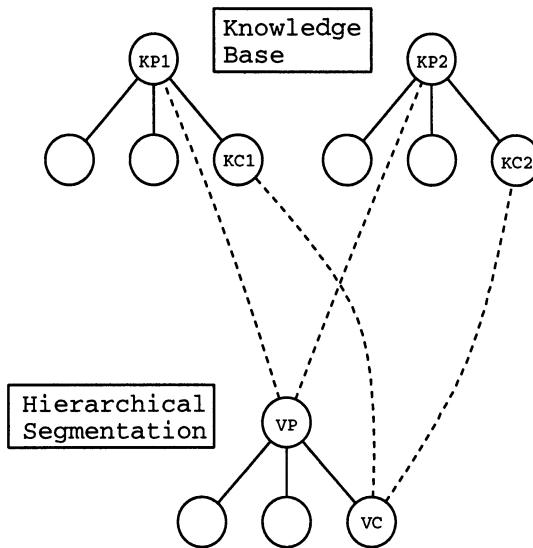


Figure 4.3: Illustration of Inadequacy of Direct Labelling

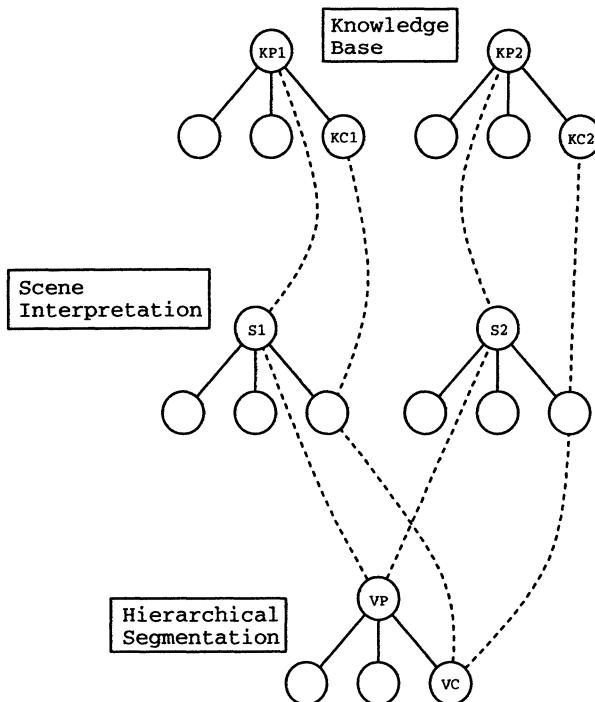
pability sets must also be hierarchical. The most appropriate structure is a graph of similar construction to the knowledge base and visual interpretation, which is called the *scene interpretation*.

The direct labelling example in Figure 4.3 is shown with the intermediate scene interpretation (SI) structure in Figure 4.4. The instantaneous scene description, including all the appropriate ambiguities, can now be read directly from the SI graph. The scene now contains two alternative interpretations, embodied in nodes labelled S1 and S2. These scene interpretation nodes establish the compatibilities between hypotheses such that, for example, the KP1-VP hypothesis cannot now be associated with the KC2-VC hypothesis. When one considers that *Cite* operates over deep hierarchies with a knowledge base where a node may belong to a number of parents, the scene may contain multiple instances of the same object and where multiple views of the scene need to be integrated, the scene interpretation graph becomes a fundamental requirement.

In many ways the scene interpretation graph represents a full analysis of the scene. However, it contains no data other than the hypothesis links to the KB and VI nodes. In this sense, *Cite*'s three data structures represent an elegant and formal separation of a vision system into the knowledge, data and interpretation constituent parts with a generalised hypothesis scheme connecting the three structures.

### Scene Interpretation Structure

There is only one node type in the scene interpretation (SI) graph, called an SI node. Like the knowledge base, there must be at least one child-to-



**Figure 4.4: Hierarchical Compatibility Sets via the Insertion of the Scene Interpretation Structure**

parent and one parent-to-child relationship in every cycle. The SI graph is unbounded with the exception that there is one node with no parents as this represents the entire scene. Leaf nodes with no children represent the lowest level (finest detail) of scene analysis.

Each SI node contains an unweighted list of parents and a weighted list of children, similar to VI nodes. Each SI node also contains a weighted list of VI nodes that are interpreted as being the visual support for that particular scene element. Multiple VI nodes may be connected to a SI node during top-down hypothesis of the existence of missing or occluded parts.

Each SI node contains a weighted list of knowledge base nodes that represent possible object labellings for that SI node. Each weight in the KB node list can be seen as a probability, and as a result of the normalisation process these will add to 1.0. Multiple SI-KB labellings represent an ambiguous classification and these are resolved through a process of relaxation labelling.

### 4.3.3 A Formal Definition of the VI, SI and KB Structures

**Definition 1:** A C-node is a graph vertex containing a set of edges to parent C-nodes and a set of edges to child C-nodes. In each C-node these two sets contain no duplicates and are disjoint.

**Definition 2:** A C-graph is a semi-restricted graph of C-nodes which has one C-node with an empty parent set, and all cycles contain at least one parent-to-child edge and at least one child-to-parent edge.

**Definition 3:** A Part-Of node is a C-node additionally containing a text label, a unary matching strategy and a binary matching strategy.

**Definition 4:** A Type-Of node and a View-Of node are C-nodes additionally containing a text label.

**Definition 5:** The knowledge base (KB) is a C-graph with un-attributed edges where each C-node is a Part-Of, Type-Of or View-Of node.

**Definition 6:** A visual interpretation (VI) node is a C-node additionally containing a set of  $(x,y)$  pixel locations, a set of unary features, a set of binary features, a segmentation history list, and a set of attributed hypothesis links to the scene interpretation.

**Definition 7:** A VI graph is a C-graph where each edge is attributed with a single weight in the range  $[0,1]$  and each vertex is a VI node.

**Definition 8:** A scene interpretation (SI) node is a C-node additionally containing a set of attributed hypothesis links to KB nodes and a set of attributed hypothesis links to VI nodes.

**Definition 9:** A SI graph is a C-graph where each vertex is a SI node.

**Definition 10:** A unary matching strategy is an algorithm containing the required data and processes to compute the degree of similarity to a VI node, and to adaptively update these data and processes to incorporate positive and negative examples in the form of VI nodes.

**Definition 11:** A binary matching strategy is an algorithm containing the required data and processes to compute the degree of similarity to a set of VI nodes with a common parent, and to adaptively update these data and processes to incorporate positive and negative examples in the form of a set of VI nodes with a common parent.

Cite contains one knowledge base graph as defined in Definition 5. For the current scene, Cite contains one scene interpretation graph as defined in Definition 9, which is renewed as each scene is viewed. For each image of the scene there is one visual interpretation as described in Definition 7.

## 4.4 Operational Overview

Many vision systems have a simple flow-through architecture. Typically this involves an image segmentation stage, followed by a feature extraction stage, followed by a matching stage. In each stage the data flows in one

direction only and there are no other control signals. Some systems, such as BONSAI [111] do have a coarse-grained feedback mechanism that makes corrections to the object hypothesis by image back-projection. *Cite* attempts to further refine these feedback processes, and the resultant complexity requires a more structured description.

By way of introduction, Figure 4.5 shows an example of the three main data structures in *Cite* and the operators which construct and manipulate these data structures. In this section, a simple example is followed through in detail, followed by a brief description of each operator. Figure 4.5 serves primarily as an example road-map for the operation of *Cite*, and Figure 4.1 should be consulted for a description of the temporal data flow sequence.

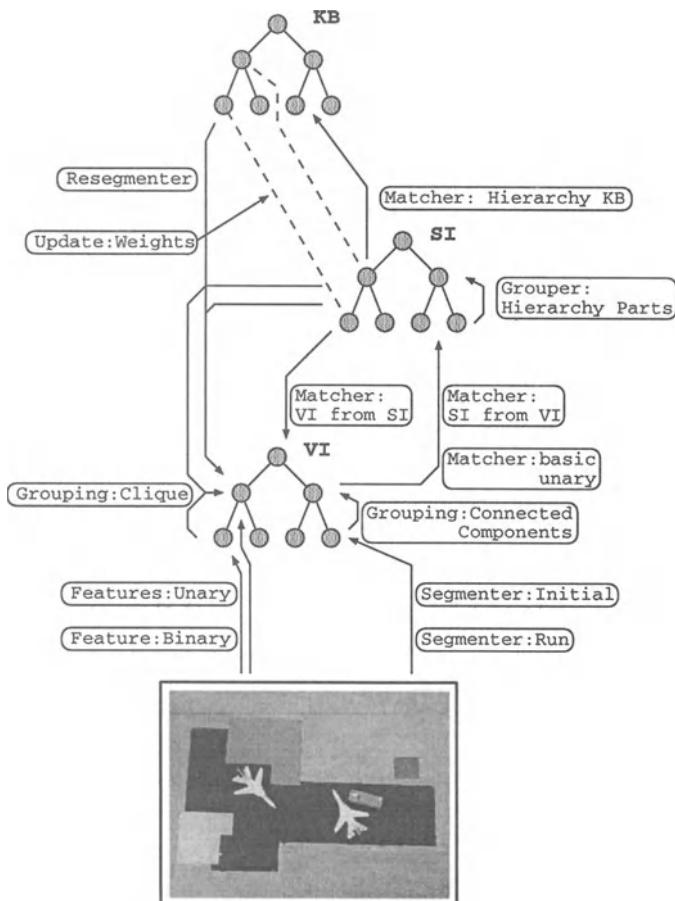


Figure 4.5: Overview of Data Structures and Operators within *Cite*

#### 4.4.1 A Simple Example

For the purposes of illustrating the function and data flow surrounding the main operators in *Cite*, a simple example of recognising a Norfolk Island Pine using the Botanist knowledge base is considered. The sample image and knowledge base (KB) are shown in Figure 4.6.

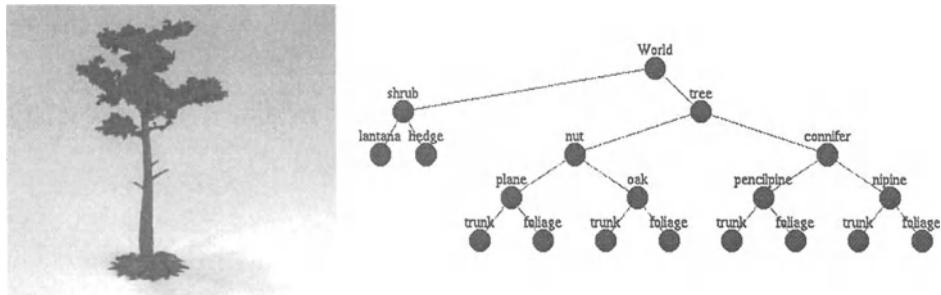


Figure 4.6: Norfolk Island Pine Image and Botanist Knowledge Base

When *Cite* first loads an image a default top level VI node is created. The segmentation initialisation process detects that this VI node has not yet been segmented and begins this process. As no recognition has occurred yet, *Cite* uses the top level KB node to determine the initial segmenter and its parameterisation.

When the segmentation operator has completed, it generates one leaf VI node for each image region. Figure 4.7 shows this initial segmentation and the VI graph at this point. Note that this initial segmentation has resulted in an over-segmentation of the image. The objective is to end up with just two parts, the trunk and the foliage, as described in the knowledge base.

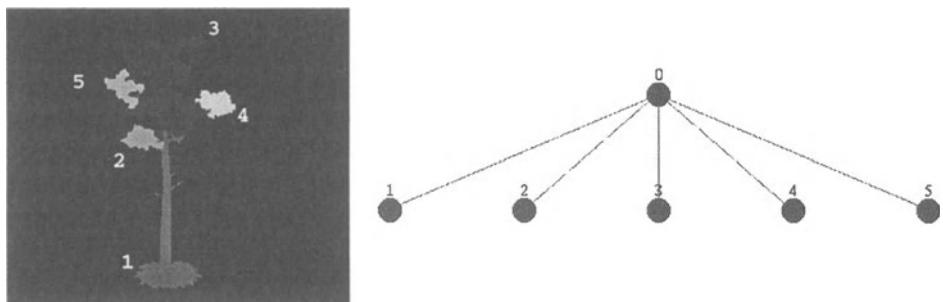


Figure 4.7: Initial Segmentation and Corresponding VI Graph

After the initial segmentation, the unary feature extraction operator detects unattributed VI nodes and begins calculating the unary features

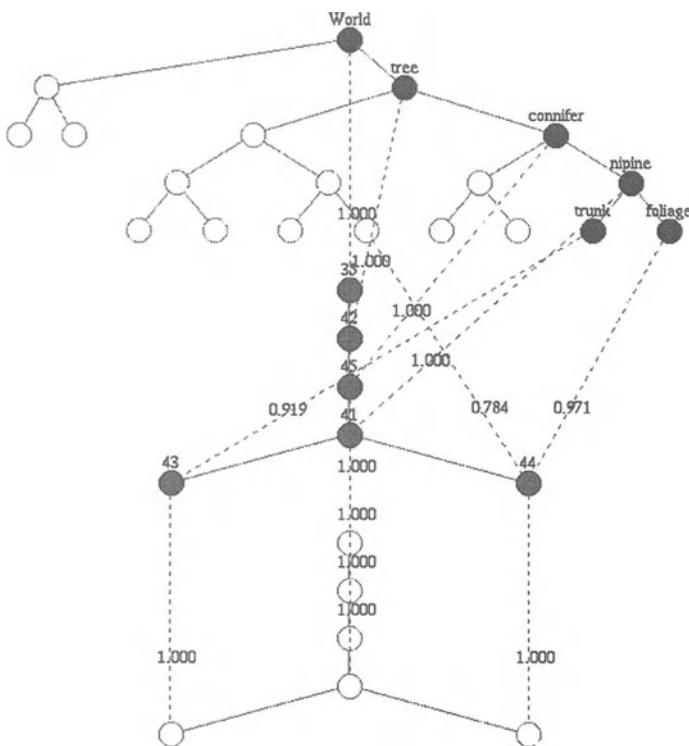


Figure 4.8: Example Data Graphs after the Second Segmentation

for these nodes. At the same time, the connected components grouper constructs a parent VI node representing the collection of VI leaf nodes. The grouped VI nodes are then processed by the SI-from-VI matching operator, and the SI graph is started. The basic unary matching then occurs, matching the VI nodes with the knowledge base and generating SI hypotheses linking the SI nodes to the knowledge base. When this is complete the hierarchical knowledge based matching operator detects missing levels in the SI graph and fills these in. The VI-from-SI matching operator then propagates these down to the VI graph.

In conjunction with the SI and VI node construction operators, the knowledge driven re-segmentation operator scans the SI graph for nodes that may require re-segmentation. This process is discussed in more detail in Section 4.8, suffice to say at this point that although *Cite* believes the object to be an oak tree it clearly has too many children and can be re-segmented using the segmentation parameterisation stored in the “Oak” KB node. This re-segmentation results in the dismantling of most of the VI and SI nodes, leaving just the parent object node which it knows is a

“Tree”.

Figure 4.8 shows the new data graphs after the system has stabilised. The hypothesis for the object is now that it is a “nipse” (Norfolk Island pine), which is correct. Note that one SI leaf node has multiple hypotheses linking it with the knowledge base. This will be resolved trivially by the hierarchical relaxation labelling process, which has not been discussed in this example. The labelled segmentation is shown in Figure 4.9. *Cite* can also produce a text description of the scene, which is listed in Figure 4.10. This example has shown the basic methods by which *Cite* operates. The most important aspects are the three interconnected data structures and the range of operators which create and modify these data structures.

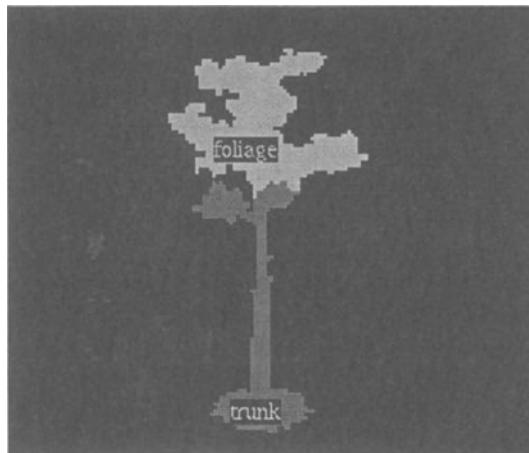


Figure 4.9: Final Labelled Segmentation of the Example Image

```

World[0] (1.000) Consisting of:
└── tree[11] (1.000) Of Type:
    └── conifer[12] (1.000) Of Type:
        └── nipse[6] (1.000) Constructed from:
            └── trunk[9] (0.919)
                └── foliage[10] (0.971)

```

Figure 4.10: Text Description of Example Scene

#### 4.4.2 Operator Scheduling

*Cite*'s operators are all loaded into a process stack when the system is initialised. The process scheduler cycles through this stack and executes each operator in turn. The process scheduler is not interrupt or timer

driven but relies on each operator returning control to the scheduler after a short time interval. There is no inter-process communication other than via the status of the interpretation structures. A commercially oriented implementation of *Cite* in a multi-processing environment could easily fork each operator as a separate process.

The choice of a cyclic scheduler over a control flow scheduler was made so that each operator behaves independently. This provides the system with an “agented” flavour, however this description would be inaccurate as the processes do not duplicate or remove themselves and do not contain any permanent data. Another motivation for this approach is that the data structures in *Cite* represent the current scene interpretation, and the interpretation functions decompose easily into a number of separate and independent processes.

#### 4.4.3 Overview of Operators

##### Segmentation Operators

The segmentation operators are responsible for grouping pixels in the image into regions of common texture or colour, for the grouping of these regions into possible higher level objects, and for the knowledge driven re-segmentation. The segmentation algorithms are discussed in more detail in Section 4.8, but each will be described briefly here.

The `PROCESSSEGMENTINITIAL` operator scans the SI graph searching for nodes with no child visual support. That is, if the visual support node or nodes for the SI node under consideration have no children then *Cite* knows that segmentation is required on the childless VI node. When this condition is detected *Cite* determines the best segmenter algorithm and its parameterisation to be run on the VI node. This is loaded into the *segmentation stack* in the VI node.

The `PROCESSSEGMENTRUN` operator scans the VI graph searching for nodes with incomplete segmentations loaded in their *segmentation stack*. When an incomplete segmentation is detected, this process executes one cycle of the segmentation before returning control to the process scheduler. When the segmentation process on the given VI node is complete, the `PROCESSSEGMENTRUN` operator removes the segmentation from the segmentation stack and places it on the *segmentation history stack* in the VI node. The segmentation history is maintained so that the knowledge driven segmentation process knows what segmentations have been tried in the past on each VI node.

The `PROCESSRESEGMENT` operator is similar to the `PROCESSSEGMENTINITIAL` operator in that it scans the SI graph looking for nodes that require segmentation. However, the re-segmentation operator only operates on SI nodes whose VI nodes have had at least one segmentation already, and where multiple knowledge base hypotheses exist or, in the case of part-of

nodes, the number of children does not match the number in the knowledge base. This operator will initiate a re-segmentation of the VI node in question if labelling subsequent to the earlier segmentations gives rise to a better hypothesis of what the scene element is. The net result of this knowledge driven process is the dismantling of the local VI sub-tree, and the loading of a new segmenter into the segmentation stack of the parent VI node.

The PROCESSSEGMENTCC operator is used to group regions by connected components analysis to form objects. This is a standard bottom-up part grouping algorithm in common usage in machine vision systems. Although fast, it only works on isolated objects (objects that do not touch other objects) and so is only used in early learning. The PROCESSCLIQUE-GROUPING operator is a more sophisticated part grouping algorithm that creates groups of parts that may belong to the same object. This algorithm typically generates a number of different groupings when there are multiple touching or overlapping objects.

### Feature Extraction Operators

The feature extraction process is currently carried out by two processes; PROCESSBASICUNARY and PROCESSBASICBINARY. These calculate the default unary and binary features and store them in the appropriate VI nodes. One important part of these operators is that they must recompute features after re-segmentation takes place, which is achieved by time stamping the features in each VI node.

### Hypothesis Generation Operators

The PROCESSUNARYMATCH operator takes low level VI nodes and tries to match them with the knowledge base. This is achieved by executing the EVALUATEPOINT function of the learning/matching strategy stored in the knowledge base with the unary features of the part. The hypotheses generated link the associated SI node with the knowledge base nodes that best match the unary region properties, using a measure defined separately by each unary matching process.

The PROCESSSIFROMVI operator examines the hierarchical structure of the VI graph to make sure that the hierarchical components are reflected in the SI graph. The SI graph is not necessarily identical to the VI graph but must reflect its topology. This is a bottom-up process. The top-down version of this is the PROCESSVIFROMSI operator which propagates SI hierarchical structure down to the VI graph. The SI structure typically comes from the knowledge base and hence is a knowledge driven top-down process.

The PROCESSHIERARCHYMATCHGROUPS operator takes possible groupings of parts and examines the knowledge base for objects that the parts could belong to. This process combines unary and binary information to generate the initial object hypotheses. The PROCESSHIERARCHYMATCHKB operator examines the knowledge base hierarchy and ensures that the SI

structure reflects the Type-Of and View-Of knowledge. This operator effectively works hand in hand with the PROCESSHIERARCHYMATCHGROUPS operator to provide knowledge driven (top-down) hierarchical scene decomposition.

#### Relaxation Labelling Operators

*Cite* can generate multiple labelling hypotheses, multiple grouping hypotheses and multiple parent relationships for nodes in the SI and VI graphs. Each of these is weighted, and the weights can be interpreted as probabilities. A hierarchical form of relaxation labelling is used to update these probabilities and propagate constraints across the various graph structures.

The relaxation labelling process is a two stage process; the first stage generates new estimates for each hypothesis, and the second stage normalises these hypotheses with respect to the SI and VI graphs. Consequently, all the relaxation labelling algorithms are combined into one process called PROCESSUPDATEWEIGHTS which does both the estimation and then the normalisation on all hypotheses in *Cite*.

#### Ancillary Operators

In addition to the main operators described in the previous sections, *Cite* contains a number of maintenance, status and consistency checking operators. These make *Cite* more usable as well as automatically detecting potential problems in the interpretation structures. This latter facility is valuable because the components of the interpretation structures are constantly being moved, created and dismantled.

## 4.5 Learning World Knowledge

The database in any information processing system can be built in three main ways. It is possible to “hard-code” algorithms optimised for the detection of certain signals based on the experts knowledge of these signals. Secondly, knowledge can be constructed from idealised internal representations such as computer aided design (CAD) object models. Finally, knowledge can be obtained by learning from examples that are sensed in the same manner as in normal run-time operation. *Cite* builds its knowledge base using incremental forms of supervised learning. Learning from examples avoids the requirement in model-based vision of having complete model descriptions of each object.

Recognition of complex objects and scene elements requires some level of solution to what is known as the *parts indexing problem*. To illustrate this problem, consider a face recognition application where two eyes, a nose and a mouth can be segmented out and described by a set of *unary* features covering the colour, size and shape of each part. The system can also compute *binary* features such as the distance and relative orientation between pairs of these parts. A system could conceivably contain a unary

classification mapping and a binary classification mapping, but a problem exists when trying to put all of this evidence together to detect the presence of the “face” object.

Adopting a simple approach of accumulating evidence for the “face” object from the unary and binary components will produce usable results but it will not, for example, be able to distinguish the two images shown in Figure 4.11. This is because the set of unary and binary features for the parts in these two objects is identical.

An unambiguous interpretation requires not only unary and binary features to be matched, but the correct indexing of these parts. For example, a system may know that there exist two parts 5cm apart, but it must know which parts are 5cm apart in order to arrive at a correct interpretation. The worst case ambiguity arises when matching objects with identical parts and one can typically, for  $n$  parts, have  $n!$  objects which could map perfectly to it without full part indexing.

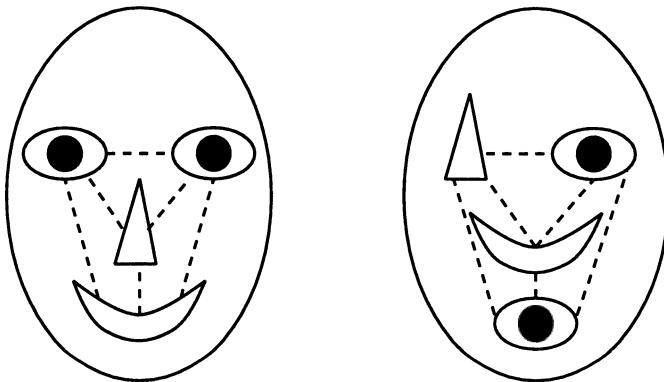


Figure 4.11: Two Objects Indistinguishable Without Correct Indexing Between Unary and Binary Features

A number of solutions to the indexing problem have been proposed. A simple solution is to store a graph representation of the object such that at recognition time the best possible cross-indexing of parts can be determined (usually a costly process). More sophisticated approaches have been proposed to reduce the graph sub-isomorphism complexity (such as the CLARET algorithm in Chapter 3.9) and for recognising objects using a probabilistic mapping on parts (such as CRG and its variants in Chapter 1).

In *Cite*, the binary matching algorithms may be different at each applicable node in the knowledge base and may include exhaustive graph matching and other techniques which exhibit differing degrees of solution of the part indexing problem. Relaxation labelling is always used to resolve wider scope label incompatibilities represented by the hypotheses in the

scene interpretation. This can influence the way the binary matching results are propagated through the SI graph, but does not play a role in how the binary matching results are generated. This is solely done by the binary matching algorithms stored at each Part-Of parent node in the knowledge base. These binary matching algorithms are discussed in detail later in this section.

#### 4.5.1 Overview of the Learning Strategy in *Cite*

The hierarchical learning in *Cite* is made relatively easy by the fact that the hierarchy is defined contextually according to the manner in which the system is used. That is, the descriptions of the objects that the operator gives *Cite* during an incremental learning step contain sufficient information to build the hierarchy in a reasonably straight forward manner. Hierarchical learning in *Cite* is of less interest than the incremental learning which occurs at each node within the knowledge base, and of the way the hierarchy is used during classification.

##### Learning Options

If *Cite* correctly classifies a new instance of an object already in the knowledge base, then, in general, it can be claimed that the knowledge base is sufficiently general to recognise the new object instance. In some cases, such as when the learning algorithm being used is based on a set of probabilistic rules, it would be appropriate to run the **LearnTrue** function on this correctly labelled object to reinforce its rules. In most other learning methods there is nothing to be gained from learning already correctly labelled objects.

When *Cite* incorrectly labels an object that is already in the knowledge base there are three possible reasons for this, each of which requires a different action:

- The incorrectly labelled object may be a completely new view of the object, and hence a new view must be added to the knowledge base.
- The matching algorithm may not be utilising the correct or available features (whether unary or binary) so these may need to be added to or deleted from the existing knowledge base.
- The object may be a different example of an existing view of an object, but the matching algorithm may have incorrect rules or parameters that require incremental retraining.

Of these three options, incremental rule modification is the least computationally and memory expensive, while adding a new view is the most expensive in these two regards. In the current implementation of *Cite*, the features calculated are fixed and all are considered during learning although

they may not all be considered during recognition. Deciding between learning a new view and learning a new example of a current view is determined by the user when learning is required.

### 4.5.2 Unary Learning Algorithms

In each node of the knowledge base, a unary matching strategy and all its required data structures can be stored. The unary matching strategy may be different at each node, and there need not be one at each node, although there does need to be at least one at every leaf node. Each matching strategy must be able to perform three operations:

1. to determine the degree of match or mismatch to a candidate part,
2. to learn a new positive example,
3. and to learn a new negative example.

In addition it is desirable that the learning strategy adhere to the two incremental learning algorithm requirements of bounded memory usage and bounded computational complexity when incrementally learning.

*Cite* currently contains two unary matching strategies; a bounded rule based method and a prototype nearest-neighbour based method. In their current form, neither of these algorithms conform to the bounded memory and complexity requirements of ideal incremental learning algorithms. However, this chapter does contain a significant new method for converting non-incremental to incremental algorithms which does conform to the bounded memory and complexity requirements. This method, known as the explanatory least generalisation (ELG) method is described in detail in Section 4.5.2 where it is applied to the ID-3 minimum entropy decision tree algorithm [270]. The ELG method could be applied to the two unary learning strategies used in *Cite*, but with the modest size of the knowledge base there would be no practical performance improvement.

#### Learning Unary Bounded Rules

A common rule representation in machine learning and pattern recognition is that of the conjunction of bounded attribute rules. For example, one may say that if an object's redness is between 0.8 and 1.0, its circularity is between 0.7 and 0.9 and its size is between 0.15 and 0.2, then it is an apple. Such rules effectively form hyper-rectangular volumes in feature space, with each providing evidence for the object that is represented by the rule. Objects may have multiple overlapping rules, and some approaches have rules representing evidence for the object and other rules representing evidence against it. Generalisation is achieved by having rule conjunctions covering a volume in feature space greater than the set of individual training examples.

The UnaryBounded rules are incrementally built from example positive and negative points. Bounded rules are evaluated as a conjunction of feature upper and lower bounds. That is, test point  $\mathbf{u}$  activates rule  $i$  if, for all features  $j \in \{1\dots n\}$ ,  $r_{i,j}^{\text{low}} \leq u_j \leq r_{i,j}^{\text{up}}$  where  $r_{i,j}^{\text{low}}$  is the lower bound of rule  $i$  in feature  $j$  and  $r_{i,j}^{\text{up}}$  is the upper bound of rule  $i$  in feature  $j$ .

In the UnaryBounded learning algorithm, rules are constructed as pictorially described in Figure 4.12. To learn a new positive example, both positive and negative rules are considered. Each positive rule is extended to include the new point (1), and then any of these extended rules which overlap negative rules are not considered (2). If there are no extended rules remaining, a new small volume rule is constructed centred around the new point (3). If there are multiple extended rules which do not overlap any negative rules, the rule of smallest volume increase (4) is kept and placed in the final positive rule set (5). In the case of learning a new negative example, the same algorithm is applied but negative rules are extended and not permitted to overlap existing positive rules. The net result is that positive rules may overlap each other, as may negative rules, but positive and negative may not overlap each other.

### Learning Unary Prototype Rules

An alternative to the UNARYBOUNDED algorithm is to construct prototypes of each object in the feature space, and then classify according to the test point's distance from each prototype. Depending on the distance metric used, the classification boundaries are a form of tessellation.

In *Cite* this matching strategy is called the UNARYCLOSEST algorithm. This is a simple system, and the prototype points may be built by a number of methods. In a non-incremental form the prototypes can be built using clustering techniques. The incremental form proposed uses a straight forward time series averaging algorithm, although many other approaches could be used.

The incremental learning algorithm for the UNARYCLOSEST matching strategy moves the prototype points towards the training samples when correctly matched, or creates new prototype points when not correctly matched. The Euclidean metric is used to determine proximity to the prototype points, which end up following a trajectory towards the mean position of the local training samples.

### Learning by Explanatory Least Generalisation

Categorical machine learning is concerned with the classification of object instances by concepts learned from examples of these objects. The objective is to form concepts (rules) that operate on object attributes such that the system can be used to classify further unknown objects. Incremental learning is concerned with learning these rules from a serial presentation of object instances, as opposed to having a large *training set* of instances available for learning.

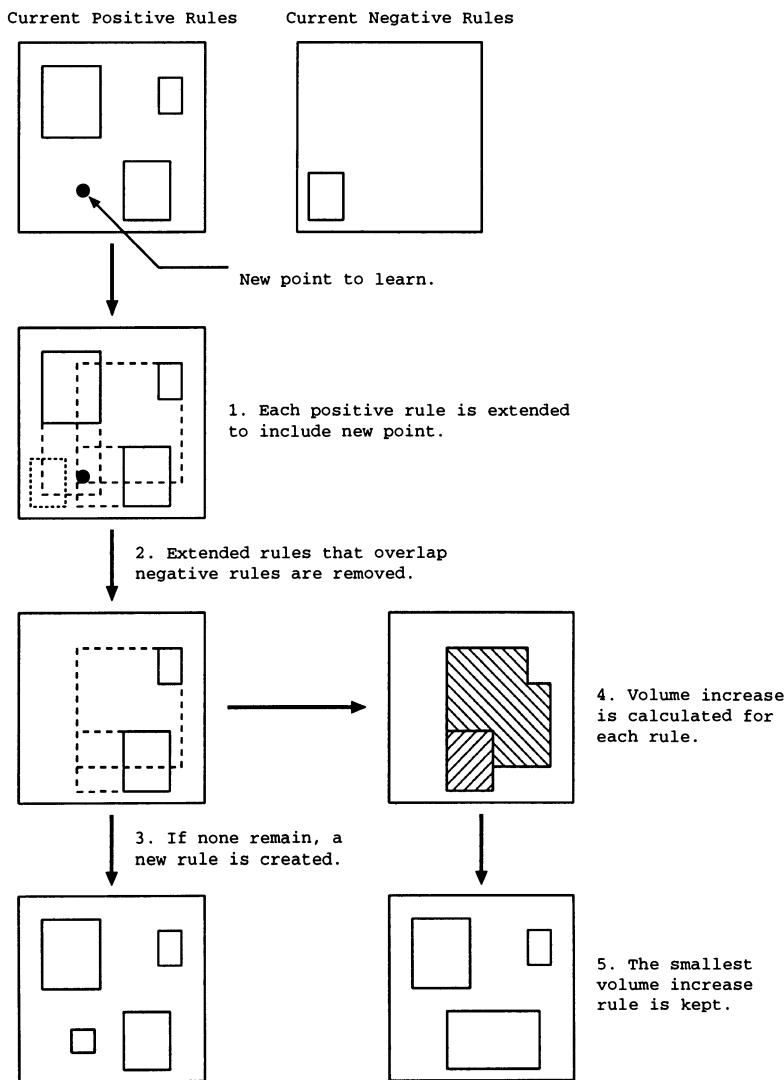


Figure 4.12: UnaryBounded Learning Algorithm Flow Chart

As discussed earlier, incremental learning algorithms should be sub-linear in memory and complexity with respect to the number of training examples, and should converge to at least an equivalent solution as the non-incremental form of the algorithm given that the presentation order to the incremental algorithm is first-order stationary. Published incremental learning theories in machine learning, such as Schlimmer and Fisher's ID-4 [303] and Utgoff's ID-5R [342], do not completely satisfy the above

requirements. ID-4 keeps insufficient information to converge to the same solution provided by the non-incremental variant, ID-3 [270]. The ID-5R algorithm essentially keeps the full set of instances distributed across the concept hierarchy.

The following section is concerned with a general approach to the development of incremental supervised learning methods that conform to the conditions described above, and a specific example is demonstrated in the development of a memory-bounded and convergent incremental learning system for decision trees on real-valued attributes. This general approach is achieved by storing a least generalisation of the data which gives a minimum explanation of the current concept hierarchy. This is termed the explanatory least generalisation (ELG) incremental learning method.

At each iteration, an incremental learning system is provided with a new instance of data that may or may not be classified correctly according to the current learned concepts. If the new data is not classified correctly then the current concepts must be modified to accommodate the new data. Local concept restructuring will not always result in the optimal solution[342], so global restructuring is necessary. However, global restructuring on available instances is not only computationally intensive, but also requires storage of these instances, which does not satisfy the incremental learning requirement of bounded memory.

The solution lies in a general approach of storing both the concept hierarchy and a *sufficient least generalisation of the instances observed to explain the current concept hierarchy*. This least generalisation is termed an *explanatory least generalisation* (ELG) of the instances with respect to the particular concept hierarchy used. The ELG can be constructed by maintaining a least generalisation of instances that map into each rule. At each iteration, the ELG is used in restructuring the set of classification rules (the knowledge base), and then these rules are used to split and merge the ELG. The result is that at the end of each iteration the ELG is valid for the current set of categorical rules. In order to formalise this theory, the following sections demonstrate the ELG approach applied to binary decision trees.

### **The Incremental Decision Tree Algorithm**

Pattern recognition typically involves real-valued attributes which are often visualised as a Cartesian space. Object instances are represented as points in this feature space, one-sided rules are represented as partitions, and conjunctive rules are represented as hyper-rectangles. Both partitions and hyper-rectangles are generalisations in the feature space of dimensionality equal to the number of features.

Formally the attributes for each instance are denoted as  $f_{i,j}$  for attribute  $j$  of instance  $i$ . The binary decision tree is a hierarchy of feature partitions constructed from single feature splits, as shown in Figure 4.13. Each node in the decision tree is either a categorisation (a leaf node) or a decision

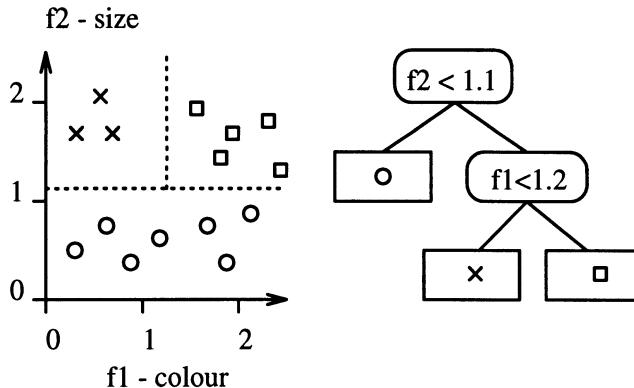


Figure 4.13: Example of a Feature Space and the Computed Binary Decision Tree

node. At each decision node, one and only one feature value is one-sided tested, with the left branch being true, and the right-branch being false. This gives rise to a tiled hyper-rectangular feature space partitioning.

Non-incremental decision tree algorithms examine a large set of training data in order to arrive at the optimal set of feature partitions. The ELG method constructs and maintains the ELG alongside the decision tree and uses this, rather than the training data, to construct the decision tree.

The ELG that is maintained at each leaf node in the decision tree is a feature space hyper-rectangle that contains, in addition to the feature bounds of instances generalised by the hyper-rectangle, the time the hyper-rectangle was created and the number of instances generalised by the hyper-rectangle. This latter information is used in merging, splitting and removing hyper-rectangles.

At each application of a new, correctly labelled feature instance, the algorithm creates a single-point hyper-rectangle of no volume. This hyper-rectangle, along with the previous explanatory least generalisation hyper-rectangles are then processed to form a binary decision tree (Figure 4.14). Splitting is done to minimise the partition entropy, which is a common metric used in real-valued decision trees. Partition entropy is calculated as  $E = P_0 + P_1$  where  $P_0$  is the entropy of the ELG partitions below the nominated decision boundary and  $P_1$  is the entropy above the nominated decision boundary. Entropy is calculated as  $P = -\sum_i^N p_i \ln(p_i)$  where  $N$  is the number of different categories, and  $p_i$  is the fraction of instances of class  $i$ . The number of instances that fall into each ELG partition needs to be kept in order to calculate the partition entropies. Apart from this information, however, the exact feature vector  $\tilde{f}_i$  is ignored after this point. The important distinction between this and previous methods is that the data atoms are the ELGs, which are hyper-rectangles, as opposed to the

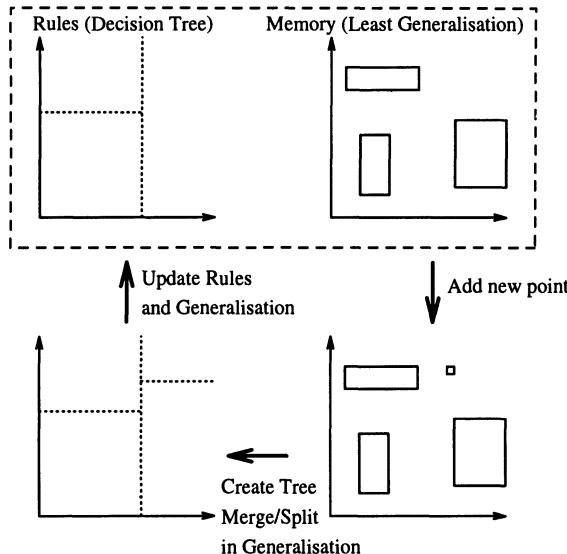


Figure 4.14: Incremental Learning Rule and Least Generalisation Model

full training set, which is a large set of feature points. This results in the possibility of splitting an ELG hyper-rectangle, in which case point instances are distributed uniformly amongst the split regions.

The new set of hyper-rectangles is then merged according to the feature space partitioning dictated by the decision tree thus created. This resultant set of hyper-rectangles is the correct ELG for the next iteration, and the new decision tree can be used for correct classification at that point in time.

The method for generating decision trees is similar to ID-3 and other decision tree systems. However, the generation of potential feature splits is far more complex because the “raw data” is a set of non-zero extents on each feature axis (pairs of upper and lower bounds from each ELG partition). Similarly, once the feature and the position of the best split is determined, it is necessary to split the ELG partitions. This can become complex as a number of special cases arise if the new data point happens to land in a current ELG partition.

Typically, complete restructuring is rare and only occurs when a new instance sets up a long chain of concept violations. The complexity of this reconstruction is bounded approximately by the number of leaf nodes in the decision tree. This compares favourably with the heavily computational task of reconstructing the decision tree on a large number of individual training instances (as is the case in non-incremental learning). As the complexity of the ELG matches that of the decision tree and dictates the amount of data used in each reconstruction process, the amount of mem-

ory and computer time used is directly related to the complexity of the classification problem, not the amount of data used.

### 4.5.3 Binary Learning Algorithms

Binary matching in *Cite* occurs when the operator HIERARCHYMATCHKB detects an unmatched parent with multiple children each of which has at least one initial label hypothesis. Binary matching can also occur as a result of the clique resolving algorithm generating possible groupings of parts. The binary matching algorithm is run similarly to the unary matching algorithm except that the test point is a binary feature table rather than a single unary feature point.

*Cite* contains two binary matching algorithms, BINARYBIPARTITE and BINARYCLOSEST. The choice of binary matching algorithm used can be different for each node in the knowledge base.

#### Learning Binary Closest Matching

The BINARYCLOSEST matching algorithm is a simple high-speed binary matching algorithm that relies on the correct ordering of the unary parts from the unary matching process. The degree of match is computed as the Euclidean proximity to the closest prototype point, which is very similar to the UNARYCLOSEST algorithm.

If we let  $\tilde{f}_{i,j}$  be the binary feature vector describing the relationship between parts  $i$  and  $j$ , and  $\tilde{M}_{i,j}^k$  be the binary feature vector describing the relationship between model parts  $i$  and  $j$  in prototype  $k$ , then the degree of match is defined as the following:

$$\text{match}_{\text{bc}} = 1 - \frac{1}{N} \left( \min_{k \in 1..N} \sum_{i,j} \|\tilde{f}_{i,j} - \tilde{M}_{i,j}^k\| \right) \quad (4.1)$$

Updating of the prototype models  $M$  is done in the same manner as in the UNARYCLOSEST algorithm. When learning a new positive example, the closest prototype is time-series updated if it is within a pre-set threshold distance of the closest prototype point, otherwise a new prototype point is added to the model set.

The BINARYCLOSEST matching algorithm does not fully address the part indexing problem. For many objects the parts are correctly indexed by combining the unary matching outcome with relaxation labelling, then the binary matching determines which is the best “group” description of the parts. For this, the BINARYCLOSEST algorithm works perfectly well, especially in conjunction with the clique resolving process. However, for objects that have multiple similar parts with different spatial relations to other parts in the object, the part indexing problem needs to be taken more

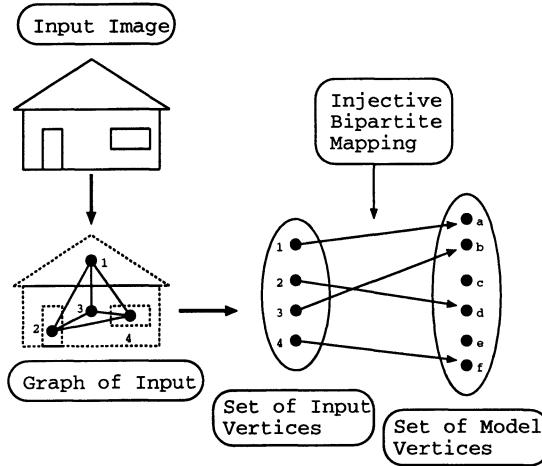


Figure 4.15: Direct Implementation of Bipartite Matching

seriously. To cover this range of objects the `BINARYBiPARTITE` algorithm, described in the next section, is also included in *Cite*.

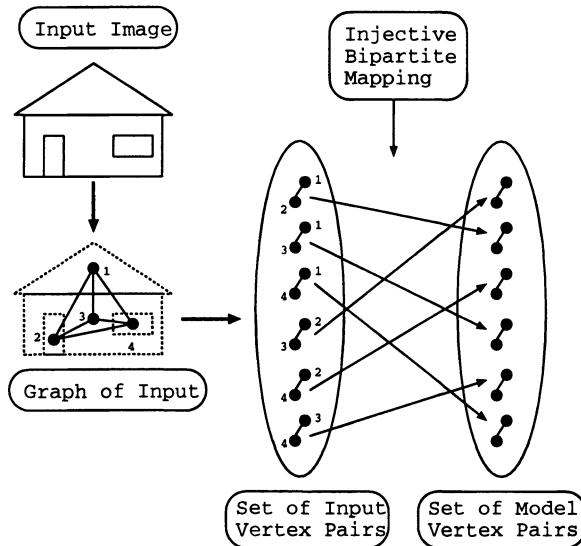
### Learning Binary Bipartite Matching

Complete solutions to the graph isomorphism problem are generally computationally exhaustive<sup>1</sup> or memory intensive with large pre-compilation complexity [207]. Algorithms providing different degrees of solution exist, ranging from the naive complete solution (which is  $O(n!)$  in the number of parts) to simple methods that effectively ignore the part indexing problem and accumulate evidence based on occurrence of features (such as in EBS [50] or `BINARYCLOSEST` above). For different applications, different degrees of solution are required to provide satisfactory answers and there is usually a trade-off between speed and uniqueness.

Bipartite matching is an intermediate solution to the graph matching problem which, when implemented directly, incorporates unary information but not binary or *relational* information other than the injective mapping between the data and model graphs. Figure 4.15 illustrates this direct implementation of the bipartite matching scheme. Such a scheme has been used in object recognition, usually in conjunction with other methods to incorporate relational information[160]. Bipartite matching has been shown to be polynomial in the number of parts [141].

Bipartite matching can be easily extended to include some relational information by matching pairs of data nodes to pairs of model nodes. This is achieved by creating the  $n(n - 1)$  pairs of data nodes and  $m(m - 1)$  pairs of model nodes and then applying bipartite matching to these. Figure 4.16

<sup>1</sup>The best algorithm for the worst case scenario is known to be  $O(2^{\frac{n}{3}})$  in the number of parts [332]



**Figure 4.16: Bipartite Matching Extended to Include Partial Relational Information**

illustrates this process using the same input graph as in Figure 4.15. Note that in extended bipartite matching each matching node now represents two unary nodes (vertices) and the binary relations between them (the edge).

Graph bipartite matching of this form using partial relational information does not guarantee a unique solution and in some cases can provide “non-physical” solutions. This arises from contradictory pair-wise matchings which can only be resolved using the full part indices. However, if these are used, the algorithm would be fully solving the graph matching problem, which is computationally exponential.

The *measure* of match for bipartite graph matching in *Cite* sums the total error based on a Euclidean distance metric and the best match is declared as that which has the smallest error. A discrete form of this is presented in [160] in which matches occur based on proximity subject to a global noise threshold. The number of nodes to be matched in extended bipartite matching is  $n(n-1)$ , however this still yields a polynomial solution for the most efficient algorithm.

#### 4.5.4 Interaction of Unary and Binary Matching with Relaxation Labelling

*Cite*'s recognition process operates at a number of levels. At the “local” level, recognition is distributed between four operator classes which cover unary and binary matching, group hypothesis generation and relaxation

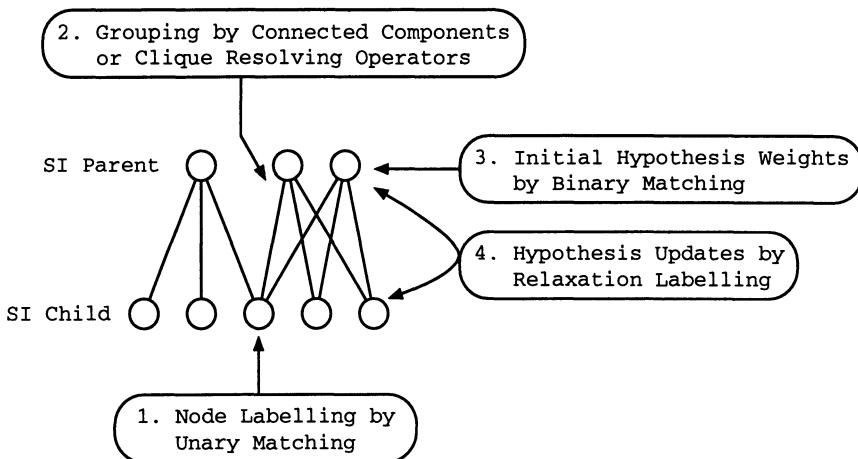


Figure 4.17: Interaction of Unary and Binary Matching, Relaxation Labelling and Hypothesis Generation

labelling. At a wider scale, other operators such as the knowledge driven re-segmentation and hierarchical matching also contribute importantly to the recognition outcome.

Figure 4.17 illustrates the local recognition processes on a small section of a SI graph. Before the parent SI nodes exist, the child nodes are generated indirectly from the segmentation process. These are then matched to the knowledge base using the unary matching strategy (1) present at each applicable KB node. From this, the connected components or clique resolving operator groups the child nodes into multiple possible parent groupings (2). This is an indirect process because grouping actually occurs in the hierarchical segmentation (VI graph) and is propagated to the SI by another operator. When these possible groupings have been established, the binary matching operator determines the match to each parent using the matching strategy at each applicable node in the knowledge base (3). These degrees of matching combined with unary matching and support from related sections of the SI graph then serve to initialise the hierarchical relaxation labelling process (4).

Although the initial order of these operations for a single object will be as described, for multiple objects the process becomes more temporally complicated. Certain objects will be classified faster than others, and the re-segmentation process will restart the entire local recognition process for different sub-parts of the SI graph at different times. This on-going change means that the relaxation labelling process must be able to cope with the creation and removal of unary and binary matching strengths at each point in the SI graph at different times.

## 4.6 Hypothesis Generation

*Cite* generates classification labels and grouping hypotheses in the visual interpretation and scene interpretation structures based on segmentation results, unary and binary feature matching and examination of knowledge base data structures. Hypotheses exist at many different levels within *Cite* and are resolved by a process of hierarchical relaxation labelling and constraint propagation. By extending traditional relaxation labelling to operate over a hierarchy, top-down and bottom-up recognition processes are seamlessly integrated. Through interaction with other operators the results of the relaxation labelling can then result in re-segmentation of image regions and the prediction of missing parts or objects.

This approach departs from the processes used in most object recognition systems where there is little or no feedback between the various stages during recognition. One innovation in *Cite* is that of considering all components of the data structure, including the hierarchical segmentation structures, as hypotheses. This view enables an elegant and general application of relaxation labelling as the constraint propagation process.

### 4.6.1 Hypothesis Types and Storage

Every aspect of the VI and SI graphs is considered as a *hypothesis* within *Cite*. The existence of a VI node or a SI node is a *unary hypothesis*, and the links between these nodes and through to the knowledge base are called *binary hypotheses*<sup>2</sup>. Each hypothesis contains a weight which represents the probability of that hypothesis being true.

In the visual interpretation graph, a VI node represents the probability that a certain group of pixels can be described as a single object or scene element. The VI-VI links represent the child VI node as being a valid partial decomposition of the parent VI node. In the SI graph, a SI node represents the probability that an instance of an object exists in the scene. The SI-SI links represent the child SI node as being a valid component, view or type of the parent SI node. The VI-SI links represent the probability of the VI node providing image support for the SI node, and the SI-KB links represent the probability of labelling the SI node as an instance of the KB node in question.

*Cite*'s data structures essentially form a decomposition web and analysis at multiple levels of abstraction which range from image pixels through to knowledge base nodes. This unifying concept allows common treatment of hypotheses and permits powerful relaxation operations because all aspects of the data structures are weighted and can thus be modified in a similar manner.

---

<sup>2</sup>Note that unary and binary hypotheses are unrelated to unary and binary *features* or *matching*.

### Hypothesis Notation

Table 4.1 describes each of the hypothesis types and where they are found in the graph structures. For completeness,  $P_i^K$  and  $P_{i,j}^K$  could be included and defined as the probability of KB node  $i$  existing and the probability of KB node  $i$  being a child of KB node  $j$  respectively. However, these values would always be 1.0, and the probability normalisation would not be applicable because the KB graph can remain as a graph with each node having multiple parents.

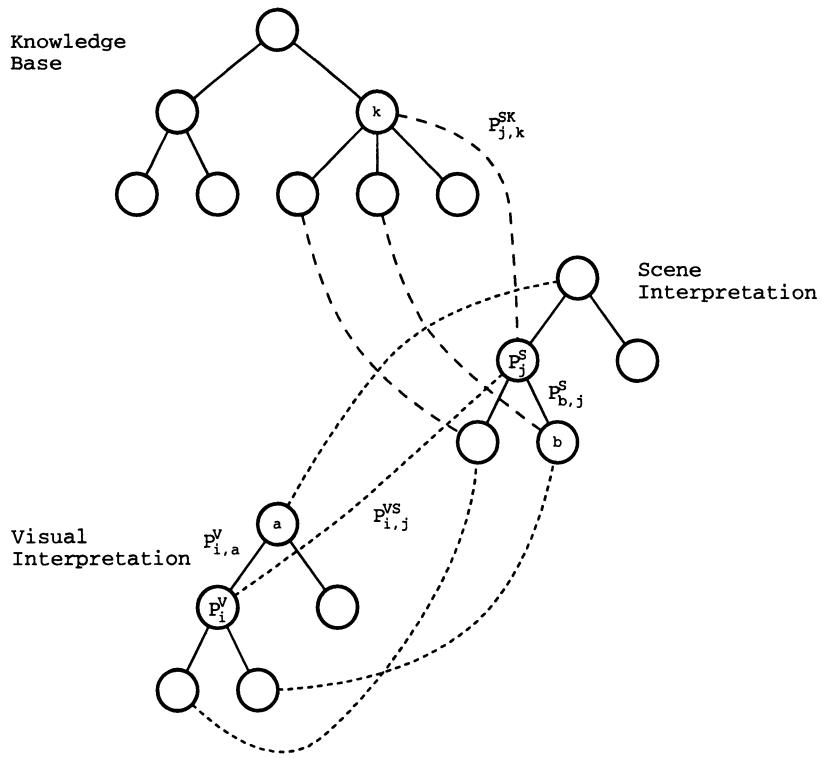
Figure 4.18 illustrates the different hypothesis types and where they appear in the graph structures. Starting with the VI graph, which represents the hierarchical segmentation, we have  $P_i^V$  being the probability that VI node  $i$  exists, and  $P_{i,a}^V$  being the probability that VI node  $i$  is a child of VI node  $a$ . Linking this with the SI graph,  $P_{i,j}^{VS}$  is the probability that SI node  $j$  is represented by VI node  $i$  in the image. Inside the SI graph,  $P_j^S$  is the probability that SI node  $j$  exists, and  $P_{b,j}^S$  is the probability that SI node  $j$  is the parent of SI node  $b$ . Making the final connection between the SI and the KB,  $P_{j,k}^{SK}$  is the probability that SI node  $j$  is an instance of object  $k$ . Note that this figure is for notation purposes only, and typical full graphs contain multiple parent connections and multiple labelling hypotheses for each node.

In order to simplify the hypothesis update expressions, it is necessary to define a notation for the child and parent lists in each of the three graph structures. Table 4.2 contains a description of these index sets.

### Hypothesis Storage

Name	Location	Description
$P_i^V$	VI Node	Probability that VI node $i$ exists. Interpret as the probability that VI node $i$ is a correct segmentation.
$P_{i,j}^V$	VI-VI Edge	Probability that VI node $i$ is a child of VI node $j$ , where both are in the same image.
$P_{i,j}^{VS}$	VI-SI Edge	Probability that VI node $i$ is the segment of the image corresponding to SI node $j$ .
$P_i^S$	SI Node	Probability that SI node $i$ exists.
$P_{i,j}^S$	SI-SI Edge	Probability that SI node $i$ is a child of SI node $j$ .
$P_{i,j}^{SK}$	SI-KB Edge	Probability that SI node $i$ matches KB node $j$ .

Table 4.1: Notation of Hypothesis Types

Figure 4.18: Hypothesis Notation in *Cite*

Unary hypotheses are stored simply by their creation and addition to existing data structures. Binary hypotheses are stored alongside the reference to the link destination node. The VI-VI, SI-SI and VI-SI binary hypotheses

Name	Location	Description
$V_i^C$	VI	Indices of children of VI node $i$ .
$V_i^P$	VI	Indices of parents of VI node $i$ .
$V_i^S$	VI	Indices of SI nodes with hypothesis links from VI node $i$ .
$S_i^C$	SI	Indices of children of SI node $i$ .
$S_i^P$	SI	Indices of parents of SI node $i$ .
$S_i^K$	SI	Indices of KB nodes with hypothesis links from SI node $i$ .
$K_i^C$	KB	Indices of children of KB node $i$ .
$K_i^P$	KB	Indices of parents of KB node $i$ .

Table 4.2: Notation of Node Index Sets

are stored both at the source and destination nodes. The weights on these hypotheses are stored only at one node to avoid duplication. The weights themselves are stored as a floating-point number in the range [0.0, 1.0]. In addition, there is a *temporary* weight storage used by the first phase of the relaxation labelling process, and a weight history stack which records the epoch (iteration number) and value of the weight whenever it is changed. The weight history is only used to graph the weight for analysis purposes.

#### 4.6.2 Hypothesis Generation Procedures

There are two basic types of hypothesis generation procedure. The generation of a SI or a VI node is achieved by *unary* hypothesis procedures, and the generation of VI-VI, VI-SI, SI-SI and SI-KB hypotheses is performed by *binary* hypothesis procedures. There will be some situations where multiple binary hypotheses can be generated as a result of a single unary hypothesis. For example, in clique resolving by connected components a parent VI node is generated with binary hypotheses to the image regions that are its children. This can only be sensibly achieved in a single procedure, which can generate both unary and binary hypotheses.

#### 4.6.3 Generating VI Nodes from Image Data

The process of image segmentation is usually a low-level bottom-up procedure which is executed as the first stage in most object recognition and scene understanding systems. Segmentation is invariably fragile and requires significant parameter adjustment to provide even vaguely useful results. *Cite* improves on the standard bottom-up segmentation process by also including top-down knowledge driven segmentation. That is, multiple re-segmentations of regions within the image can occur as *Cite* builds up evidence for the current interpretation.

The knowledge driven segmentation process and specific details of the segmentation algorithms are described in detail in Section 4.8. At this point, it suffices to describe the segmentation processes as producing VI leaf nodes, each of which represents one region in the image when viewed at the lowest level of detail. VI nodes themselves are generated by the PROCESSSEGMENTRUN operator, which executes the segmentation processes loaded into the VI structures by the PROCESSSEGMENTINITIAL operator and the PROCESSRESEGMENT operator.

VI nodes generated by the first segmentation will always end up as leaf nodes added under the top level VI node. Nodes generated by subsequent re-segmentations will be attached as leaf nodes at the appropriate point in the hierarchy, which is typically under the parent VI node from which the re-segmentation occurred. There will be occasions when nodes generated from re-segmentations will be placed higher in the hierarchy, but this is

described later in Section 4.8.1.

#### 4.6.4 Generating VI Nodes from The VI Graph

Intermediate parent VI nodes can be generated directly from the image data and VI graphs if the connected components operator, PROCESSGROUPINGCC, is activated. Connected components grouping is only used during early training when the system is shown single objects and has not yet learned enough to reliably run the clique resolving process. The decision to switch from the connected components grouper to the clique resolving grouper is controlled from the user interface. The algorithm for the connected components grouping is straight forward and moderately efficient<sup>3</sup>.

Forming objects by simple connected components analysis is common in object recognition systems. It fails when there are multiple overlapping objects, or objects comprising non-connected parts. Connected components grouping is used in *Cite* only as a boot-strap algorithm that is deactivated after the first few objects have been learnt.

#### 4.6.5 Generating VI Nodes from the SI Graph

There are two processes which build VI nodes by examining the SI graph structure. The first is the re-segmentation process which will take a VI node and its leaf node children and apply a new segmentation process to this based on the SI node knowledge base hypothesis. The second is the top-down process of ensuring that higher level labelling structures (such as taxonomic expansions) are reflected in the VI graph. Both of these processes are top-down, one representing knowledge driven re-segmentation, the other knowledge driven image analysis.

The PROCESSVIFROMSI operator is responsible for propagating hierarchical scene information down to the VI graph. This algorithm executes recursively on the VI graph, and scans the VI graph looking for nodes and their children whose SI nodes have an extra layer or layers of nodes between them. If found, this implies that there is hierarchical structure in the SI which needs to be propagated down to the VI graph. Once such a node has been found, an appropriate SI node from the middle SI layers is located. The final stage is that an intermediate VI node is inserted and linked to the intermediate SI node.

#### 4.6.6 Generating SI Nodes from the VI Graph

When grouping occurs in the VI graph this structure must also be translated into the SI graph. Region grouping will occur in the VI graph as a result of connected components grouping, which is only used during early training,

---

<sup>3</sup>A full discussion of linear time connected components labelling is given in [85]

and the more sophisticated region grouping by clique resolving which is used a majority of the time.

The PROCESSSIFROMVI operator is run recursively on the VI graph and operates on VI nodes with no SI hypothesis but with children that each have at least one SI hypothesis, as shown in Figure 4.20. Once such a node is found, the MOSTCOMMONPARENT algorithm is used to determine the attachment point in the SI structure, and an intermediate SI node is created. This new SI node has as its children the SI nodes of the VI node's children.

#### 4.6.7 Generating SI Nodes from the KB Graph

The PROCESSSIFROMKB operator examines the knowledge base looking for missing nodes to insert into the SI graph. This is a top-down process which expands the SI graph to include taxonomy and view information relating to already hypothesised objects.

Figure 4.19 illustrates the three main stages of the algorithm, which is not greatly different from the PROCESSVIFROMSI algorithm. The first stage determines nodes in the SI which require insertion of their descendants based on the structure of the knowledge base. There must exist missing structure, and the best hypothesis for the SI node in question must be sufficiently greater than the others. In the second stage of the algorithm, the most appropriate KB to insert is determined by looking for commonality of ancestry of the best KB node's children. Once this new KB has been determined, a new SI node is added in stage three with one hypothesis pointing to this KB node.

#### 4.6.8 Binary Hypothesis Generation by Basic Unary Matching

The basic unary matching operator generates SI-KB hypotheses for leaf VI nodes by matching them to KB nodes. This is achieved by running the EVALUATEPOINT method in the unary matching strategy stored in the KB node being matched to.

Matches are made based on a ranking and threshold system, and candidates are taken from a pruned set of KB nodes based on higher level matching. Although this algorithm runs on the VI graph, it creates SI-KB hypotheses which are stored in the scene interpretation graph.

#### 4.6.9 Resolving Clique Membership

One important factor in categorising object recognition and scene understanding systems is their ability to isolate and identify multiple touching or overlapping objects. The process of grouping parts or image features into

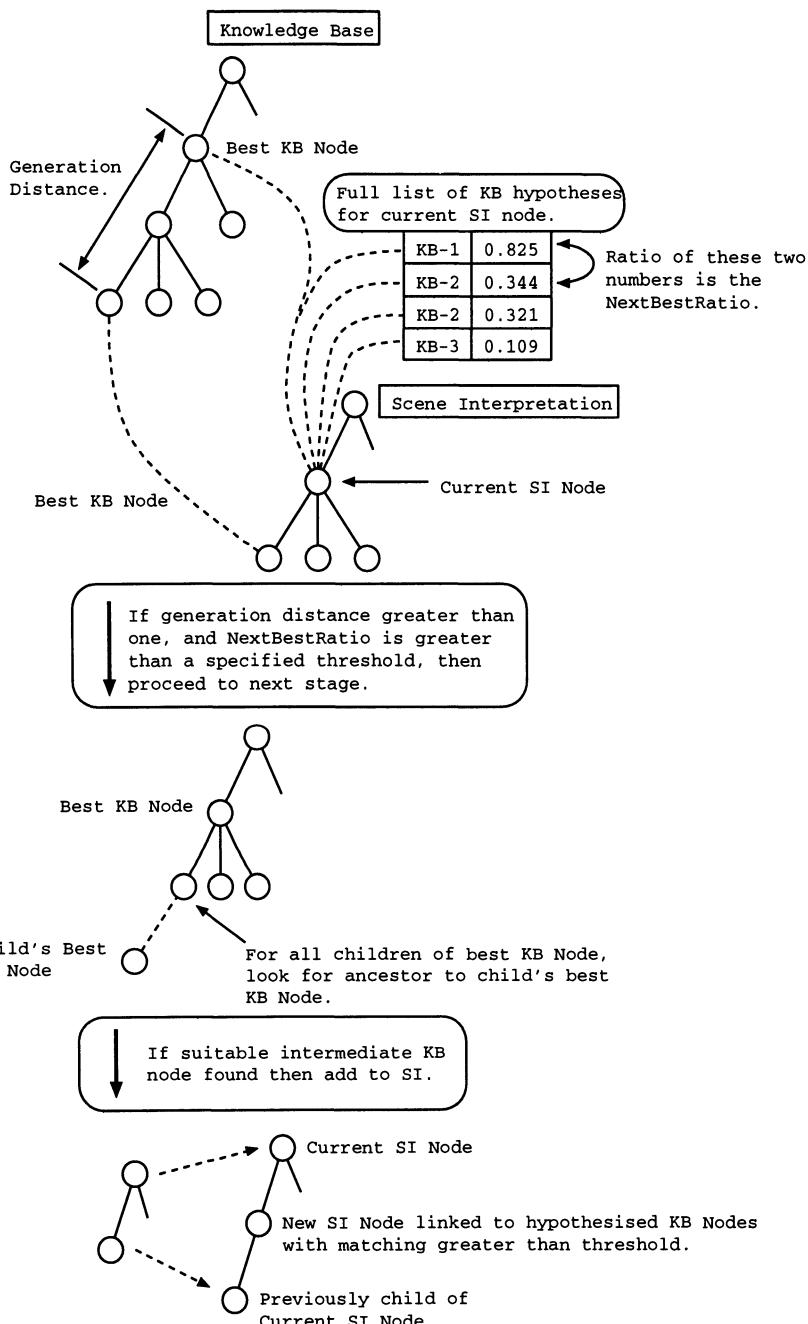


Figure 4.19: Description of ProcessSIFromKB Algorithm

objects or scene elements is called the *clique membership* or *parts clique* problem. This problem becomes extreme when parts-based recognition is used in industrial inspection style bin-of-parts problems.

Connectivity is a common constraint used to reduce the complexity of the clique membership problem in conventional object recognition systems. In this situation, the connectivity constraint simply states that any object or scene element in the image will be constructed from connected regions. This constraint is useful for physically constructed objects, but is too strong for higher level scene elements, such as those found in the airport scenes. Note that object recognition systems which are restricted to single isolated

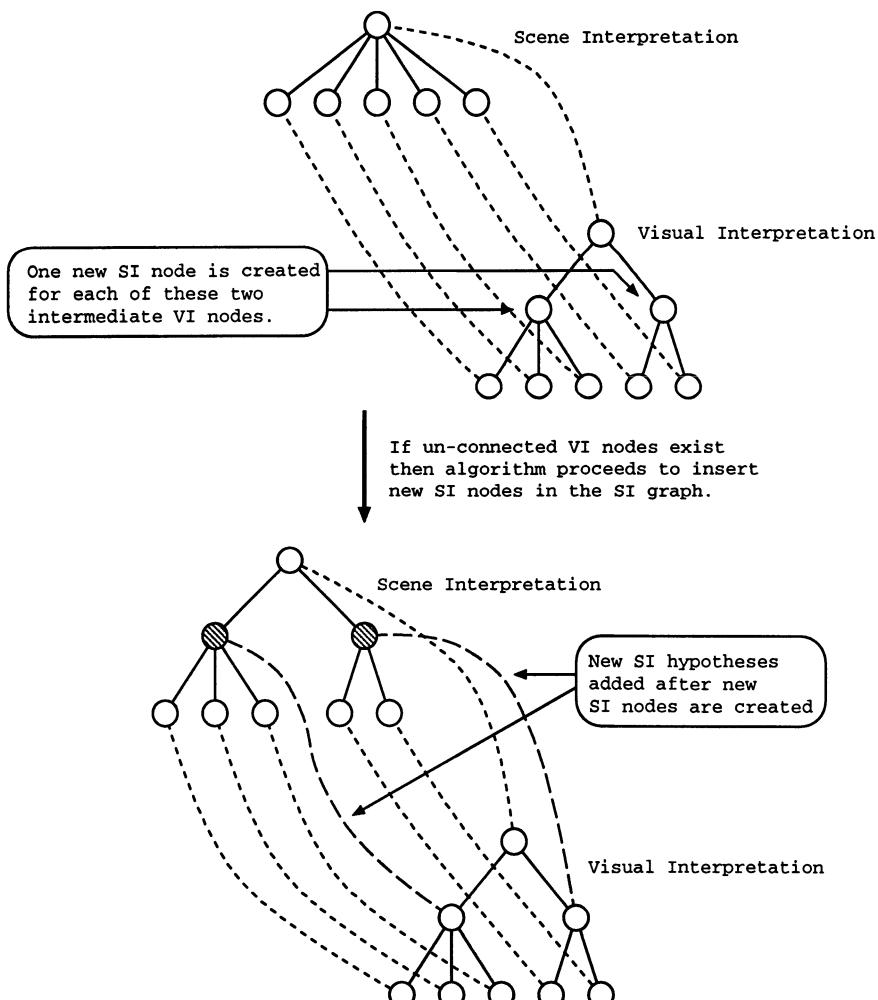


Figure 4.20: Description of ProcessSIFromVI Algorithm

objects do not address the clique membership problem.

The clique membership problem is solved in *Cite* using interaction between a number of different processes. Operating on the VI graph, the SEGMENTGROUPCLIQUE process generates sets of possible VI parent nodes, each of which represents a possible clique. Intermediate SI nodes are then generated by the MATCHSIFROMVI process operating on the SI graph. These generated nodes then interact with the normal relaxation labelling update processes until stable hypothesis weights are achieved.

The clique membership process SEGMENTGROUPCLIQUE operates on groups of VI nodes that have SI hypotheses which cover KB nodes with different parents. This situation is better understood by examining Figure 4.21. The VI node on which the algorithm is being run is indicated as the “current vi”. This VI node has a number of children, each of which has an SI hypothesis. The SI nodes each have hypotheses linking them to the knowledge base. One can see in Figure 4.21 that children of objects labelled “A”, “B” and “C” have each been hypothesised.

The objective of the clique membership process is to produce possible groupings of VI nodes based on their common KB parent hypotheses. This is done by indexing the VI nodes against the children of each node in the parent KB node list (denoted “kbplist” in Figure 4.21). A full set of possible matching child VI nodes is constructed by enumerating these index sets.

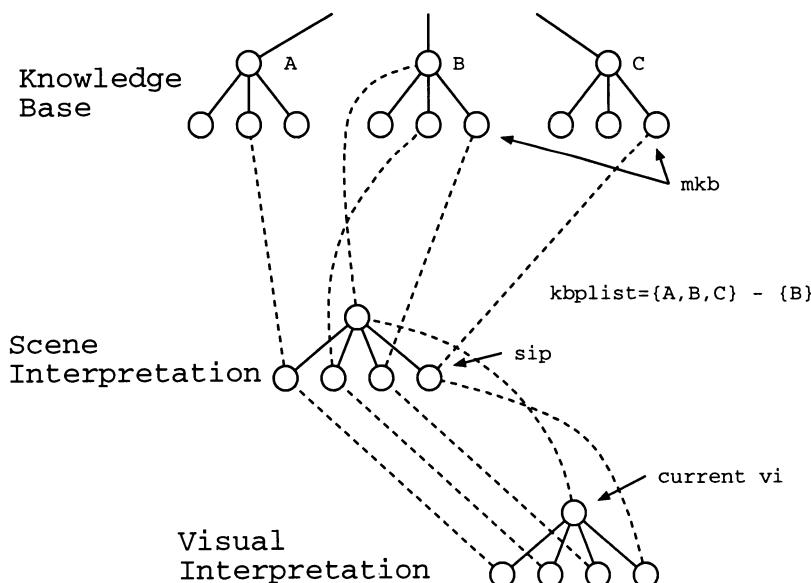


Figure 4.21: Illustration of Variables for Clique Membership Algorithm

The typical result is a small number of possible cliques that match each KB parent node. Following this, the clique membership process generates all of what now becomes an intermediate layer of VI nodes, each one representing part of a possible decomposition of the original set of VI nodes (the children of the “current vi” node).

An important consequence of this type of approach to solving the clique membership problem is that it is essentially a local operation. Once created, each clique is treated independently of the others. Multiple parent hypotheses in the VI graph are resolved through relaxation labelling processes. It is important to note that the local nature of this clique membership resolving algorithm is local *with respect to the knowledge base*, not with respect to the image. This is an important knowledge-based constraint that is based on the *structure* of the knowledge base, and not on the structure of the viewed image.

#### 4.6.10 Hypothesis Generation by KB Matching

When groups exist in the SI graph they need to be matched against the knowledge base to determine the most likely object labelling. This involves examining the unary matching strengths in addition to executing the binary matching strategy. *Cite* uses unary matchings to prune the possible parent object labellings, and the process which conducts the matching is the PROCESSMATCHKB operator. In this, possible parent labellings are pruned by the child matches, and the degree of match which is finally loaded into the parent SI node hypothesis combines unary and binary matching strengths.

### 4.7 Relaxation Labelling with Hierarchical Constraints

There are typically multiple labelling and grouping hypotheses generated by *Cite* for any image region or set of image regions. These multiple hypotheses are resolved by a process of relaxation labelling and constraint propagation.

Relaxation labelling is a method of resolving multiple hypothesis labellings with respect to a compatibility function describing the compatibility between pairwise labellings. This method is essentially gradient descent, and consequently cannot guarantee a globally optimal solution [292]. However, the iterative nature of the relaxation labelling process is ideal for the purposes of *Cite*.

#### 4.7.1 Non-Hierarchical Relaxation Labelling

Relaxation labelling [146] is a process in which initial probabilities for object labellings are updated iteratively according to predefined compatibilities of

these object labellings. One property of this process is the ability to propagate local constraints through the interaction of compatible or incompatible labels.

Let  $\mathbf{B}$  be a set of objects  $\{b_1, \dots, b_n\}$ , and  $\Lambda$  be a set of labels  $\{1, \dots, m\}$ . For each object  $b_i$  we can obtain an initial label probability vector  $\mathbf{P}_i^0 = (p_{i1}^0, \dots, p_{im}^0)$  where  $0 \leq p_{ij}^0 \leq 1$ , for  $i = 1 \dots n$  and  $j = 1 \dots m$ , and  $\sum_j p_{ij}^0 = 1$ , for  $i = 1 \dots n$ .

Each initial probability vector is interpreted as the prior probability distribution of labels for that object, and can be computed from the initial unary and binary matching. The compatibility constraints can be described as a  $n \times n$  block matrix  $\mathbf{R}$ , where each  $R_{ij}$  is a  $m \times m$  matrix of non-negative real-valued compatibility coefficients, denoted  $r_{ij}(1..m, 1..m)$ . The coefficient  $r_{ij}(\lambda, \mu)$  is a measure of the compatibility between object  $b_i$  being labelled  $\lambda$  and object  $b_j$  being labelled  $\mu$ . The relaxation labelling algorithm iteratively updates the probability vectors  $\mathbf{P}$  using a normalised weighted sum equation:

$$p_{i\lambda}^{t+1} = \frac{p_{i\lambda}^t q_{i\lambda}^t}{\sum_{\mu=1}^m p_{i\mu}^t q_{i\mu}^t} \quad (4.2)$$

where the denominator is the normalisation factor and:

$$q_{i\lambda}^t = \sum_{j=1}^n \sum_{\mu=1}^m r_{ij}(\lambda, \mu) p_{j\mu}^t \quad (4.3)$$

The objective is to end up with a unique label for each object. Depending on the compatibility coefficients  $r_{ij}(\lambda, \mu)$  it is not possible to guarantee convergence. In this form of relaxation labelling, the number of objects is constant and the number of iterations,  $t$ , is the same for each object. However, *Cite* contains hierarchical knowledge and can generate and remove image regions dynamically. Static hierarchical relaxation labelling has been discussed briefly in the literature [79], but not considered further.

#### 4.7.2 Relaxation Labelling in *Cite*

To simplify the notation, the relaxation labelling process is described in terms of operations on the probabilities listed in Table 4.1. Relaxation labelling is achieved in a two stage process. The first stage involves computing the un-normalised estimator for the probability, and the second involves normalising this against other competing hypotheses.

There are a total of six different types of hypothesis in *Cite*. There are two unary existence hypothesis types (SI and VI), two inter-graph hypothesis types (VI-VI and SI-SI), and two intra-graph hypotheses (VI-SI)

and SI-KB). In many respects, the most important of these is the SI-KB hypothesis which represents the object labelling of each scene element.

One advantage in having a hierarchical scene description is that these hierarchies can be used to prune the hypothesis update process. This effectively puts large blocks of 0-value into the compatibility matrix  $\mathbf{R}$  in situations where the hierarchical structure of the knowledge base would be violated. As such, it is more efficient to recast the relaxation labelling process as a hypothesis update process and only store non-zero hypotheses.

The sparse compatibility matrix used in *Cite* also needs to be dynamically recomputed with the creation and destruction of hypotheses. It is more efficient to decompose the compatibility matrix down to constituent parts applicable to each of the six types of hypothesis, and explicitly represent these as dynamic calculations which are computed at each iteration. These equations are dynamic in that the index sets over which the partial summations are computed can change from one iteration to the next.

### Updating SI-KB Link Hypotheses

SI-KB link hypotheses represent the best matches of each SI node to nodes in the knowledge base. They are updated according to the level of support given by the SI children *and* SI parents, as follows:

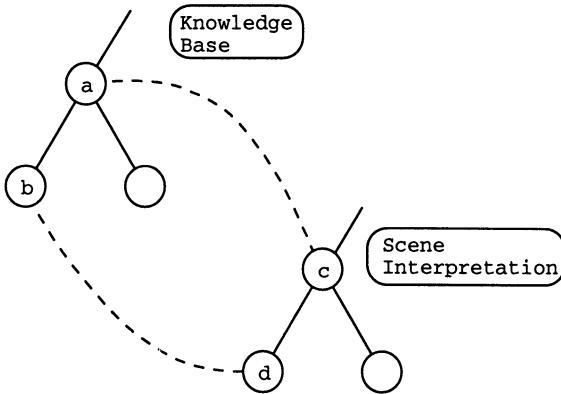
$$\hat{P}_{i,j}^{SK} = (1 - \alpha_{pc}) \sum_{\lambda \in S_i^C} P_\lambda^S P_{\lambda,i}^S \sum_{\xi \in K_j^C} P_{\lambda,\xi}^{SK} + \alpha_{pc} \sum_{\lambda \in S_i^P} P_\lambda^S P_{i,\lambda}^S \sum_{\xi \in K_j^P} P_{\lambda,\xi}^{SK} \quad (4.4)$$

The initial hypothesis value is set by the unary and/or binary matching from the operator which created the given SI-KB hypothesis. The update ratio of parent and child support ( $\alpha_{pc}$ ) reveals some asymmetry in the update procedure in terms of the relative importance of children and parents. This is necessary because, in general, there are fewer parents of a SI node than children.

In Equation 4.4, the double summations represent the summing over what are termed *compatibility cycles* between the scene interpretation and knowledge base graphs. Figure 4.22 illustrates a compatibility cycle including knowledge base nodes (a) and (b) and scene interpretation nodes (c) and (d). The compatibility cycle is a cycle comprising four hypotheses and is computed through the parent chain (right half of Equation 4.4) and through the child chain (left half of Equation 4.4). There are only three terms, rather than four, in each half of the update equation because the parent-child knowledge base link has a set hypothesis weight of 1.0. An extension to the knowledge base facilitating partial belief in the knowledge structure could be achieved by including a non-unity term into this equation.

### Updating SI Link Hypotheses

The SI link hypotheses are only updated in a parent-looking direction. This is done simply because the SI-SI hypothesis links are undirected and the same weight applies to the parent→child interpretation as

Figure 4.22: Update Diagram for  $P_{i,j}^{SK}$ 

the child→parent interpretation of the hypothesis. The SI-SI hypothesis update method is straight forward, with each parent weight being updated according to the unary hypothesis strength for the existence of that SI node:

$$\hat{P}_{i,j}^S = P_{i,j}^S P_j^S \quad (4.5)$$

#### Updating SI Existence Hypotheses

The SI existence hypothesis,  $P_i^S$ , is updated from the sum of the visual interpretation support and the SI-KB hypothesis links, as follows:

$$\hat{P}_i^S = \frac{1}{\|\{j : i \in V_j^S\}\|} \sum_{\lambda \in \{j : i \in V_j^S\}} P_{\lambda,i}^{VS} + \frac{1}{\|S_i^K\|} \sum_{\lambda \in S_i^K} P_{i,\lambda}^{SK} \quad (4.6)$$

This is clipped to the range [0.0,1.0], and saturates when there is firm belief that the SI node in question actually is a real scene element. When either the matching support is removed (bad matches, or competing cliques), or when the visual interpretation support is removed (better labelled cliques, or better cliques resulting from re-segmentation) the SI hypothesis strength drops rapidly to zero.

#### Updating VI-SI Link Hypotheses

The VI-SI links are updated subject only to the unary SI and VI probabilities, as follows:

$$\hat{P}_{i,j}^{VS} = P_{i,j}^{VS} P_i^V P_j^S \quad (4.7)$$

#### Updating VI Link Hypotheses

The VI-VI hypotheses are initialised to unity and then normalised at each update stage in a parent-traversing direction weighted by the VI unary hypotheses. The update equation is simply:

$$\hat{P}_{i,j}^V = P_{i,j}^V P_j^V \quad (4.8)$$

### Updating VI Existence Hypotheses

The visual interpretation nodes are updated according to the number of pixels not uniquely associated with that region and according to the SI unary hypothesis strength for any SI nodes that the VI node is hypothesised to, as follows:

$$\hat{P}_i^V = \left( \sum_{\lambda \in V_i^S} P_{i,\lambda}^{VS} P_\lambda^S \right) - \alpha_{mr} \text{PixelOverlapRatio}(i) \quad (4.9)$$

The `PIXELOVERLAP` function returns the fraction (between 0.0 and 1.0) of pixels in multiple regions, and  $\alpha_{mr}$  is a mixing coefficient (default value of 0.1).

## 4.8 Knowledge Driven Segmentation

The purpose of image segmentation is to group pixels into regions that represent parts or objects. Once these groupings have been made, recognition is achieved using computed properties of these parts or objects. A common misconception (caused undoubtedly by the simplicity of the approach) is that it is possible to perform image segmentation in a reliable manner using primitive attributes such as colour and texture with no prior knowledge of image content. There is no foundation for this belief, and *Cite* demonstrates the importance of using knowledge to drive the segmentation process.

In *Cite*, segmentation is a controlled process in which the current expectation of scene content is used to determine which segmentation algorithm and parameterisation is to be used. This is achieved by storing in each knowledge base node a list of those segmentation procedures and parameterisations which have been the most effective at segmenting the object or scene element represented by that node. Image segmentation may occur on any arbitrary sub-set of the image, and it is not uncommon for *Cite* to be executing half a dozen different segmentations at once. By closing the loop on segmentation, *Cite* is able to apply multiple segmentation algorithms and parameterisations to different regions of the image based on ongoing hypotheses of scene content. This provides the overall system with greater robustness to noise and distortion, and permits a much broader range of objects and scene elements to be recognised using the one methodology.

### 4.8.1 The Segmentation Cycle

Segmentation and re-segmentation occur continuously in *Cite*. Every node in the visual interpretation graph has a *segmentation stack* which contains the segmenter currently being executed on that part of the image represented by that node. Segmenters are added to the segmentation stack by bottom-up and top-down processes. The segmenters are executed effectively at random and will take varying amounts of time to compute depending on the algorithm being run and the size and complexity of the region.

When an image is first loaded the *default segmenter* is executed on this image. The default segmenter is the segmenter contained in the top level “World” knowledge node. Segments produced from this initial segmentation will then cause labelling hypotheses and part groupings to be generated, and in this way the scene interpretation structure will begin to take form. As the hypothesis generation and relaxation labelling processes proceed, there will be numerous occasions when *Cite* realises that although it may believe a certain object to exist in the scene, it may have an incorrect number of parts or badly matching parts. In these situations *Cite* consults the knowledge base to determine the best possible segmenter to apply to the sub-image of the ambiguous object or scene element, and loads this segmenter into the segmentation stack in that VI node. When the new segmenter has been executed, *Cite* may have a better idea of what object is present and may initiate another re-segmentation based on this refined knowledge.

#### The Re-segmentation Algorithm

Once an initial segmentation has been completed and hypotheses created, *Cite* tests these hypotheses to determine if sub-region re-segmentation is possible and appropriate. If the SI node currently being considered for re-segmentation has multiple KB hypotheses and the best of these is significantly better than the others, then this node is considered for re-segmentation according to the dictates of the KB node hypothesised. *Cite* accesses the knowledge base to determine the best segmenter to run on the region, and this is then loaded into the segmenter stack of the region’s VI node.

To determine the best segmenter to use at each re-segmentation stage, *Cite* accesses the most likely KB parent node. If this KB node has a segmenter, this is used. If it does not, the system recursively looks back up through the knowledge base to obtain the next closest node that does contain a segmenter. Once located, the segmenter is compared to those already run on the region by examining the *segmenter history stack* in the VI node for that region. If the new segmenter has not been previously executed on this region, the re-segmentation occurs.

An important graph maintenance function is undertaken when re-segmentation occurs. The SI nodes and their supporting VI nodes below the

SI node to be re-segmented are deleted. All SI nodes with only one parent and one child above the SI node to be re-segmented are also deleted. This second phase of deletion is important in situations where deep instantiations of taxonomic decomposition have been created before the re-segmentation process became possible.

### Interaction of Relaxation Labelling and Re-segmentation

The relaxation labelling and re-segmentation processes in *Cite* interact quite closely in a dynamic way, which is illustrated by a simple example of a two stage segmentation and recognition of an oak tree. Figure 4.23 illustrates four hypotheses as a function of time as the relaxation labelling and re-segmentation processes interact. In this example, the object in question is a single tree that matches well to the “oak” and to the “pencilpine” after the initial segmentation. The object has been imperfectly segmented (in this case under-segmented) and only one part is available for matching. The initial segmentation occurs at iteration 0, followed by relaxation to iteration 30 at which point the region is re-segmented. During the re-segmentation the hypothesis weights move back towards equal values because the bottom-up support VI nodes have been removed. At iteration 52, the new segmentation creates the new child nodes and the hypotheses are relaxed to their final levels. Without re-segmentation, a weakly matched under-segmented result would have been the best that could have been obtained.

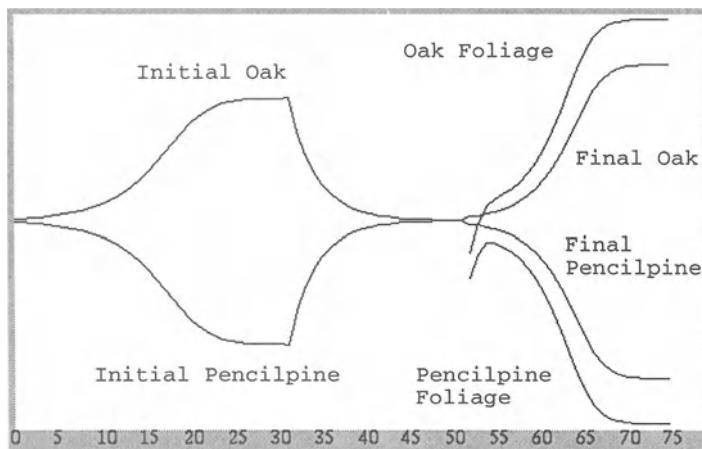


Figure 4.23: Hypothesis Graph Window Illustrating Selected Hypothesis Weights

### 4.8.2 Evidence of the Power of Re-segmentation

The Botanist knowledge base classification results are given as an illustration of the power of knowledge driven re-segmentation. The Botanist knowledge base used is shown in Figure 4.6 (in Section 4.4), and a sampling of the images covered by this knowledge base is shown in Figure 4.24.

The classification results are shown in Table 4.3. Each column in this table represents results obtained with a different default segmenter (the segmenter stored in the top-level knowledge base node). The system achieves an average classification rate of 99% when operating normally. Each instance where a re-segmentation has occurred is noted; for example (7→3→2) means that the initial segmentation gave 7 parts, the first re-segmentation gave 3 parts and the second re-segmentation gave 2 parts. If the knowledge driven re-segmentation is deactivated the average classification rate drops to 59%.

### 4.8.3 Segmentation Algorithms

All of the segmenters in *Cite* have a number of common properties. Each segmenter must be able to operate on an arbitrary subset of any of the images currently being processed. The segmenter must be able to operate on integrally scaled images, and to run in parallel with other segmenters running on the same or different regions.

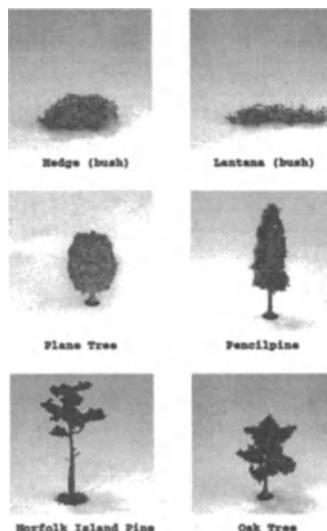


Figure 4.24: Sample of Images used in Botanist Knowledge Base

Image	NIPine Seg	PencilPine Seg	Oak Seg	Plane Seg
nipine1	Correct	Correct (2→3)	Correct	Correct (5→2)
nipine2	Correct	Correct	Correct (3→2)	Correct (7→3→2)
nipine3	Correct	Correct	Correct	Correct (4→2)
nipine4	Correct	Correct	Correct	Correct (3→2)
penciline5	Correct (1→2)	Correct	Correct (3→2)	Correct (7→3→2)
penciline6	Correct (1→2)	Correct	Correct	Correct (3→2)
penciline7	Correct (1→2)	Correct	Correct (1→2)	Correct (11→2)
penciline8	Correct (1→2)	Correct	Correct	Correct (5→2)
oak5	Correct (1→2)	Correct	Correct	Correct (4→2)
oak6	Correct (1→2)	Correct	Correct	Correct (5→2)
oak7	Correct (1→2)	Correct	Correct	Correct (4→2)
oak8	Correct (1→2)	Correct	Correct	Correct (9→2)
plane3	Correct	Correct	Correct	Correct
plane4	Correct (1→2)	Correct (1→2)	Correct (1→2)	Correct
plane5	Correct (1→2)	Correct (1→2)	Correct (1→2)	Correct
hedge1	Correct	Correct	Correct	Correct
hedge2	Correct	Correct	Correct	Correct
lantana1	Correct	Correct	Correct	Correct
lantana2	Correct	Incorrect	Correct	Correct

Table 4.3: Botanist Knowledge Base Classification Results

When a segmenter is loaded into a VI node for execution, it is initialised with the parameters stored in the best hypothesised knowledge base node. The three segmentation algorithms available in *Cite* are described briefly below.

### Segmentation by Clustering and Growing

The SEGMENTSEED algorithm initially builds a small RGB histogram to find seed points for region growing. The most common colour is chosen as a seed point and regions are then grown from all pixels of this colour. The regions may incorporate pixels of slightly different colours according to a colour distance threshold. This process is repeated until all pixels in the area of the image to be segmented have been labelled.

Each region in the label image is then relabelled so that the labels are unique. Regions are then grown to remove small regions. A small region is one whose size is less than a specifiable ratio of the total cover. In the region growing algorithm, regions are relabelled to the nearest neighbour with the highest matched colour. The region growing process continues until there are no small regions.

### Segmentation by Clustering and Ratio Growing

The SEGMENTERSEED algorithm was found to work well for a large number of objects, but failed on objects with low contrast parts. This was mainly because colour distance is calculated as an absolute measure, so could not detect changes in badly illuminated objects. A modified version of this segmenter was constructed which computed all colour differences as ratios rather than absolute differences.

### Segmentation by Edge Extraction and Merging

The SEGMENTEDGE algorithm initially performs a standard edge de-

tection based on a small local window and added across each of the three colour planes. The set of edge pixels  $E$  is created by thresholding all pixels in the edge map against an upper threshold. From these edge pixels, the edges are grown in an eight-connected manner against a lower threshold. This edge growing is iterated until there are no more pixels being labelled as edges.

Once edge growing is complete, uniquely labelled regions are created by a standard recursive paint filling algorithm. The edge pixels are then eroded away to leave a full covering of the region of the image being segmented. Regions below a specified fraction of the overall region size are then merged in a two phase operation. The first phase considers small regions only connected to other small regions, and then propagates their labels into other small regions. The second phase looks at all remaining small regions and relabels them as their closest large neighbour, for which there will always be at least one. This region merging is ideal for complex textured regions which yield a large number of small regions placed close together. In cases where small regions may be generated not only from large textured surfaces, a different segmenter can be chosen.

## 4.9 Feature Extraction

Segmentation and feature extraction are the two most dramatic data reduction stages in a machine vision system. Segmentation places pixels together into groups that have something in common, and feature extraction produces measures on those groups of pixels. Quite often an image region which might represent tens of kilobytes of data can be summarised by a few numbers after segmentation and feature extraction.

Features calculated on single regions (*unary* features) typically include colour, texture, size and shape descriptions of the region. Relational (*binary*) features are calculated on pairs of regions and typically concentrate on geometric properties such as the distance between the regions, length of common border and so on. It is conceivable (although not common) that binary features could describe unusual relational features such as how similarly coloured or textured the two parts were.

Cite uses both unary and binary features, which can be calculated across the visual interpretation hierarchy. This represents a broadly quantised contextual scale invariance in the feature extraction process that goes a small way to address this issue of such large information reduction.

### 4.9.1 Feature Storage and Calculation

When calculated, features are stored in the visual interpretation nodes. Binary features stored at a node describe the relationships between the

children of the node, whereas the unary features stored at a node describe the node itself, as illustrated in Figure 4.25.

For the purposes of presenting the learning and classification systems with uniform data, all unary and binary features within *Cite* are scaled to the range 0.0 to 1.0. In the case of features which have no upper bound, a reasonable scaling is applied and then the feature is clipped to 1.0.

#### 4.9.2 Unary Features

The choice of unary features calculated for each region is based on knowledge about the current scene interpretation and learning results, as governed by the matching algorithms in the knowledge base. Table 4.4 contains a list of the unary features.

Feature	Type	Description
Illumination	Colour	Average region illumination
Red	Colour	Average region red component
Green	Colour	Average region green component
Blue	Colour	Average region blue component
VarRed	Texture	Variance in region red component
VarGreen	Texture	Variance in region green component
VarBlue	Texture	Variance in region blue component
Size	Geometric	Ratio of region to parent
Elongation	Geometric	Ratio of diameter to area
HeightWidth	Geometric	Height to width ratio
Straightness	Geometric	Vertical straightness measure
AOnPSqr	Geometric	Area to Perimeter ratio
Extent	Geometric	Whether region has infinite extent

Table 4.4: Summary of Unary Features

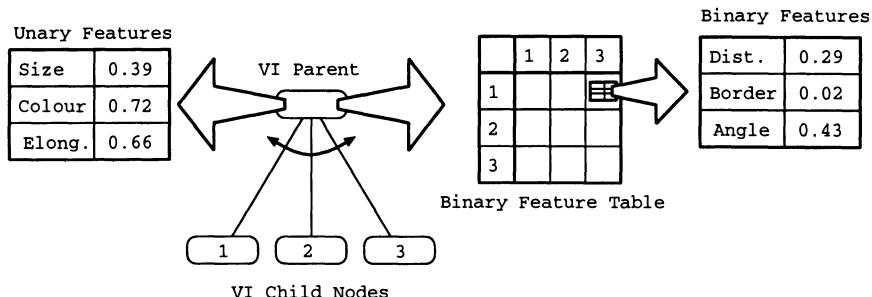


Figure 4.25: Storage of Unary and Binary Features in VI Nodes

The unary size feature of a region is given as the square root of the ratio of its size (in pixels) to its parent's size. A normalised *height-to-width* ratio of a region is calculated as the ratio between the height of the region and the sum of its height and width. The *elongation* of a region is defined as twice the ratio of the maximum diameter to the perimeter. A common region shape descriptor is the ratio of area to perimeter squared, called “A on P squared”, or “compactness”. This feature is unit-less and bounded by  $\frac{1}{4\pi}$  corresponding to a circle. An estimate of the vertical straightness of a region can be obtained by fitting a quadratic line to the mid-point of each scan line in the region. The straightness of the region is then inversely proportional to the coefficient of the second order term. The unary extent feature gives *Cite* an indication of whether a region is bounded. For example, sky and ground regions can be considered to have “infinite” extent.

### 4.9.3 Binary Features

Table 4.5 contains a list of the binary features available in *Cite*. A binary feature is said to be *directed* if  $F(a,b) \neq F(b,a)$ . Most learning and binary matching systems are designed to use directed features, and they incorporate undirected features simply by computing the feature using both region orderings. *Cite* adopts this strategy.

Feature	Description	Directed
Size Ratio	Ratio of image sizes	Yes
Distance	Image distance to size ratio	No
Boundary	Ratio of shared boundary	Yes
Above/Below	Proportion above/below region	Yes

Table 4.5: Summary of Binary Features

The binary size ratio feature provides an indication of how large or small one region is compared to another. This is computed as a non-linear ratio of region sizes bounded such that a region eight times larger (or greater) gives a size ratio of 1.0, and a region of identical size will give a size ratio of 0.5. A common binary feature is to compute the distance between the centroids of the two regions, scaled by their average area. Another common binary feature implemented in *Cite* is the fraction of shared boundary between two regions. Given the hierarchical nature of *Cite*'s segmentation structure, and the fact that pixels may belong to a number of regions at the same level in the hierarchy, there are two ways to compute the binary boundary feature, as illustrated in Figure 4.26.

The *boundary-region* method counts up all boundary pixels of region A that are within one (four connected) pixel of region B. The *boundary-boundary* method counts up all boundary pixels of region A that are within

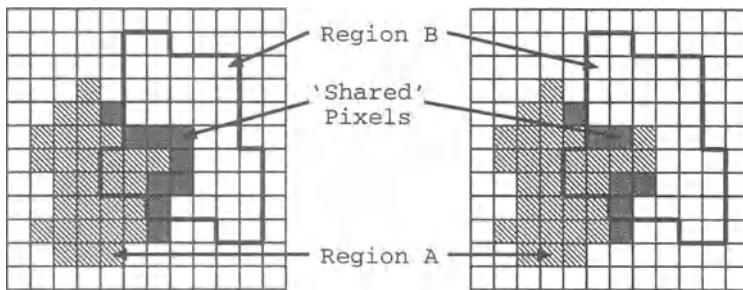


Figure 4.26: Two Methods for Computing Shared Boundary Ratio

one (four connected) pixel of the boundary pixels of region B. Both methods give identical results when regions are not permitted to overlap, but because in *Cite* this restriction is lifted, the boundary-region method is used. The actual binary boundary feature is the ratio of the overlapping boundary pixels in region A to the total boundary length of region A.

The above/below feature computes the number of pixels of region A that are above region B,  $n_{above}$ , and the number that are below region B,  $n_{below}$ , and computes the above/below feature as  $R_{abovebelow} = \frac{1}{2} \left( 1 + \frac{n_{above} - n_{below}}{N_a} \right)$ .  $N_a$  is the total number of pixels in region A, which means that  $R_{abovebelow}$  is correctly bounded between 0.0 and 1.0.

#### 4.9.4 Feature Invariances

An important characteristic of both unary and binary features is their invariance characteristics. Of most importance to *Cite* is invariance to two dimensional translation, rotation, scale and illumination. Translation invariance is typically the easiest to obtain, however for reasons of global positioning of objects, sometimes features are used that are not invariant to translation.

The colour and texture unary features are invariant to translation, rotation and scale, but not to illumination. The geometric unary features are invariant to all four types of perturbation, although the Height-Width and Straightness measures are (deliberately) variant under rotation.

The binary features in *Cite* are all invariant to illumination, rotation, scale and translation with the exception of the Above-Below feature which varies significantly with rotation. The image distance feature is the only binary feature in *Cite* that is symmetric (undirected).

## 4.10 System Performance and Results

*Cite* is demonstrated on four scenarios; views of street scenes, views of office objects, aerial views of airports, and the context sensitive tree taxonomies. The results summary for these scenarios is listed in Table 4.6. In total, 253 scene elements were detected from 38 images of the four scenarios. The knowledge bases for the scenarios were constructed from a collection of 85 images. The knowledge bases themselves contain a total of 155 scene elements at different levels of abstraction. Of the 253 detected scene elements, an average classification success of 96% was achieved.

Scenario	KB Elements	Elements Tested	% Correct
Street Scenes	24	38	95
Office Scenes	64	73	95
Airport Scenes	19	70	100
Tree Taxonomies	48	72	93
Total	155	253	96

Table 4.6: Summary of Results Presented

### 4.10.1 Views of Street Scenes

The street scene scenarios provide a simple demonstration of *Cite* in a general environment. The knowledge base was constructed from isolated and in-situ examples of objects. In the case of objects of a finite extent such as the buildings and vehicles, these were learnt from isolated examples. The objects of unlimited extent such as the various ground covers and the sky were learnt from landscapes containing just those scene elements.

In the first town image example *Cite* correctly recognises the objects present on the first pass and then uses the knowledge driven re-segmentation to obtain a finer resolution description. The resulting object labels and segment boundaries are shown overlaying the original image in Figure 4.27. The results are shown both at the leaf node description (most detailed) and at the next level up.

In the more complex street scene shown in Figure 4.29 a total of 24 scene elements have been detected. Of these, one is a mis-classification and three are the result of spurious segmentations. One of the spurious segmentations, the shadow under the fuel-truck, generates an incorrect “trunk” object. The other two are under and over segmentations, but are correctly labelled as “grass”. Figure 4.30 shows the full text description generated by *Cite*, including the groupings of the low level parts into constituent objects.

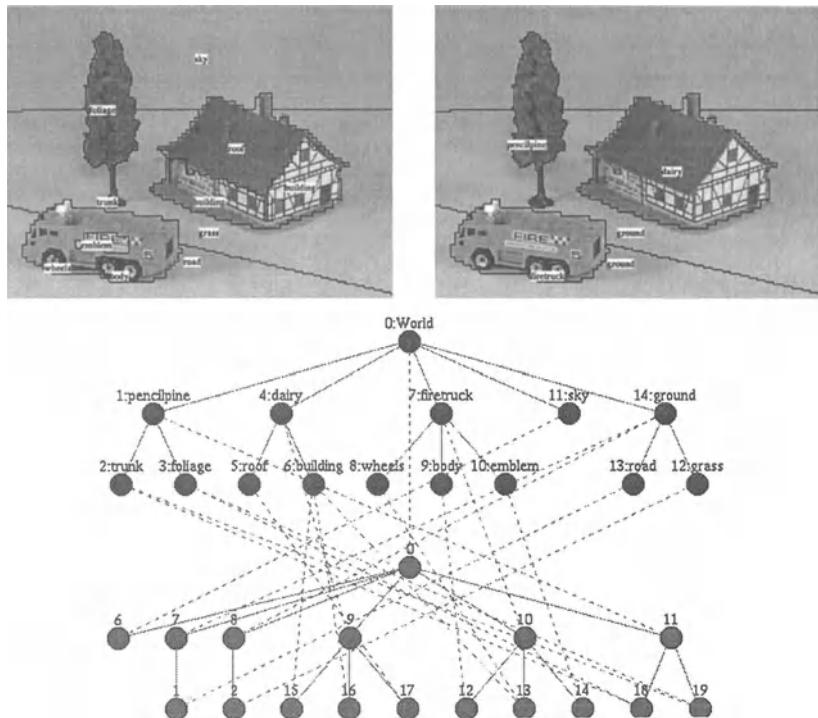


Figure 4.27: Images and Analysis Graph of Simple Street Scene

```

World[0] (1.000) Consisting of:
└── sky[6] (1.000)
└── ground[7] (1.000) Of Type:
    └── road[1] (1.000)
└── ground[8] (1.000) Of Type:
    └── grass[2] (1.000)
└── dairy[9] (1.000) Constructed from:
    └── building[15] (1.000)
    └── building[16] (1.000)
    └── roof[17] (1.000)
└── firetruck[10] (1.000) Constructed from:
    └── body[12] (1.000)
    └── wheels[13] (1.000)
    └── emblem[14] (1.000)
└── pencilpine[11] (1.000) Constructed from:
    └── trunk[18] (0.981)
    └── foliage[19] (0.981)

```

Figure 4.28: Text Description of Street Scene

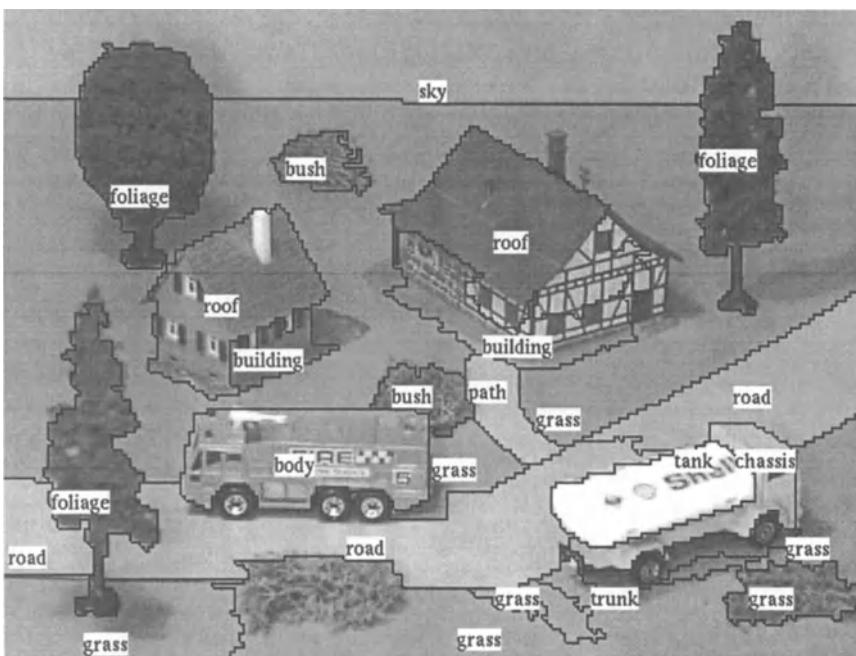


Figure 4.29: Images and Analysis Graph of Complex Street Scene

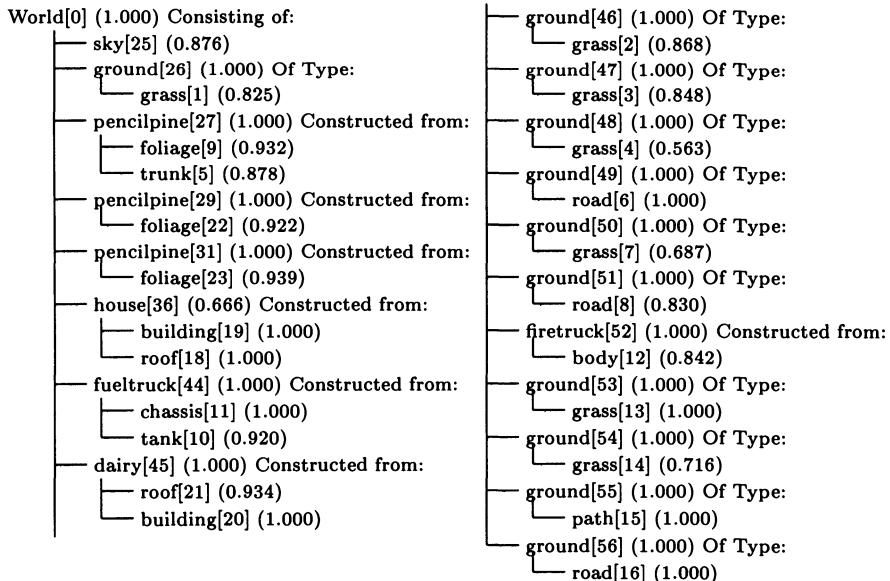


Figure 4.30: Text Description of Street Scene

### 4.10.2 Views of Office Objects

This section illustrates *Cite* in operation recognising small collections of objects from a database of twenty objects arranged in a hierarchy containing sixty four elements. The full knowledge base is listed in an expanded text form in Figure 4.31. Following this are two pages showing the results for the system recognising small collections of partially overlapping objects. Figure 4.32 shows perfect recognition and scene decomposition of four objects.

Four of five objects in Figure 4.34 are correctly recognised, with the tea-cup being mis-matched for part of a type of mug. This small error can be corrected with subsequent incremental supervised learning, but has been included to demonstrate one possible failure mode of *Cite*. In total, five previously unseen images were tested using this knowledge base, with an average correct labelling rate of 95%.

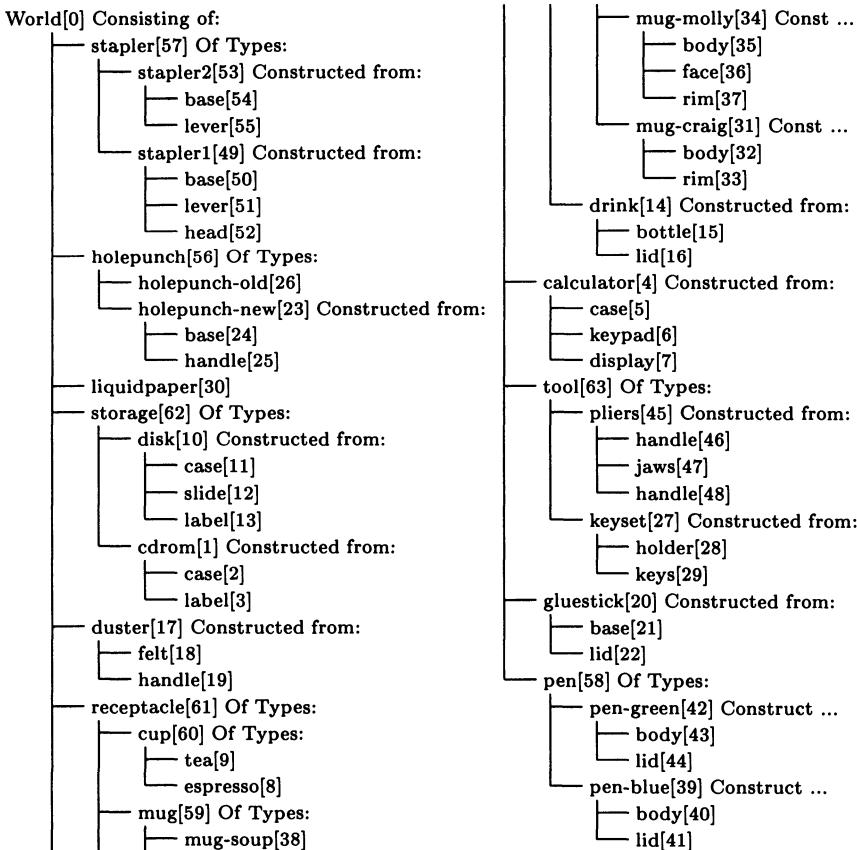


Figure 4.31: Text Description of Office Knowledge Base

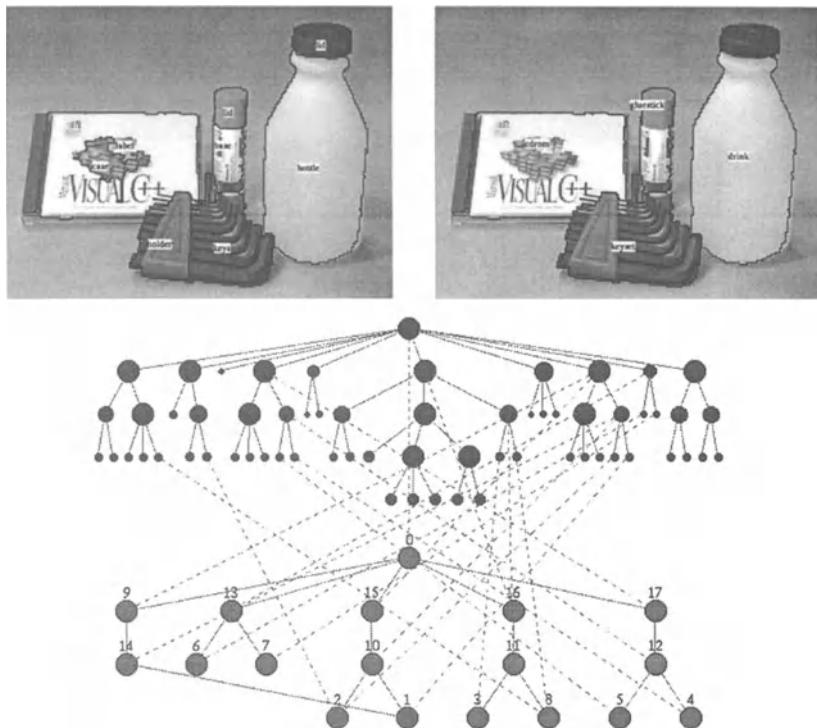


Figure 4.32: CD, Keyset, Gluestick and Drink Scene

```

World[0] (1.000) Consisting of:
  └── gluestick[13] (1.000) Constructed from:
    └── base[6] (1.000)
      └── lid[7] (0.831)
  └── tool[15] (1.000) Of Type:
    └── keyset[10] (1.000) Constructed from:
      └── holder[2] (1.000)
        └── keys[1] (1.000)
  └── receptacle[16] (1.000) Of Type:
    └── drink[11] (0.896) Constructed from:
      └── bottle[3] (0.906)
        └── lid[8] (0.891)
  └── storage[17] (1.000) Of Type:
    └── cdrom[12] (0.545) Constructed from:
      └── case[5] (0.725)
        └── label[4] (0.706)

```

Figure 4.33: Text Description of CD, Keyset, Gluestick and Drink Scene

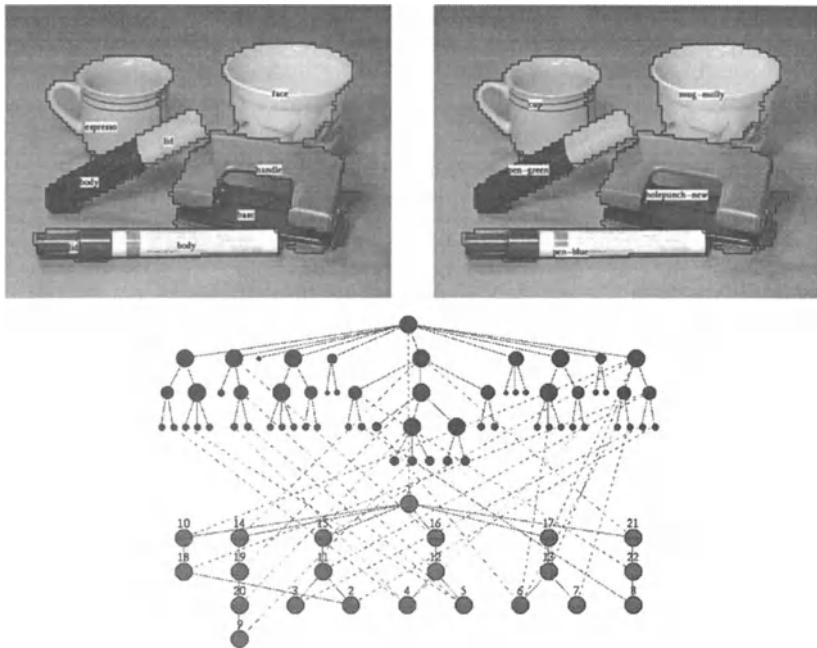


Figure 4.34: Two Pens, Two Cups and Holepunch Scene

```

World[1] (1.000) Consisting of:
  └── receptacle[14] (1.000) Of Type:
    └── mug[19] (1.000) Of Type:
      └── mug-molly[20] (1.000) Constructed from:
        └── face[9] (1.000)
  └── pen[15] (1.000) Of Type:
    └── pen-blue[11] (1.000) Constructed from:
      └── body[3] (1.000)
      └── lid[2] (1.000)
  └── holepunch[16] (1.000) Of Type:
    └── holepunch-new[12] (1.000) Constructed from:
      └── base[4] (1.000)
      └── handle[5] (1.000)
  └── pen[17] (1.000) Of Type:
    └── pen-green[13] (1.000) Constructed from:
      └── body[6] (1.000)
      └── lid[7] (1.000)
  └── receptacle[21] (1.000) Of Type:
    └── cup[22] (1.000) Of Type:
      └── espresso[8] (1.000)

```

Figure 4.35: Text Description of Pens, Cups and Holepunch Scene

### 4.10.3 Aerial Views of Airports

The objective of this section is to demonstrate the operation of the clique resolving process and the hierarchical relaxation labelling as a constraint propagation mechanism. To demonstrate these features of *Cite*, seven aerial images of an airport are analysed to determine not only the labelling of the constituent parts, but the local configurations that they are in. These local configurations have been chosen to heavily overlap so that the system must rely on relational information to resolve ambiguities.

The full knowledge base is shown in Figure 4.36. This was constructed by learning each object from a single image, followed by learning each higher level configuration from full scene examples. As can be seen, the “plane” object plays a central role, being involved in five higher level objects; the “loading”, the “emergency”, the “service”, the “landing” and the “refuelling” objects. When a plane is detected, the system must use relational information to determine which of the five possible parent objects the plane is most likely to belong to, and then propagate this through the scene interpretation graph using relaxation labelling. The clique resolving operator generates the most likely set of possibilities, and relaxation labelling resolves these to determine a subset of high compatibility.

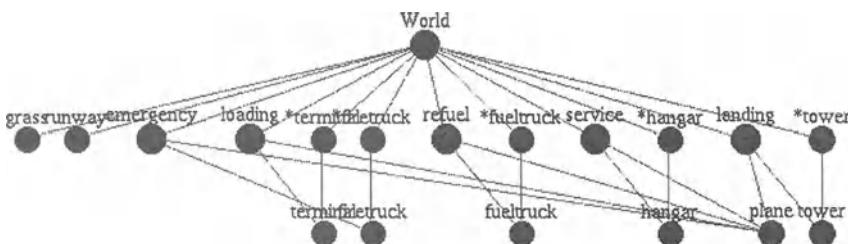


Figure 4.36: Full Knowledge Base used in Airport Analysis

A further complication in these scenes is that a number of objects in the database can appear in isolation, or as part of a higher level object. The determination for this in each case rests solely on the strength of the binary matching, and the propagation of the resulting hypotheses through the scene interpretation structure.

The first result (Figures 4.37-4.38) shows one single instance situation and the second result (Figures 4.39-4.40) shows a situation in which two planes are involved. In this second test, *Cite* must determine not only if a particular situation is occurring, but which objects are involved in that situation. In all cases this is determined solely by the propagation of relational matching through the hierarchical scene description. Five other images were tested and *Cite* correctly recognised the various objects and the higher level situations in which they were involved.

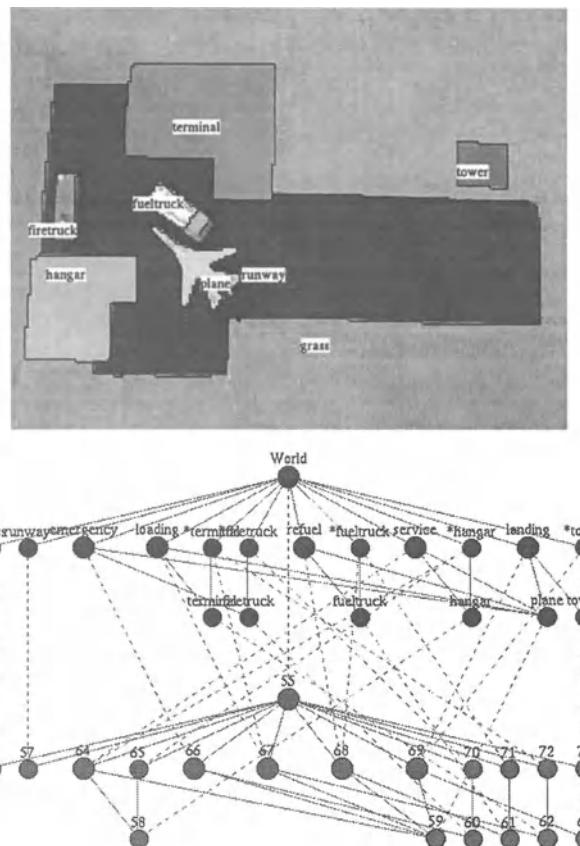


Figure 4.37: Refuelling Image and Analysis Graph

World[55] (1.000) Consisting of:  
 └── grass[56] (1.000)  
 └── runway[57] (1.000)  
 └── \*hangar[65] (1.000)  
 └── refuel[68] (0.946) Constructed from:  
     └── fueltruck[61] (1.000)  
         └── plane[59] (1.000)  
     └── \*firetruck[70] (1.000)  
     └── \*terminal[72] (1.000)  
     └── \*tower[73] (1.000)

Figure 4.38: Text Description of Refuelling Scene

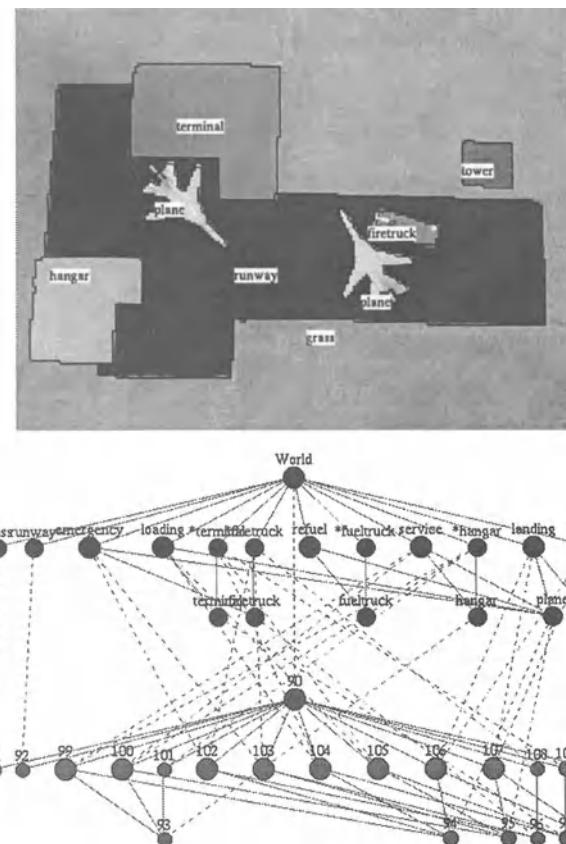


Figure 4.39: Emergency and Load Image and Analysis Graph

```
World[90] (1.000) Consisting of:  
  └── grass[91] (1.000)  
  └── runway[92] (1.000)  
  └── *hangar[101] (1.000)  
  └── emergency[102] (0.946) Constructed from:  
    └── plane[94] (1.000)  
    └── firetruck[96] (1.000)  
  └── loading[105] (0.946) Constructed from:  
    └── plane[95] (1.000)  
    └── terminal[97] (1.000)  
  └── *tower[110] (1.000)
```

Figure 4.40: Text Description of Emergency and Load Scene

## 4.11 System Operation

*Cite* is implemented as a single X-Windows based Unix program. The graphical user interface uses the Motif widget set, and was developed on a Silicon Graphics Indy using GNU C++. The system dynamically determines whether the live video capture hardware is present, and if so allows the user to grab images directly. However, most of the testing of *Cite* was done using pre-saved images which can be loaded into *Cite* at any point. Each of the main windows of the *Cite* system is described in the following paragraphs.

### The Command Window

The command window is the main control window in *Cite*. It lets the user enter commands from pull-down menus or from a one line text command prompt. Figure 4.41 shows the command window as it normally appears. The top text window contains the status and progress reporting messages from *Cite*. The middle text window is a history of the previous entered commands, and the lower text window is the command prompt.

Under the main menu in the command window are menu options which duplicate the commands that can be entered at the command prompt. In addition, there are menu options for displaying a live video window and for grabbing the live video input and saving it to a file (Silicon Graphics

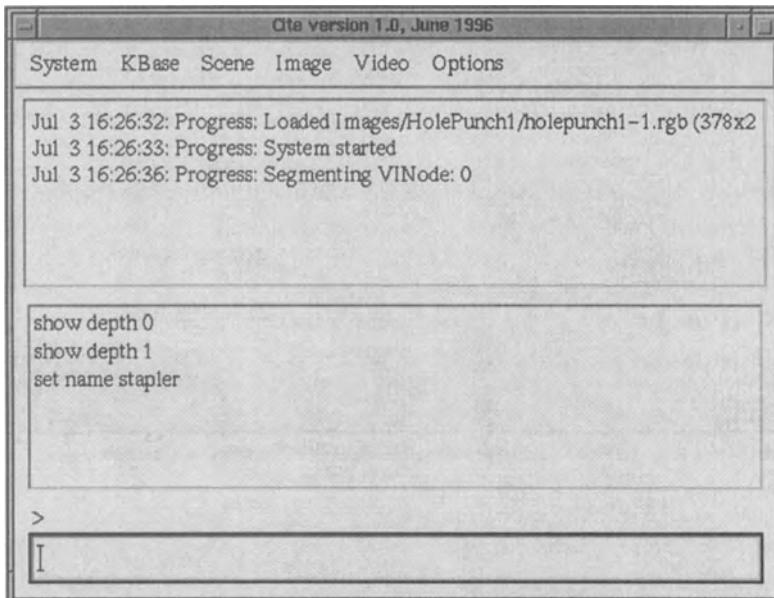


Figure 4.41: The Main Command Window in *Cite*

machines only). There is an additional window under the “Options” window which contains a slide-bar for each of the real-valued options under the “set” command.

### The Image Window

The image window contains the images of the current scene and their segmentations. When labelling is complete the labels are placed over the segmented image. The user can choose to display the segmentation at any level, such as at leaf nodes, at parents of leaf nodes, and so on.

### The Data Graph Window

The data graph window is the most important visual display in *Cite*. It contains three panels arranged vertically. The top panel contains the knowledge base, the middle panel contains the scene interpretation, and the bottom panel contains the visual interpretation for each image.

There are many display options which control what is displayed on the screen. This includes displaying hypotheses from the visual interpretations to the scene interpretation, and from the scene interpretation to the knowledge base. The user can also select nodes for editing, grouping or deleting

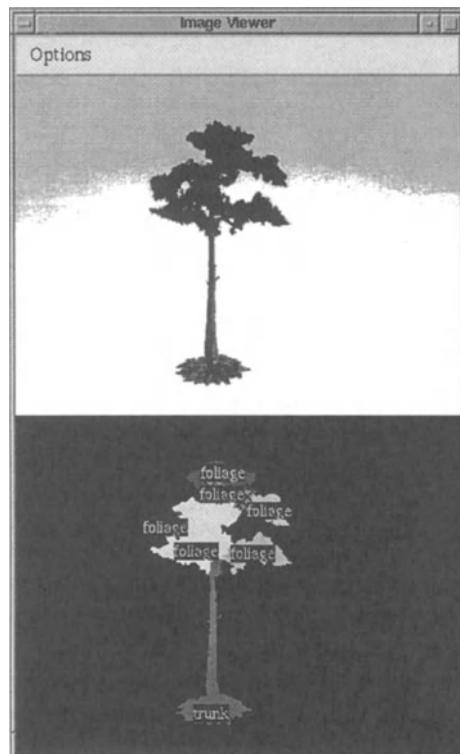


Figure 4.42: The Image and Segmentation Window

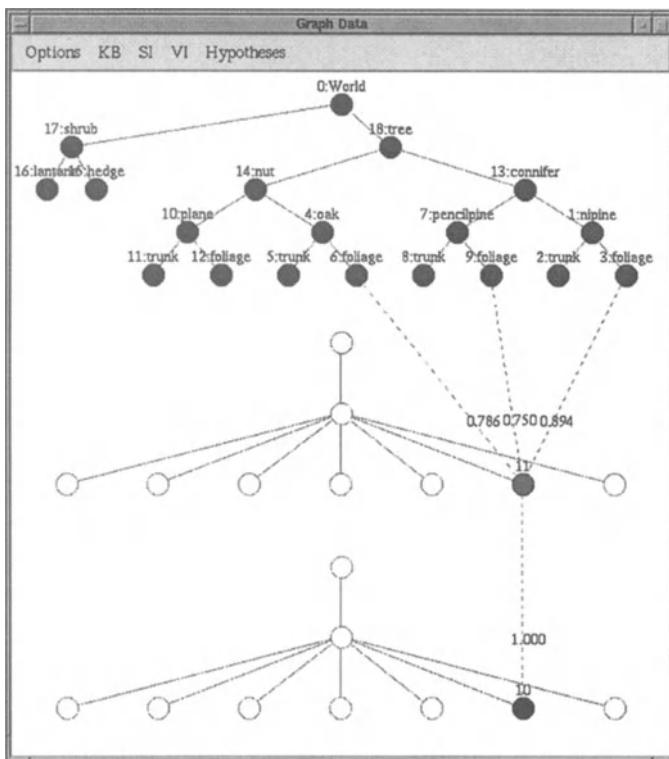


Figure 4.43: The Data Graph Window

with the mouse. The data graph window is shown in Figure 4.43.

The menu options in this window enable various levels of display detail of hypotheses. There is a considerable amount of information in the three graphs represented in this window, and to display all this at once would be virtually impossible. Nodes may be selected by clicking on them with the left mouse button. Clicking on them with the right button brings up the display window for that node. There are three basic types of node display window; one for VI nodes, one for SI nodes and one for KB nodes.

#### The Hypothesis Weight Window

A more detailed analysis of the hypothesis weights can be obtained via the hypothesis weight window. This displays a graph of hypothesis weights as a function of time. The weights that are displayed can be individually selected from the node display windows by clicking on the text display of the particular hypothesis of interest. Figure 4.44 shows the hypothesis weight window with three SI-KB hypotheses selected. Every hypothesis has a unique integer identifier, and this is displayed on this window as well as the node display window.

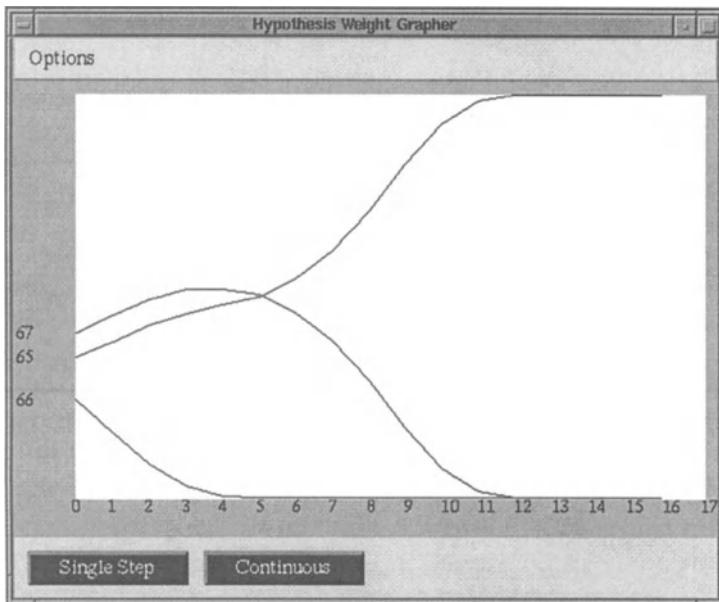


Figure 4.44: The Hypothesis Graphing Window

The horizontal scale in the hypothesis weight window is the iteration number where each iteration represents one complete pass through the process stack when a change occurred in at least one hypothesis. This prevents the system running to infinity with subsequent loss of the interesting detail of the hypothesis curves. The horizontal axis is auto-scaling and the vertical axis ranges between 0.0 and 1.0.

#### The Profile Window

*Cite* has a cyclic process scheduler which runs each loaded operator for a short period of time before going to the next. The total amount of time spent in each operator is stored in the operator description class, and displayed in the profile window shown in Figure 4.45. The columns in the profile window show the total time spent executing each operator, the overall percentage, a short-term percentage, a long-term percentage, and the absolute time spent in the most previous execution. In addition there is a performance measure expressed relative to the original development platform of a 150 MHz R4400 processor. The profiler gives useful information as to the relative computation required for each sub-task within *Cite*.

#### Results Window

In order to produce the highest quality results, there is an additional results window which shows the image in the original size with black-bordered regions and the region labels in a more printable font. The results window

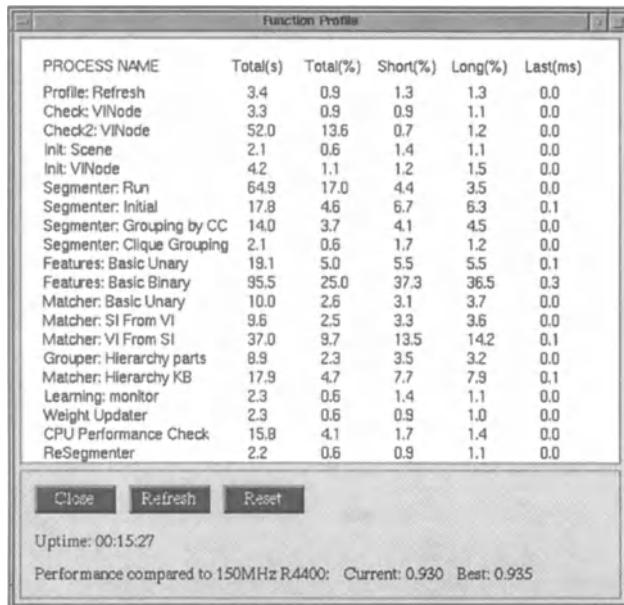


Figure 4.45: The Operator Profile Window

is shown in Figure 4.46 and is displayed from a menu entry under the main system menu in the command window. The results shown in Section 4.10 were generated using this window.

## 4.12 Conclusion

This chapter has presented a new theory of machine vision which integrates bottom-up and top-down processing to integrate object recognition and scene understanding capabilities within the same framework. The more novel aspects of this theory are in the use of hierarchical structures to describe world knowledge in addition to scene and visual decompositions, knowledge driven re-segmentation, incremental supervised learning methods, and hierarchical relaxation labelling.

Explanatory least generalisation on decision trees is presented as one method of converting non-incremental learning algorithms into an incremental form that obeys the identified desirable qualities of sub-linear growth in memory and computational complexity. Results are presented illustrating the performance improvement gained by closing the loop on segmentation with the knowledge driven re-segmentation algorithm.

This chapter also describes the *Cite* system which has been built to experimentally test the theories presented. *Cite* provides an ideal test-

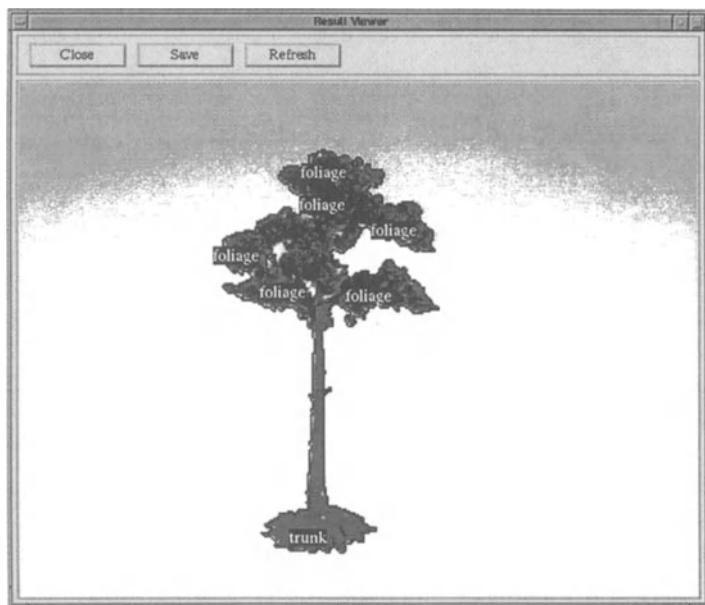


Figure 4.46: The Results Window

bed for comparing various segmentation, feature extraction, matching and learning algorithms.

# Chapter 5

## See<sup>++</sup> : An Object Oriented Theory of Task Specific Vision

Tom Drummond and Terry Caelli

### Abstract

Building machine vision systems remains difficult despite advances in both low-level image processing and high-level knowledge-based reasoning. In this chapter, a theory is proposed which integrates low-level and high-level processes into a task-specific vision system, See<sup>++</sup>. Supervised incremental machine learning is used to acquire and embed task-specific knowledge within a lattice structure which forms the knowledge base. Low-level image processing requirements are provided by an image query language which is controlled by feedback from the knowledge base as a function of partial image interpretation. The See<sup>++</sup> framework has application to a spectrum of problems such as teleoperation in a low bandwidth environment. This domain is discussed and used to illustrate the system.

## **5.1 Introduction**

Advances in computer technology have made the building of machine vision systems a tractable proposition. Indeed, this is partly responsible for the enormous body of research on all aspects of the problem, from digital signal processing of images to knowledge based systems and interpretation strategies. However, building such systems remains a difficult problem and it is becoming increasingly apparent that work on low-level image processing and high-level image and scene interpretation must be integrated into complete systems.

A systems philosophy makes it possible to address two important issues that are difficult to consider when subsystems, such as segmentation strategies, are constructed in isolation. Firstly, it is possible to verify the behaviour of component subsystems with respect to the task for which the system has been built. Secondly, correct interaction mechanisms between these subsystems, including feedback, can be established.

The See<sup>++</sup> project is concerned with tasks which depend on object recognition or scene understanding. It aims to build systems for such tasks by providing:

1. An architecture for task specific vision systems.
2. A mechanism for building systems in this architecture.

The architecture comprises frameworks with generic components for both low-level and high-level vision which can be specifically engineered with respect to accomplishing a given task.

The architecture is one in which task-specific knowledge is embedded within a directed acyclic graph (or lattice) structure. This knowledge base is primarily responsible for applying high-level reasoning and constructing the scene or image interpretation. The knowledge base is served by a low-level vision system (the Image Query Language subsystem) which provides a structured environment for image acquisition, processing and segmentation. This system supplies the knowledge base with segment information for further processing.

See<sup>++</sup> systems are built through the use of supervised machine learning which is responsible for acquiring knowledge. The use of machine learning permits the construction of a knowledge base which intimately reflects the needs of the particular task for which the system has been built.

See<sup>++</sup> is not designed to be a general vision system. However, it is intended to solve a spectrum of machine vision problems, adapting a generic framework to particular applications.

One particular scenario, which has been particularly influential in the design and construction of See<sup>++</sup> is teleoperation in a low bandwidth environment.

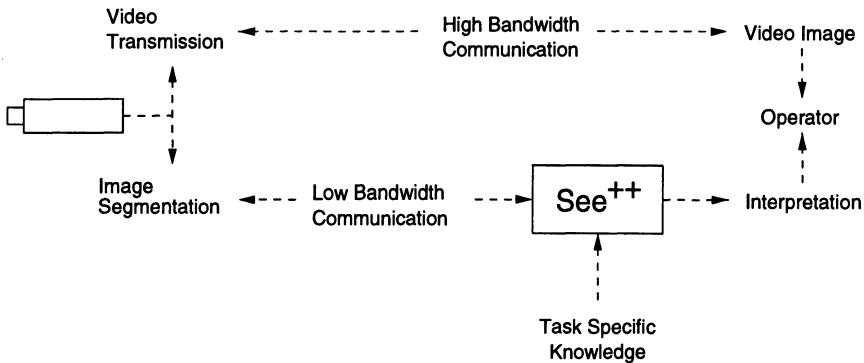


Figure 5.1: Comparison of conventional and knowledge-based communication in teleoperation

### 5.1.1 Teleoperation Scenario

Teleoperation, the control of a remote vehicle, is usually accomplished by transmitting real time video from the vehicle to an operator. The operator can then control the vehicle and make decisions about other actions to be taken. However, in many environments in which teleoperation is desirable (those which are inaccessible or hazardous) the very nature of the environment prohibits the use of video bandwidth transmission. This problem is usually solved by having the vehicle tow an umbilical cable which carries the video feed. This approach can often be unsatisfactory due to the concomitant problems of reduced manoeuvrability, cable snags and limited range.

The See<sup>++</sup> teleoperation system is designed to operate in an environment in which video transmission is infeasible but lower bandwidth transmission is available. The aim in this environment, then, is to place sufficient computational resources on the vehicle such that it is able to annotate the environment in adequate detail that a symbolic representation can be remotely reconstructed for use by the operator.

This is achieved by using the See<sup>++</sup> system (see Figure 5.1) to create a symbolic representation of the environment containing enough information for the operator to navigate and perform other useful functions.

## 5.2 See<sup>++</sup> Theory of Vision

### 5.2.1 What is Knowledge?

*“Data is not information, information is not knowledge and knowledge is not wisdom”*

Whilst engineering wisdom in computer systems remains an intractable problem; data, information and knowledge are certainly within the scope

of the See<sup>++</sup> project and the dictum quoted above, goes some way to illustrating the view taken. To place these terms in the context of machine vision:

**Data** is the raw sensor material absorbed by the system, in the form of grabbed rgb images

**Information** is value added data (i.e. data that has been processed and typically reduced in some way). As in many systems, See<sup>++</sup> processes data in two stages:

First, sensor data is processed in a very simple way to extract the key information. The aim of this step is to reduce the size of the data set without discarding useful information.

Second, this information is processed by a more complex system to obtain the output required.

**Knowledge** is the engine which performs this processing; in particular, the second step identified above.

Here, the term *information* should be differentiated from its use in information theory, in which it corresponds to the entropy of the distribution from which the data comes. It is worth observing at this point that when computer programs process data (such as that acquired from sensors), information, in the technical sense, can only be lost. Indeed, it is typically the task of such programs to substantially reduce the input data to a much smaller amount of information. For example, only a few hundred bytes of information may be all that is required from an input image originally consuming over a megabyte.

### 5.2.2 Vision Paradigms

See<sup>++</sup> follows the view, often referred to as purposive, or goal-directed vision [6, 70, 147], that vision systems should be specifically tailored for a given task. It is not intended as a general vision system which can recognise a large number of objects across a large number of environments. Rather, it is intended to be a generic system which can adapt to solve a spectrum of problems.

The stance taken is that vision should be environment specific. There is no need for any particular vision system to operate well in all environments when the domain of operation is constrained. Clearly, however, not all environments have the same characteristics and consequently, See<sup>++</sup> is designed to adapt and become specialised to the particular environment in which it will operate. Thus the system can take advantage of any local constraints that may be present in the given domain.

See<sup>++</sup> uses a common set of components to tackle a large range of vision problems, requiring object recognition capabilities. These components are assembled into an adaptive system which is tailored to specific tasks through the use of supervised machine learning. Thus See<sup>++</sup> is a *generic* system, rather than a *general* one.

In a related paradigm, that of active vision [5, 17, 286, 328], the view is taken that vision systems should exercise control over the image acquisition process. Whilst direct control over imaging is not always possible or necessarily desirable (for example in a fixed camera surveillance system), this idea can be employed in a more general form. In See<sup>++</sup>, dynamic control is exercised over all low-level processes, possibly including image acquisition as well as image processing aspects of the system. This control is dynamic in the sense that it is determined as a function of partial high level interpretation. Thus See<sup>++</sup> incorporates high-level feedback which is obtained by exploiting the object recognition capabilities of the system. Furthermore, the control system is also adaptive and adjusts to optimise task performance. Again, this is achieved by use of machine learning.

### **5.2.3 Low-level Vision and Feedback**

The See<sup>++</sup> system employs a recognition-by-parts approach [26] , that is, one in which discrete tokens, representing features such as regions or edges, are extracted from the image for further processing.

One of the principal reasons that this approach is attractive, is that pixels are expensive to process. In a typical See<sup>++</sup> image, there are approximately 250,000 pixels. Consequently, any process that must be applied to each pixel is going to be computationally expensive unless the number of operations per pixel is small.

By using a two stage process in which parts are extracted from the image using a simple segmentation scheme and are then interpreted using more complex processing, it is possible to keep processing costs to a minimum. In particular, this occurs because the size of the data set processed by the complex system is greatly reduced from the original image. Many object recognition systems [198] (See<sup>++</sup> included) adopt this approach of having separate high-level and low -level systems.

Whilst the major research effort in See<sup>++</sup> is concerned with engineering high-level knowledge, there are several important considerations in low-level vision.

In general, segmentation schemes are not very robust, and it has been argued [156] that poor segmentation is the reason there are comparatively few successful vision systems. There is a substantial body of work on high level vision that relies on synthetic low-level information [176, 177] and it must be conjectured that, in part, this is due to poor performance of existing low-level systems in terms of high level requirements.

Most attempts to improve the performance of low-level systems can be broadly characterised as open loop. That is, they contain heuristic data driven rules which effectively model the segmentation problems that occur in a given domain. The disadvantage of schemes like this is that the processing cost per pixel rises, often quite substantially, causing a large computational overhead.

See<sup>++</sup> attempts to solve segmentation problems in a closed loop fashion, using high-level task-specific knowledge to recognise segmentation failures and to issue corrective resegmentation requests. The advantage of this type of approach is that, initially, segmentation can be performed very cheaply, using crude techniques that work only most of the time. Where these techniques fail, parameters can be modified or other, more expensive, techniques can be applied. Because the expensive processes are only applied to the regions which failed under crude initial segmentation, the total computational cost can be reduced by comparison to having applied them initially over the whole image.

This kind of feedback approach to improving low-level performance requires that high-level knowledge be able to identify and correct low-level performance. This is achieved in See<sup>++</sup> by exploiting the object recognition capabilities of the system and adding new object categories which represent the various modes of segmentation failure which we wish to detect. Detection of these new categories triggers rules which act to correct the segmentation failure.

Many segmentation schemes are characterised by having a number of parameters which can be used to tune segmentation. Often this can be regarded as a weakness as the best choice of parameter values may not be immediately apparent. The feedback mechanism in See<sup>++</sup> turns this weakness into a strength, as it allows the system to adjust values for resegmentation and adaptively home in on the optimal parameter set.

### **5.2.4 Phenomena and Noumena**

In Kantian philosophy [164], the terms *phenomenon* and *noumenon* are used to distinguish between that which is perceived (*phenomenon*) and that which is in itself (*noumenon*). These ideas are used in the context of machine perception to distinguish between what can be measured in the sensor data by the computer and what is inferred from those measurements.

See<sup>++</sup> has a phenomenal theory of vision; all reasoning takes place in terms of perceived phenomena in the image, rather than with respect to the noumena which they infer. Since all machine perception systems are necessarily constrained to perform their reasoning with the data acquired, it could be argued that all systems are phenomenal in nature. By maintaining the description of the process in phenomenal terms, however, it is possible to maintain a clearer picture of the process and this perspective influences

a number of design choices.

There are a plethora of possible segmentations that can be applied to an image. Each segmentation can be seen as specifying an image phenomenon (for example a region characterised in a particular way) and the job of the segmenter is to find the instances of this phenomenon in the image. Each segmentation typically produces a non-overlapping set of image regions which cover some region of the image (possibly the whole image). However, if a number of segmentations are applied to an image, as a result of feedback, a given pixel is likely to be a member of more than one segment. When the interpretation is constructed, inconsistencies may result, with a given pixel labelled as belonging to more than one object. See<sup>++</sup> takes the view that low level vision is not the place to resolve these inconsistencies since both phenomena really do exist and that this conflict should be resolved at a higher level when the attributes and relationships of these segments are examined.

Furthermore, when an image query is executed and, for example, an image is under-segmented, it can be tempting to consider some of the segments as being incorrect and that the segmentation system should be fundamentally modified to generate the desired parts. From a phenomenal perspective, however, these parts *are* valid instances of the phenomenon specified in the original query. From this standpoint, it is more desirable to use these segments and their characteristics to direct the feedback process for appropriate resegmentation (i.e. to use these parts to evidence the need to use a different segmentation which will generate the parts needed for recognition).

Interpretation takes place by descending and instantiating a hierarchy of phenomena in the image. Only at the final stage are noumena inferred from the most complex phenomena. Thus the reasoning is phenomenal and phenomena evidence noumena.

### **5.2.5 Image as Database**

See<sup>++</sup> adopts the view of the image (or the world) as analogous to a database [17]. This analogy extends to the use of database terminology to describe image operations:

$$\begin{array}{lcl} \text{Query} & \iff & \text{Segmentation} \\ \text{Record} & \iff & \text{Segment} \\ \text{Field} & \iff & \text{Attribute} \end{array}$$

In this view, segmentation requests are queries submitted to the database and segments created by this operation are records in the database which satisfy the given query. Further, the attributes of the image segments (size, shape etc.) can be viewed as fields in each record.

This analogy is consistent with the phenomenal view of vision taken by See<sup>++</sup>. Clearly the segments returned *are* the records in the image database which satisfy the given query. Furthermore, since the image segments are the primitives in this paradigm, it is possible for pixels to belong to multiple image segments - there really is more than one image phenomenon which incorporates that pixel.

On first sight, this view appears to admit the possibility that a pixel can not only belong to more than one part, but consequently belong to more than one object. Whilst See<sup>++</sup> does not explicitly exclude this possibility, the system is designed so that this incompatibility can be implicitly reconciled in downstream processing

## **5.2.6 Knowledge Representation and Acquisition**

A key issue in the design of complex sensory processing systems is the engineering of knowledge. These systems need to store information about the environment and objects of interest, in such a manner that an operational recognition scheme can be enacted.

### **5.2.6.1 Procedural vs. Declarative**

A distinction is often made between procedural and declarative representations of knowledge, where a procedural representation is one which explicitly codes how the data is processed and a declarative representation contains explicit models of the data. It should be noted that a procedural representation carries implicit models of the data and a declarative representation implies a procedure via some operational engine.

See<sup>++</sup> lies towards the procedural end of the spectrum in that object models are not explicitly coded. Instead, discriminatory rules constitute the knowledge in a procedural manner. However, one of the aims of the representation system used is to bring object descriptions to the surface such that the procedural rules can be viewed in a way that permits the implicit object models to be readily comprehended.

### **5.2.6.2 Discrimination vs. Modelling**

An alternative way of presenting this view is that See<sup>++</sup> does not seek to match models onto perceived phenomena from the environment, rather the approach is discriminatory. In this approach, the system attempts to do the least work necessary to discriminate objects of interest from each other and from those items of no relevance. A result of this is that some objects may be very easy to identify in the environment of the task. For example in a road surveillance system it may be possible to identify a blob as a truck based purely on its size without having to extract any more detailed information.

Furthermore, See<sup>++</sup> does not attempt to fully label the images with which it is presented. Instead the aim is to identify those regions of importance to the task and classify them. As a result, See<sup>++</sup> does not have an exhaustive database of objects from the operational environment, only those of semantic importance to the task are included.

#### **5.2.6.3 Top-down vs. Bottom-up**

In general, the approach in See<sup>++</sup> is to integrate top-down and bottom-up processes [286, 365]. This is particularly evident in the use of knowledge driven resegmentation where the low-level processes are controlled as a function of high-level interpretation (by contrast to purely data driven processes [75]).

It must be noted that top-down processes typically correspond to use of declarative knowledge, while bottom-up processes are procedural. It is often possible, however, to pre-compile top-down processes in an efficient way so that a declarative representation is converted to an (often faster) procedural one. In See<sup>++</sup>, machine learning is effectively responsible for carrying out this process.

#### **5.2.6.4 2D vs. 3D Object Recognition**

See<sup>++</sup> rules are coded in terms of features extracted from image parts. Currently, no attempt is made to place these parts in a three-dimensional context and consequently, the features and rules relate to the two-dimensional appearance of the parts. Clearly, however, in many environments objects will be seen from multiple viewpoints. Since no attempt is made to achieve explicit 3D invariance, this must be accomplished implicitly within the discrimination rules.

A common technique for achieving this is to use a viewsphere, where a number of views of each object are encoded within the system. See<sup>++</sup> does not use viewspheres, but can implicitly code this type of information. If the discrimination rules fail to identify a given object from a new view, then new rules are created to incorporate this new version of the pattern.

#### **5.2.6.5 Machine Learning**

The use of machine learning is central to the See<sup>++</sup> philosophy, for it is the means of acquiring task specific knowledge. Alternatives to this approach include the use of human constructed knowledge, either directly or indirectly (for example by processing a series of CAD models from a database [35, 110, 226] ).

In See<sup>++</sup>, image parts and their relationships are systematically queried to form the interpretation. Thus the system can be likened to a game of twenty questions, in which the aim is to gain as much information as

possible about the scene from each question asked. This principle has been used by a number of data processing systems [273, 354]. However, there are many considerations arising from the need to operate within a relational domain, that is one in which the data comprises attributed components with attributed relationships. This is discussed further in section 5.6.

An important aspect of See<sup>++</sup> is that knowledge acquisition is automated and optimised through the use of supervised incremental machine learning. The user interface tools enable the user to train the system rapidly to identify the specified objects of interest.

## 5.3 System Architecture

There are two frames of reference in which to discuss the architecture of See<sup>++</sup>; the flow of information and the software architecture in an object oriented context.

### 5.3.1 Information Flow

Because See<sup>++</sup> employs a recognition by parts approach to object recognition, there are two major processing systems; image segmentation and part processing. These systems embody the low-level and high-level processing requirements of the system respectively and are jointly responsible for constructing an interpretation for each image. There is also a machine learning system which interacts with the user and is used to manage and construct the other systems. These systems are shown in Figure 5.2.

In this section, these components will be described briefly and the progression of data as it is processed through the system will be discussed.

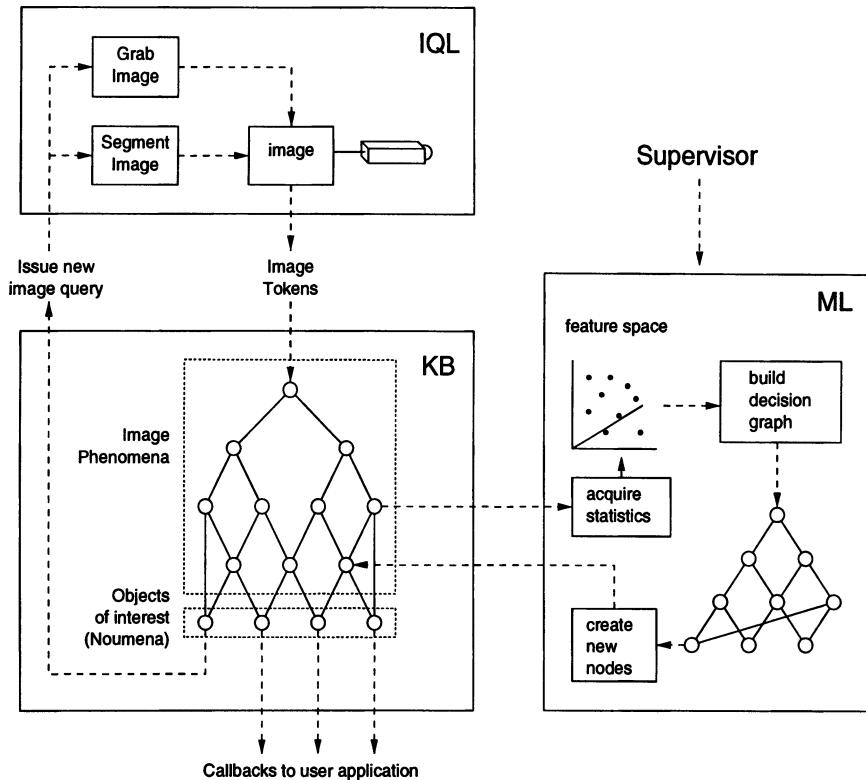
#### 5.3.1.1 Image Query Language (IQL) Subsystem

The IQL subsystem (discussed in more detail in section 5.4) is responsible for all low level vision operations. These include image segmentations which specify the segmentation scheme that should be applied to the image along with any parameters that may be required.

These queries are serviced sequentially and may generate a number of image segments, or tokens. For each token, a set of attributes are computed (the specific attributes depend on the exact type of the token) and communicated to the part processing system, or knowledge base.

#### 5.3.1.2 Knowledge Base (KB) Subsystem

The knowledge base (discussed in more detail in section 5.5) is responsible for assembling an interpretation of the image from the parts generated in

Figure 5.2: Flow of data in See<sup>++</sup>

the IQL subsystem. This is achieved by using the parts' attributes and relationships. The architecture of the knowledge base is a lattice (or directed acyclic graph) in which the nodes represent phenomena. Each node maintains a list of the tokens which are detected instances of that phenomenon in the current image. Attached to each node are a number of processing units which act on each token according to internally specified methods, propagating the tokens through the lattice. This process instantiates a series of phenomena which, in the final analysis, evidence and instantiate the noumena, or real world objects, in which the user application is interested.

### 5.3.1.3 Feedback

In addition to the processes described above, feedback is utilised from the knowledge base to the IQL subsystem. This permits the use of highly non-linear and non-stationary processing methods at the pixel level which can be tuned to task requirements.

There are substantial advantages to employing a feedback mechanism in this manner, providing greater stability and robustness for comparatively little cost.

By implementing feedback directly from the knowledge base to the IQL subsystem, it is possible to maintain all the knowledge relevant to the task in a single data structure, thus facilitating maintenance, storage and retrieval. The process of knowledge acquisition can also be unified and similar optimisation strategies applied.

#### **5.3.1.4 Machine Learning (ML) Subsystem**

The machine learning subsystem (discussed in more detail in section 5.6) is responsible for acquisition of task specific knowledge and for constructing the knowledge base.

In See<sup>++</sup>, learning takes place in the knowledge base at the interface between phenomena and noumena. The machine learning subsystem acquires statistical information from the knowledge base, combined with labelled parts from the supervisor. Entropy driven techniques are then used to update the leaf nodes of phenomena to refine the evidence that they give for the various noumena.

#### **5.3.2 Software Architecture**

See<sup>++</sup> is designed using an object oriented approach and is coded in C++. The object oriented approach impacts on several design choices due to the comparative ease with which many things can be achieved.

Principally, object oriented design serves as a unifying methodology. Due to the capabilities of object oriented languages, like C++, constructing a single knowledge base which supports multiple modes of reasoning is substantially simplified. Similarly, the image query language can support multiple segmentation techniques and token types.

Object oriented design has three important facets all of which are relevant to the construction of the See<sup>++</sup> system and are summarised here:

**Classes** in OO programming represent types of objects or concepts of which the program may contain one or more instances. Each instance provides services as specified in the class definition and is responsible for maintaining its own internal data integrity.

**Derivation** permits new classes to be based on pre-existing classes, thus allowing the concept expressed in that class to be further refined. The new (derived) class behaves in all the ways expressed in the original class but may also provide additional services.

**Polymorphism** permits the method used to provide a service to be dynamically determined at run time. The method used depends on

the exact type of the instance providing the service. This allows the construction of heterogeneous collections of objects each of which provides semantically the same service, but with individual implementations.

See<sup>++</sup> contains more than 150 classes, the majority of which provide basic infrastructure management. Figure 5.3 shows only the major classes. There are three significant class hierarchies in See<sup>++</sup> (shown in bold in the diagram). These are:

**Knowledge Methods** which implement the actions to be taken when a phenomenon is instantiated in the image. They include all the modes of reasoning incorporated in the system, together with additional actions to facilitate machine learning and user interaction.

**Image Queries** which perform all the low level image operations including image acquisition, segmentation and utilities to assist in managing the user interface. Image segmentations form a substantial sub-tree

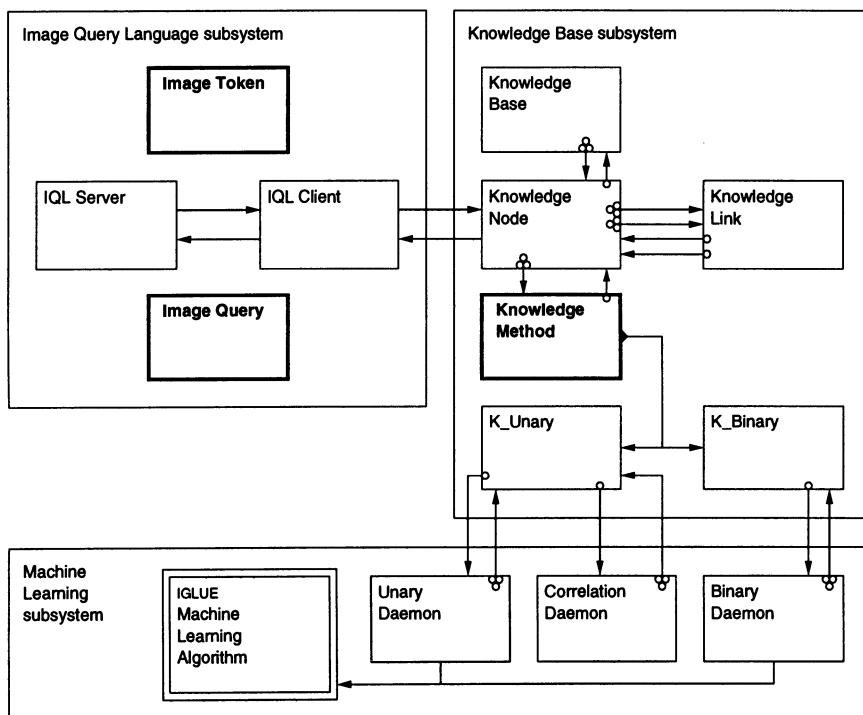


Figure 5.3: Software architecture of See<sup>++</sup>

in this hierarchy, representing different ways in which the image can be segmented.

**Image Tokens** which correspond to the types of information extracted from the image via the image query language subsystem. They include image regions with attributes for processing, chain code information etc.

See<sup>++</sup> comprises three main subsystems shown as the large boxes in figure 5.3 . This coarse grained breakdown of the system allows high-level responsibilities to be allocated, while keeping subsystem interaction to a minimum. There are implementation classes associated with each of the subsystems.

#### **5.3.2.1 Image Query Language**

The IQL subsystem contains classes for image acquisition and manipulation, together with high level classes to implement the client server architecture. The IQL subsystem communicates with the knowledge base, receiving image queries and returning image tokens.

#### **5.3.2.2 Knowledge Base**

The knowledge base contains a skeleton lattice structure of knowledge nodes to which are attached knowledge methods which apply the processing to the tokens received from the IQL subsystem.

#### **5.3.2.3 Machine Learning**

The machine learning subsystem contains generic classes for implementing the IGLUE algorithm, together with classes that store statistical information in the knowledge base and update it, using the output from IGLUE .

#### **5.3.2.4 Embedding See<sup>++</sup> Within Systems**

See<sup>++</sup> is incorporated into systems by allowing user methods to be attached to knowledge nodes. Typically, the user application creates the objects of interest, which then reside in the knowledge base as noumena. The user application can then attach special methods to these nodes so that detection of the given object causes these methods to be activated in a similar manner to callbacks used in X windows programming.

### **5.4 Image Query Language**

At the lowest layer in See<sup>++</sup> is the Image Query Language (IQL) subsystem. This subsystem performs all low level image operations, including image acquisition and segmentation.

### **5.4.1 Justification**

A key design requirement of See<sup>++</sup> is that it be able to operate in a low bandwidth environment. That is, where the communications between the user and the point of image acquisition are substantially below video rate. The main implication of this requirement is that initial image processing, at least, must be carried out local to the point of image acquisition, or camera. In See<sup>++</sup>, the higher levels of processing are carried out local to the user. This design choice was made to allow complex interaction between the operator and high level processing (as may be required in some systems) and to keep the remote processing system as algorithmically simple as possible to increase robustness. Given these design choices, it is necessary to communicate image processing requirements and the results of these, to and from the remote system.

Furthermore, See<sup>++</sup> extracts information from images in an iterative fashion by contrast to many machine vision schemes which attempt to extract all the required information from the image in a single pass. Consequently, a given image may be subject to a number of queries, all of which use different segmentation schemes or parameter sets.

Thus in See<sup>++</sup> there is a need for a scheme in which these image segmentations as well as other image operations can be carried out and communicated. This effectively implies a structured environment, or language.

This language, must be able to express all low level visual requirements, and since See<sup>++</sup> is designed to have application to a number of problems, which use differing low-level tools, there is the need for a range of low level processes to be encompassed within the language. Moreover, new tasks may require additional low-level processes, thus the language must be open to extension, incorporating new image operations as necessary.

### **5.4.2 The IQL Query Class Hierarchy**

This language is expressed as a C++ class hierarchy. At the top of this, is the abstract base class IQL Query. This class contains abstract methods for:

- Executing the query
- Handling tokens generated as a result of the query
- Transmitting itself across the client-server connection
- Reconstructing itself at the server

There are currently ten derived classes in this hierarchy in use in the See<sup>++</sup> system. These range in operation from segmentation schemes to image management (grabbing, loading, saving) and shape extraction (chain codes, run length encoding).

### **5.4.3 IQL Client and Server**

The IQL system has an asynchronous client-server architecture in which the local IQL client can connect to a remote IQL server. Once the connection is established, the user application can hand queries to the IQL client which transmits them to the IQL Server, retaining its own copy. This process is asynchronous and the user application can continue processing while the IQL server is executing the query.

Executing the query results in image acquisition, segmentation, or other image processes to occur. The results of this action, such as segment information, or a chain code, are then communicated back from the IQL server to the IQL client and hence to the user application. Thus the system communicates both queries and the results of the queries between disparate locations.

Different queries cause differing kinds of information to be sent back to the IQL client. Consequently, there is also a need for a common method for communicating this information.

### **5.4.4 IQL Tokens**

Similar to the hierarchy of queries, there is also a hierarchy of tokens. Each query executed by the IQL Server may cause a number of IQL Tokens to be sent back to the IQL Client. The client copy of the query is then passed each returning token. In the case of image segments, these are instantiated in the top of the knowledge base for further processing.

### **5.4.5 IQL in Action**

An example of typical operation is shown in Figure 5.4.

1. A Query is sent to the Client
2. The Client transmits a copy of the Query to the Server over a socket
3. The Server executes its copy of the Query
4. The Query generates a number of Tokens which are returned to the Server
5. The Server transmits them to the Client
6. The Client hands them to a local copy of the Query
7. The Query then instantiates the appropriate node in the knowledge base with each of the Tokens

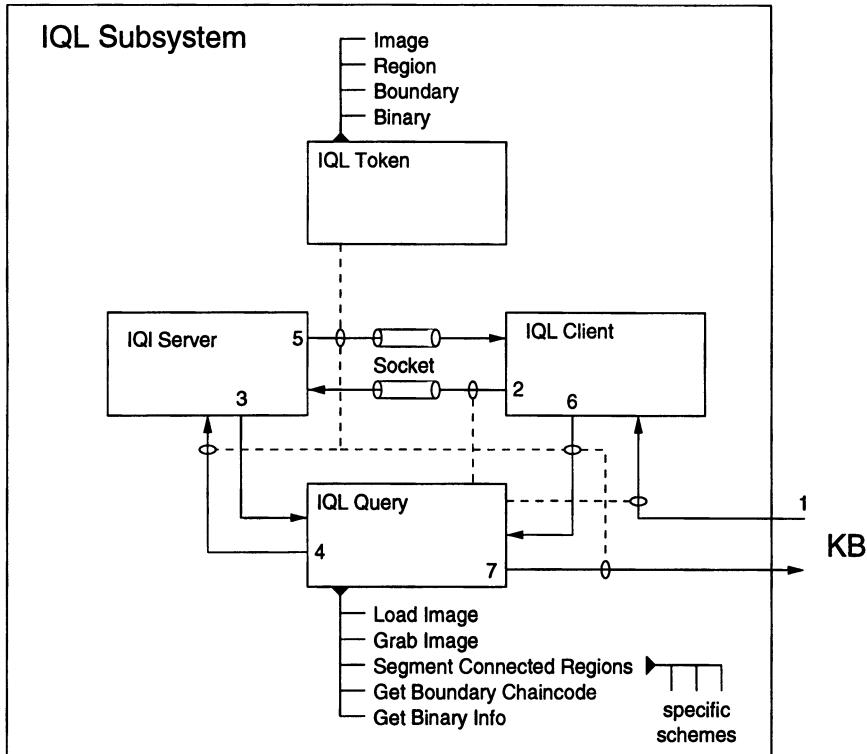


Figure 5.4: IQL subsystem

#### 5.4.6 An Example - Colour Space Segmentation

A number of image segmentation schemes have been used in See<sup>++</sup>, including RGB clustering and colour variance clustering. One of the most useful and widely applicable is edge based segmentation, similar to the Canny edge detector. A specific example of this type of segmentation is colour space segmentation.

The colour segmentation class is derived from an edge based segmentation class which, in turn, is derived from a connected components class. This relationship and the processes which take place within each layer of the class hierarchy are represented in Figure 5.5.

Colour space segmentation requires four parameters: Resolution, Colour vector, Scale ( $\sigma$ ) and Threshold. The current image is stored at several resolutions, each being a factor of two smaller than the previous. Parametrically specifying the resolution allows the use of multiple scales in resegmentation.

Initial segmentation usually occurs at the lowest resolution, to ensure

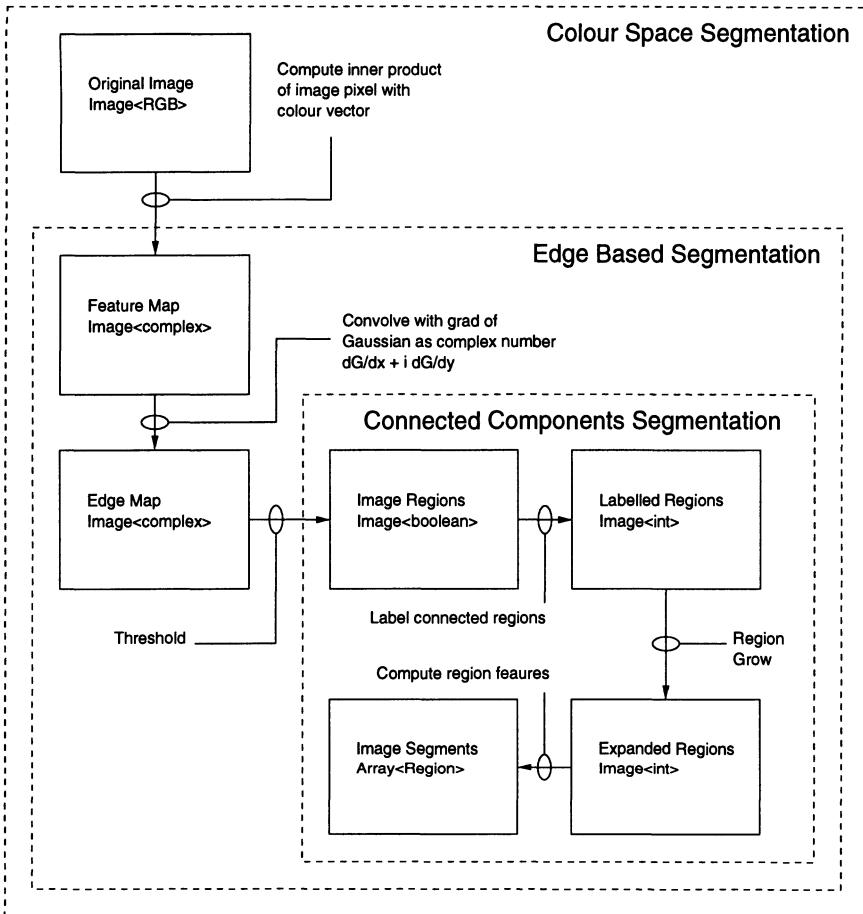


Figure 5.5: Colour space segmentation showing classes

speed of operation. For most image parts this is sufficient, but some image regions may need to be re-examined in more detail.

The resolution parameter selects the image of appropriate resolution. The inner (dot) product of each pixel in this image is then computed with the colour vector parameter. This vector is a normalised unit vector but not necessarily in the positive orthant, allowing colour opponency to be used. This results in a feature map (Figure 5.6(a)) for the given colour vector.

$$F(x, y) = \bar{I}(x, y) \cdot \bar{C}$$

From this point segmentation is handled by the immediate base class for colour resegmentations, edge based segmentation. This class accepts a

scalar (single valued) feature map (in this case supplied by the colour space segmentation sub class) in the form of a complex image. In this image, the real component holds the scalar quantity of the feature map and the imaginary component is zero. The image is convolved with the grad of an isotropic two dimensional Gaussian function with standard deviation  $\sigma$ :

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

expressed as a complex number:  $\nabla G_\sigma = \frac{\partial G_\sigma}{\partial x} + i \frac{\partial G_\sigma}{\partial y}$ . This gives the edge map shown in Figure 5.6(b):

$$E(x, y) = F(x, y) \otimes \nabla G_\sigma(x, y)$$

This computation is performed in the frequency domain and only requires two Fourier transforms to compute (the transform of  $\nabla G$  being directly computable). The result is a complex image showing the edges of the feature map when blurred over a scale of  $\sigma$  pixels. These edges are encoded as complex numbers, the magnitude of which gives the edge strength and the phase the orientation.

The edge map is thresholded (on the magnitude) to obtain a binary image in which the edge pixels are labelled false, and the non edge pixels labelled true. This binary image is then passed to the base class of edge segmentation, connected components segmentation.

Connected components segmentation uses a standard two pass technique to label connected regions [318], giving 5.6(c). These regions are grown to absorb boundary pixels using an adaptation of the chamfer (3,4) distance transform [36], producing 5.6(d). Finally a set of statistics is computed for each segment.

The advantage of constructing a layered class hierarchy to perform this segmentation is that other similar schemes can readily be attached to the hierarchy. There are many other feature maps that can be used to perform segmentation, such as the total colour variance feature (useful for simple segmentation of textured regions). Alternatives to edge based region computation can also be used, such as clustering. These segmentation schemes can pass their output directly to the connected components base class.

## 5.5 Knowledge Base

The Knowledge Base (KB) forms the core of the See<sup>++</sup> system. It sends queries to the IQL system and receives image tokens, typically image segments back. These tokens are processed by the KB and they are used to evidence (or otherwise) the objects that are of relevance to the task. The user application is then informed of each instance of each object in which it

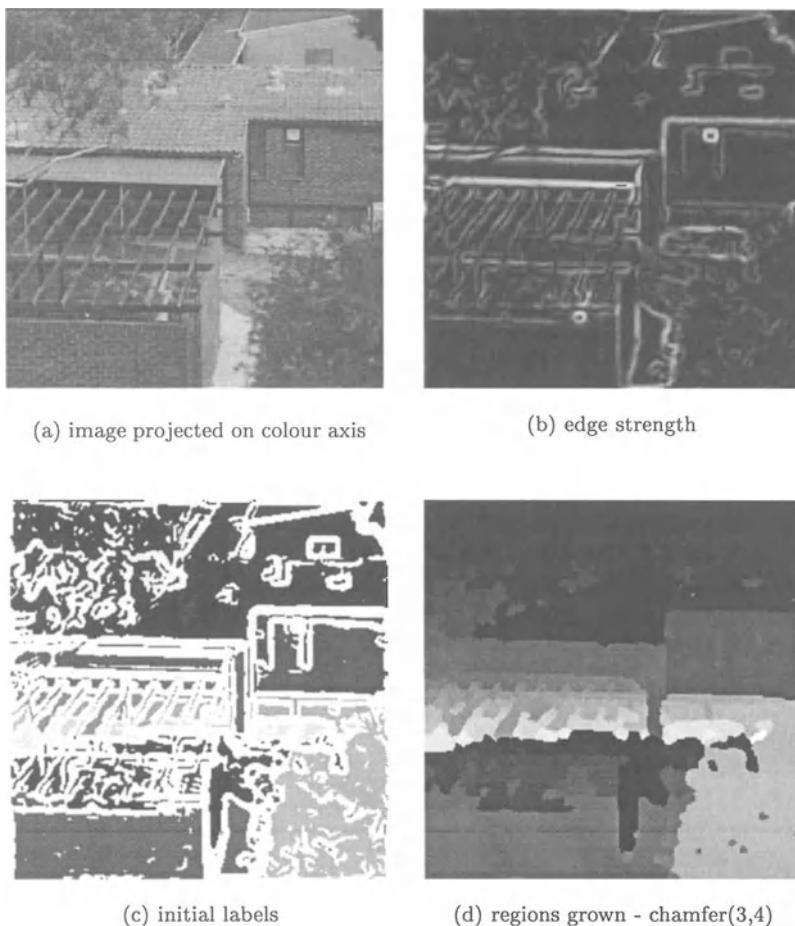


Figure 5.6: The segmentation process

is interested. The main job of the KB is to process these tokens to synthesise an intelligent interpretation of the image in a manner which is suitable for the current task. The KB also manages feedback to the IQL system and provides services to the user application.

### 5.5.1 Justification

The central issues in See<sup>++</sup> are the representation and acquisition of knowledge. The representation, operation and management of knowledge are the domain of the knowledge base, whilst knowledge acquisition is handled by the machine learning system described in section 5.6.

In a complex field like object recognition, many types of reasoning are required to reasonable performance. Furthermore, these modes of reasoning necessarily interact with each other, causing maintenance of knowledge to become a substantial burden.

In order to simplify maintenance, the See<sup>++</sup> knowledge base is designed so that all types of reasoning, or knowledge, have a common structure (and are derived from a common base class). Additionally, all pieces of knowledge are stored within a single lattice or directed acyclic graph structure.

There are considerable advantages to having a unified structure for knowledge.

- Knowledge maintenance operations (including loading and saving) are simplified.
- Internal representation of the knowledge is also unified, making the structure and behaviour of a given knowledge base more comprehensible through graphical, or textual presentation.
- The same machine learning techniques can be used to acquire pieces of knowledge relating to different modes of reasoning.

### **5.5.2 Interactions**

The knowledge base resides in the centre of the See<sup>++</sup> system and consequently has to interact with all the other subsystems. The principal responsibility of the knowledge base is to process image tokens passed from the IQL subsystem and perform object recognition. This entails several subsidiary responsibilities:

- Interaction with the IQL subsystem
  - Sending all image queries
  - Implementing knowledge driven resegmentation
  - Accepting tokens for processing
- Interaction with the Machine Learning subsystem
  - Provision of support services for statistical analysis and construction of new knowledge
- Interaction with the user
  - Storage and display of all task specific knowledge
  - Support for user queries
- Interaction with the user application
  - sending notification of all objects of interest to the user application

### 5.5.3 The Language of Knowledge

The See<sup>++</sup> system represents phenomena as nodes in a lattice (see Figure 5.7). Each node stores a list of tokens which are detected instances of that phenomenon. Also attached to these nodes are knowledge methods which may perform any one of a number of functions. If a token  $s$  is an instance of the knowledge node  $K_i$ , write  $s \in K_i$ . Each method attached to  $K_i$  is then called for each of the tokens,  $s$ , in the list. These methods may:

- Issue an image query
- Classify a token as an object of interest
- Store statistics for the token
- Perform some user process
- Propagate the token through the knowledge base hierarchy

This last item includes some of the most important methods which identify and instantiate more complex phenomena. This is essentially achieved in two ways:

1. **Unary Queries** which interrogate the properties of the instance of the current phenomenon

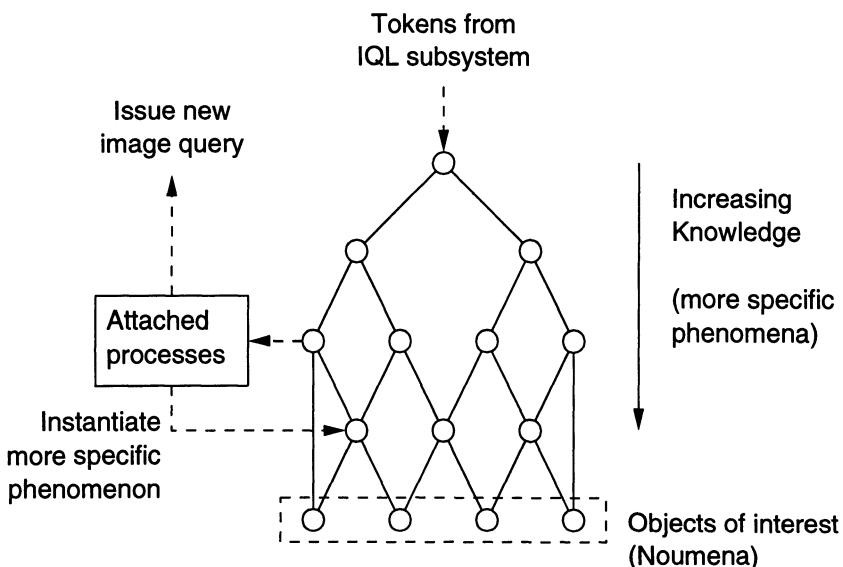


Figure 5.7: Knowledge base schematic

2. **Binary Queries** which interrogate the relationship between the instance of the current phenomenon and any instances of another phenomenon

These two types of queries are used to refine what is known about a token, and implicitly code new phenomena. If  $K_i, K_j$  are some nodes in the knowledge base, then a new node,  $K_{new}$  can be formed by these production rules:

$$s \in K_i \wedge s.f_k > \alpha \Rightarrow s \in K_{new} \text{ unary split (5.1)}$$

$$s \in K_i \vee s \in K_j \Rightarrow s \in K_{new} \text{ merge (5.2)}$$

$$s \in K_i \wedge \exists r \in K_j \wedge B(s,r).f_k > \alpha \Rightarrow s \in K_{new} \text{ binary split(5.3)}$$

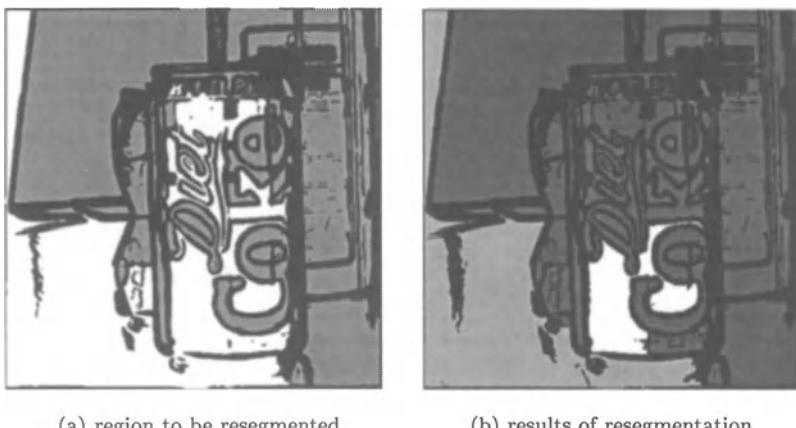
where  $s.f_k$  is the  $k$ th feature of segment  $s$  and  $B(s,r).f_k$  is the  $k$ th feature of the relationship between  $s$  and  $r$ .

Rule  $K_{new}$  is a child of  $K_i$  and possibly  $K_j$  (rules 5.2, 5.3) and represents a more specific phenomenon than  $K_i$  or  $K_j$ , (excepting rule 5.2). Generally, this means that as the hierarchy is descended through children, the phenomena represented become more specific until at the lowest layers, they are used to evidence the noumena. At the top of the knowledge base is a node  $K_0$  of which all segments are instances, all other nodes deriving from it.

These production rules define a space of representable phenomena, which is analogous to a formal language of phenomena with an associated grammar. This space is sufficiently rich to allow many object recognition tasks to be tackled but is incomplete in the sense that there exist pathological cases of patterns that are indistinguishable yet different. A rigorous solution to this problem, where the data can be represented as an attributed graph (in which both vertices and edges have labels), is essentially a solution to the subgraph isomorphism problem [167]. Whilst this problem is not known to be NP-complete there are currently no polynomial solutions and furthermore the cases presented in a machine vision scenario are often likely to be close to the decision boundary of the problem.

### 5.5.4 Knowledge Based Resegmentation

One of the tasks of the knowledge base is to control feedback to resegmentation. This is achieved by capitalising on its object recognition capabilities. The user classes of interest are supplemented by resegmentation classes which are identified using exactly the same mechanism. When this occurs, as in Figure 5.8(a), the instantiated resegmentation node issues a new query with parameters computed from the attributes of the instantiating segment. This new image query resegments the instantiating region resulting in new parts (Figure 5.8(b)) which are then processed by the knowledge base.



(a) region to be resegmented

(b) results of resegmentation

Figure 5.8: The resegmentation process

Because the object recognition system in the knowledge base is used to detect segmentation failures, the characteristics that define them can vary from task to task and form part of the body of task specific knowledge acquired using supervised machine learning.

## 5.6 Machine Learning

In See<sup>++</sup> machine learning is used to build task specific knowledge, both for object recognition and, by extension, feedback for low-level vision. This process builds the knowledge base at the interface between phenomena and noumena.

The machine learning subsystem does this by interacting with both the knowledge base and the user, who takes the role of the supervisor in the learning process. This subsystem takes user labelled part data and accumulates statistical information which is stored in the knowledge base. These statistics are then used to direct the search in the space of possible rules.

### 5.6.1 Justification

Together with knowledge representation, acquisition is an important concern in complex tasks such as object recognition. Unless well defined object models already exist, such as in a CAD database, knowledge to represent them must be obtained in some manner. This can be an intensive task and often the heuristic rules generated by an expert can be less than optimal

with respect to accomplishing the given task.

By using machine learning it is possible to partially automate knowledge acquisition by rapidly searching a large space of possible knowledge and encoding the best according to a given criterion. In supervised machine learning, the user labels data to guide the search. This process requires a lower level of expertise than hand coding of knowledge and can, in general, be accomplished more quickly. This permits faster acquisition of optimised knowledge which is finely tuned to the task. The knowledge is optimal in the weak but well defined sense that it maximises a given criterion across the space of possible knowledge that falls within the search strategy.

Clearly the efficacy of this approach depends greatly on the space of knowledge, the search strategy and the optimisation criterion.

In See<sup>++</sup>, discriminatory rules are used to perform object recognition and it is desirable that each rule separates the classes as much as possible without over-fitting. This kind of knowledge is well suited to the application of machine learning and uses correlation and entropy based techniques to select rules.

### **5.6.2 Machine Learning in Object Recognition**

There are many powerful machine learning tools for use in pattern recognition. However, most existing techniques cannot be directly applied to object recognition tasks because they are concerned with labelling or clustering multivariate data. Exemplars are viewed as being vectors in some (typically high dimensional) vector space and these techniques cluster the data in this space, or generate a classification mechanism with decision boundaries that cut the feature space into discrete regions.

The object recognition problem is characterised by patterns not only in the multi-dimensional feature space of image parts but also in the space of relationships between those parts. In this kind of problem, the data representation is more complex than can be fed directly to these simple multivariate clustering and classification schemes. Typically such data can be represented as a graph in which both the nodes and the edges reside in disjoint feature spaces. Existing multivariate learning tools are useful for working within these feature spaces but an extension is required to generate decision boundaries that incorporate relational information.

### **5.6.3 Entropy Based Techniques**

There are a number of algorithms for machine learning which use information entropy of classification to direct the search for decision boundaries [270, 354]. These systems seek to minimise the entropy measure:

$$H(X) = - \sum_{i=0}^c p_i \log_2(p_i)$$

where  $X$  is the random variable associated with the classification of an example,  $x$ , there are  $c$  classes and  $p_i$  is the probability that  $x$  is an instance of the  $i$ th class with  $\sum p_i = 1$ . Note that  $H(X) = 0$  when  $p_k = 1, p_i = 0, i \neq k$  and  $H(X)$  gives a measure of the uncertainty of classification for a number of examples. The aim of an entropy based classifier is to separate the examples into distinct groups such that the uncertainty within each group is minimised. This technique has been used most notably by Quinlan in his program C4.5 [273] which builds decision trees to classify examples.

C4.5 iteratively partitions the feature space of examples, at each stage choosing the partition which maximises the reduction in entropy of the examples. The code contains many bells and whistles (not described here) for handling things such as attributes of different types and examples with unknown attribute values.

The decision tree starts as a single leaf node with all the examples in it. At each stage, the entropy for a leaf is computed from the number of examples of each class,  $a_i$ , within it, with  $\sum a_i = n$ .

$$H(\text{Leaf}) = -\frac{\sum_{i=0}^c a_i \log_2(a_i)}{n} + \log_2(n)$$

All possible axis parallel partitions are considered (ie those of the form  $f_i > \alpha$  - is the  $i$ th feature greater than some  $\alpha$ ?). This rule,  $R$ , splits the examples in the leaf node into two sets, upper and lower. The entropy for each set is computed and an average is taken, weighted by the number of examples that fell into each set. This weighted average is the expected entropy of the examples in the leaf conditional upon knowing the outcome of the rule:

$$H(\text{Leaf} | R) = \frac{n_{\text{upper}} H(\text{upper}) + n_{\text{lower}} H(\text{lower})}{n}$$

The entropy reduction (information gain) of each partition is computed from the two sets of examples generated by the partition as the difference between the original entropy of the leaf and the expected entropy of the two partitions:

$$I(R) = H(\text{Leaf}) - H(\text{Leaf} | R)$$

In the case of binary partitions (where there are two sets generated), the information gain is between 0 and 1. When all partitions have been tested, the one which generated the most information is retained and the tree is grown so that the leaf is replaced by a decision node containing the best rule and two new leaves are created and the examples are distributed between them according to the rule.

C4.5 contains heuristic rules to prevent bad partitions and the tree building stage is followed by pruning which removes statistically invalid branches of the tree and reduces overfitting, which improves the classification performance on unseen data.

### 5.6.4 The IGLUE Algorithm

See<sup>++</sup> adapts this tree growing algorithm in a number of ways, providing incremental learning with bounded storage requirements using a modified entropy measure of decision graphs for stochastic functions.

Incremental variants of decision tree algorithms have been developed [342] which essentially store all the training data and adapt to incorporate new training data as it arrives. This constitutes a fairly weak notion of incremental learning since the storage space required can grow arbitrarily large as well as the time taken to process additional examples. The IGLUE algorithm is designed to operate with limited storage and time requirements. The class structure of the IGLUE system is shown in Figure 5.9.

Since there is no pruning stage when learning incrementally (learning is never necessarily complete), overfitting must be avoided whilst the tree is being built. This is achieved in IGLUE by using a modified entropy measure which incorporates uncertainty due to small sample sizes. Furthermore, partitioning only occurs when the information gained exceeds a given threshold value.

IGLUE also builds decision graphs [88, 234, 235] which is a generalisation of decision trees. This is achieved through the merging of leaf nodes during learning. This process increases the number of samples available in the merged node for further learning and allows small splinters of feature space to be reabsorbed, simplifying the behaviour of the system. This is particularly important when the output of the decision graph is used in a relational context.

The learning algorithm is designed to learn stochastic functions. That

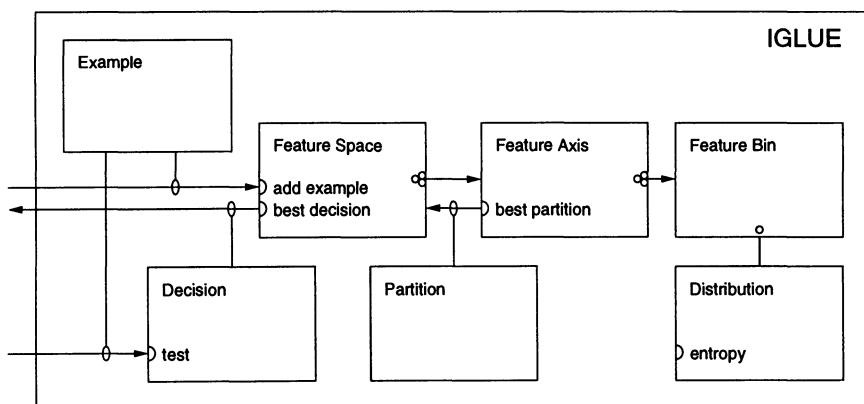


Figure 5.9: IGLUE class structure

is, those that do not necessarily have a definite classification at any point in feature space. This is often the case when dealing with relational data since the attributes of a node of the graph may not necessarily determine the classification of the node, instead they modify the classification probabilities of the node. Thus the output required is a probability distribution rather than a classification.

These factors affect the design of the algorithm in a number of ways. The requirement for bounded memory implies that the training examples cannot be simply stored, but the system must keep information from those examples to undergo further learning. The fact that the system is learning stochastic functions means that information must be lost when the training examples are represented in a restricted amount of space.

#### **5.6.4.1 Bounded Storage Requirements**

It is possible, however, to limit the amount of information discarded in this data reduction process. In See<sup>++</sup> this is done by binning the samples along each feature axis (see Algorithm 4). By doing this, See<sup>++</sup> restricts the points at which the learning algorithm can partition the data to the bin boundaries. This will result in sub-optimal partitioning, but the amount of information lost in the sub-optimality is bounded above by  $-(\alpha \log_2(\alpha) + (1 - \alpha) \log_2(1 - \alpha))$ , where  $\alpha$  is the fraction of samples falling between the optimal partition and the nearest bin boundary.

This can be proved as follows, let  
 $L$  be the random variable which is the class of a given example,  
 $R$  be the outcome of the optimal rule,  
 $B$  be the outcome of rule at the nearest bin boundary and  
 $X$  be  $O$  xor  $B$

---

#### **Algorithm 4 IGLUE Add example**

---

```
Add-Example(Example ex)
L = Find-Leaf(ex)
for i = 1 to c do
    B=L.axis(i).Find-Bin(ex.feature(i))
    B.Add-Example(ex.class)
    if B.num-samples >  $\alpha \times L.num-samples$  then
        Split B
        Merge small bins on L.axis(i)
    end if
end for
```

---

$$\begin{aligned} I(R) - I(B) &= H(L) - H(L|R) - (H(L) - H(L|B)) \\ &= H(L|R) - H(L|B) \end{aligned}$$

$$\begin{aligned} H(L|B) &\leq H(L, R|B) \\ &= H(L|R, B) + H(R|B) \\ &\leq H(L|R) + H(R|B) \\ &= H(L|R) + H(X, R|B) - H(X|R, B) \\ &= H(L|R) + H(X, R|B) - 0 \\ &= H(L|R) + H(R|X, B) + H(X|B) \\ &= H(L|R) + 0 + H(X|B) \\ &\leq H(L|R) + H(X) \end{aligned}$$

$$H(L|B) - H(L|R) \leq H(X)$$

$$I(R) - I(B) \leq -(\alpha \log_2(\alpha) + (1 - \alpha) \log_2(1 - \alpha))$$

Thus the information loss is dependent on  $\alpha$  and can be limited by choosing  $\alpha$  small enough. The bin must then contain no more than  $2\alpha$  of the examples. IGLUE manages this by implementing a dynamic rebinning strategy in which large bins are split in half and small bins are reabsorbed.

#### 5.6.4.2 Modified Entropy Measure

In order to avoid overfitting by splitting on a statistically invalid sample, a modified entropy measure is introduced. The entropy measure is designed to encompass uncertainty about the underlying distribution, due to a small sample size, within the measure. This uncertainty arises in small samples because a given observation of class numbers could be generated by a wide range of possible distributions. Any given distribution of  $c$  classes can be represented by a  $c$ -vector,  $p_i$  representing the probability of each class. These  $c$ -vectors form an  $c - 1$  dimensional simplex embedded in the  $c$ -space of the vectors. IGLUE makes the a priori assumption that each possible distribution is equally likely (in the implied metric of the simplex). The probability that each distribution would cause a given observation,  $\{a_i\}$  (with  $a_i$  instances of the  $i$ th class), can be calculated, and from this the probability density that each distribution is responsible for the observation can be obtained by application of Bayes' rule. This can be integrated over

## **218 See<sup>++</sup> : An Object Oriented Theory of Task Specific Vision**

---

the simplex to obtain the expected value of the distribution responsible:

$$E(p_j | \{a_i\}) = \frac{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{c-2} p_i} p_j \prod_{i=1}^{c-1} p_i^{a_i} (1 - \sum_{i=1}^{c-1} p_i)^{a_c} dp_{c-1} \cdots dp_1}{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{c-2} p_i} \prod_{i=1}^{c-1} p_i^{a_i} (1 - \sum_{i=1}^{c-1} p_i)^{a_c} dp_{c-1} \cdots dp_1}$$

Each integral can be solved, repeatedly integrating by parts to obtain:

$$E(p_j | \{a_i\}) = \frac{a_j + 1}{c + \sum_{i=1}^n a_i}$$

Interestingly, this is the distribution that is obtained by adding one to the number of observed instances of each class. The entropy of this distribution is used Thus the IGLUE entropy measure is:

$$H(\text{Leaf}) = -\frac{\sum_{i=0}^c (a_i + 1) \log_2(a_i + 1)}{n + c} + \log_2(n + c)$$

Using this entropy measure it is possible to avoid overfitting and the consequent need to prune.

### **5.6.4.3 Decision Graphs**

See<sup>++</sup> builds decision graphs rather than decision trees. This generates two advantages: Compactness, and a reduced number of leaf nodes.

Decision graphs can represent certain concepts more compactly than decision trees. The relation,  $(a \vee b) \wedge (c \vee d)$ , is the simplest example for which a decision tree has to represent some concept twice. This implies that

---

#### **Algorithm 5** IGLUE learn

---

```
Learn()
for each leaf, L do
    for each neighbour of L, N do
        H = entropy loss of merging L,N
        if H < ψ then
            Merge L,N
        end if
        I = best partition information
        if I > Ψ then
            Split L into L1, L2
            Update neighbour list for L1, L2
        end if
    end for
end for
```

---

the concept has to be learned in both branches of the tree. An algorithm has been developed [88, 234, 235] which builds decision graphs by using a minimum message length principle. By directly inducing over trees, it is possible to use examples from both paths to learn the relation. This facility will tend to speed the learning process in the incremental context and improve generalisation.

IGLUE achieves this by permitting leaf nodes to merge as well as to split (see Figure 5.10), however, the merging process is carefully controlled (see Algorithm 5). For leaves to be merged, two criteria have to be met:

1. The information loss due to merging must be below a threshold
2. The leaves must be neighbours in feature space

The latter criterion is present to handle the case when instances of a given classification are located in disparate regions of feature space. These regions correspond to different reasons why a part may be of that classification. Because any given feature space is unlikely to completely classify a part, it is important to retain a handle on the differing reasons for class membership so that this information can then be used to assist in building relational rules.

#### 5.6.4.4 Stochastic Functions

It is often the case that a classifier will be unable to completely separate instances of different classes. However, many classifiers have this model of the data (ie that at each point in feature space there is a unique class, possibly with some noise added). This assumption can affect the methods used

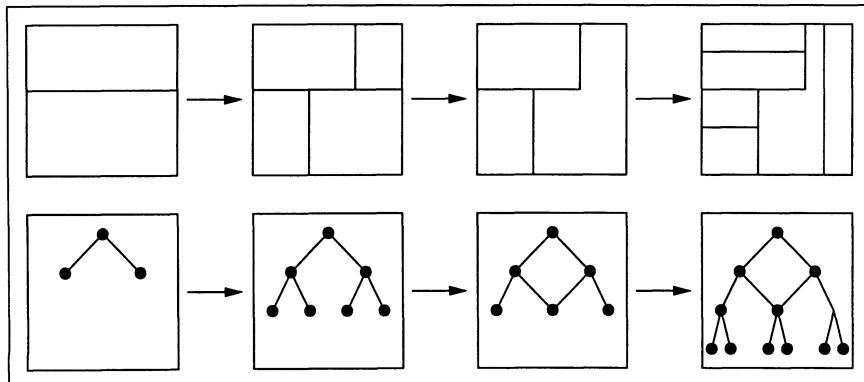


Figure 5.10: Building a decision graph

to store and separate examples (least generalisation for example). IGLUE is explicitly designed to learn stochastic functions where each point in feature space has a distribution associated with it describing the probability of seeing each class at that point in the space. Thus the output of the decision graph, when an example is classified, is the leaf in which the example ends up, together with the probability distribution for that leaf.

### 5.6.5 Relational Learning in See<sup>++</sup>

To handle relational data, See<sup>++</sup> learns in three stages (see Figure 5.11):

1. IGLUE is used to learn about image parts based on their own attributes. This breaks the feature space of parts into a number of leaf nodes.
2. Pairs of leaf nodes are identified where instances of one leaf affect the classification of instances of the other. This is achieved by computing a set of correlations for each leaf node.

For each image presented to See<sup>++</sup>, each leaf node,  $L$  has a number of instances,  $n_L$ . These instances are user labelled, thus each leaf node has a proportion of its instances in each classification,  $p_i$ . Within each leaf node  $p_i$  is correlated with  $n_L$  over a series of images. The

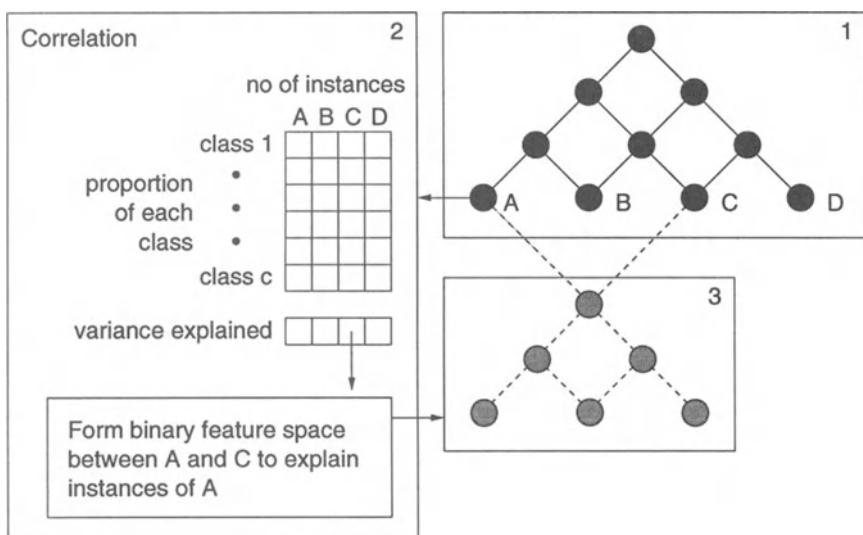


Figure 5.11: Relational learning in See<sup>++</sup>

squares of the correlations are then summed over the classes, *i*. A high value for leaf node,  $L$ , implies that the number of instances of that leaf node in an image affects the way instances of the given leaf node are classified. This then triggers the next stage of learning:

3. IGLUE is applied to each feature space defined by these pairs. The dimensions of this feature space correspond to the attributes of the relationship between pairs of instances (one from each leaf). For leaf nodes which remain unclassified, stages 2,3 are repeated until no more information can be obtained about these parts.

These methods of learning are now capable of generating rules of the form described in Section 5.5.3. Stage 1 generates the unary and merge rules (5.1, 5.2). Stage 2 identifies the  $j$  and stage 3, the  $B_k$  for the binary rule (5.3).

## 5.7 See<sup>++</sup> in Action

The See<sup>++</sup> system is designed for use in a number of scenarios over a range of conditions. A generic application has been developed which provides the user with a set of displays which allows them to interact with the knowledge base and current image interpretation. This generic core has been used in a number of ways, in particular for teleoperation.

### 5.7.1 Implementing the Generic Application

The core application comprises two main displays, both managed by the knowledge base; one for the current interpretation and the other for the knowledge base. There are also a number of subsidiary windows for managing the various daemons which control the machine learning process.

#### 5.7.1.1 Current Interpretation

A series of specialised knowledge methods have been developed which manage the display of image parts. These methods can display symbolic objects with which the user can interact via the mouse in the graphical user interface. Additional knowledge methods can obtain chain codes of part boundaries or region shapes to flesh out the display in more detail. The display objects can also be colour coded by the machine classification as in Figure 5.12(a).

Because these methods can be attached anywhere within the knowledge base, it is possible to make effective use of the IQL client-server bandwidth by reserving use of the more expensive shape transmission methods for objects of particular interest to the user.

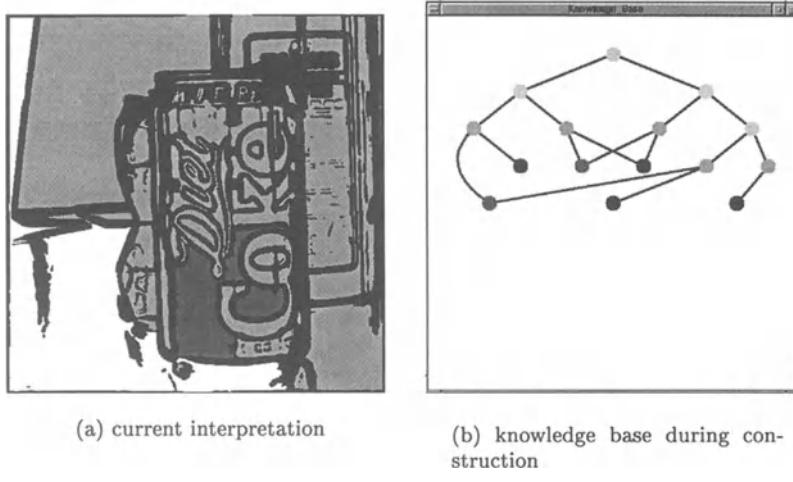


Figure 5.12: Principal system displays

During training, full image information is required to assist user labelling of the parts. In this case, the method which obtains image shapes is connected to the root node in the knowledge base hierarchy. This achieves full image display because all parts generated by the IQL server are passed to the root node for initial processing, thus the shape of every region is obtained. During run time operation, however, it is possible to choose to obtain rich shape information only for parts which have particular classifications (for example, the floor area in the teleoperation scenario).

### 5.7.1.2 Knowledge Base

The knowledge base is displayed separately. The display shows the lattice structure of the knowledge base adaptively growing as knowledge is acquired through supervised learning. The functionality of this display comes from an abstract class from which knowledge nodes are derived. The user can interact with knowledge nodes (and the methods attached to them) via this display. The nodes in the display (Figure 5.12(b)) are colour coded by the type of learning that they represent.

- pale gray nodes have generated their children through unary learning.
- mid gray nodes are performing correlation learning.
- dark gray nodes are forming binary relationships.
- black nodes are user classifications (noumena).

### 5.7.2 Results

Figure 5.13 shows a series of images from the training set used for teleoperation in an office environment. In this scenario, the system was required to identify six object classes (floor, wall, chair, bin, door and table), in addition to those classes which are used for internal purposes such as re-segmentation. In this scenario, the knowledge base was able to reduce the classification entropy from an a priori value of 1.94 down to 0.89 using unary features. This stage explains approximately 55% of the uncertainty. Correlation learning was then able to explain approximately 50% of the variance in classification probability in leaf nodes with high entropy ( $>0.1$ ). Overall, the system correctly classified 85% of all parts in the test set. Figure 5.14 shows typical output for a previously unseen image. This figure shows an image labelled by colour coding the image parts. The main objects in the display have been correctly labelled with a few exceptions such as the monitor which has been incorrectly identified as wall.

See<sup>++</sup> has also been tested in other scenarios, from simple block world test situations (used during the construction of the See<sup>++</sup> framework) to scenarios involving exterior scenes, such as surveillance. Results from these scenarios varied from recognition rates of 70% for unstructured scenes, to close to 100% for block worlds.

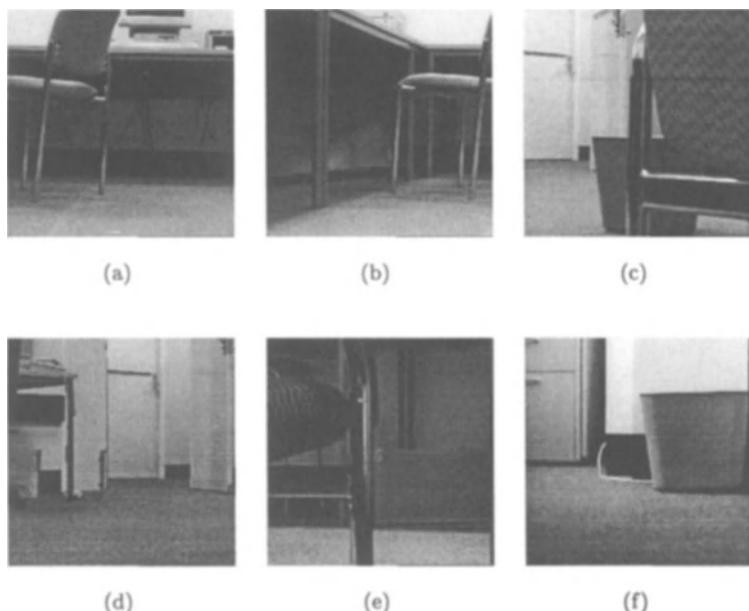


Figure 5.13: Sequence of images from the training set

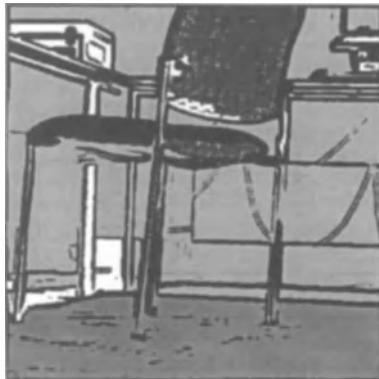


Figure 5.14: Typical output from See<sup>++</sup>

#### **5.7.2.1 Special Considerations**

A number of special modifications have been incorporated in the building of See<sup>++</sup> to enhance performance. One such is the differential weighting of image segments during learning according to their size. This modification adjusts the number of examples that are recorded in probability distributions according to the area of the region. This value is not necessarily an integer and varies between 0 and 10. Thus large regions are effectively recorded several times, while small regions only have a fractional impact. This improves the recognition performance of large regions at the cost of smaller ones.

#### **5.7.3 Future Directions**

There are a number of ways in which the See<sup>++</sup> project could be extended, principally into explicitly temporal domains. Thus part attributes would include dynamical information, such as velocity and acceleration. Additionally, binary relationships could incorporate temporal connections between events. This would permit the learning of classes that are defined by dynamical behaviour not evident in a static scene. Temporal information could be included in extensions to the IQL which use predicted motion of image parts to speed up future segmentations.

## Chapter 6

# SOO-PIN: Picture Interpretation Networks

Sandy Dance and Terry Caelli

### Abstract

Traditional methods for image scene interpretation and understanding are based mainly on such single threaded procedural paradigms as hypothesise-and-test or syntactic parsing. As a result, these systems are unable to carry out tasks that require concurrent hypotheses. In this chapter we describe our developing work in hierarchical, agent-based systems for symbolically interpreting images. The systems are able to interpret dynamic events based on object motion and interaction among objects. Such capability can be applied to many image applications such as biomedical images, traffic control, and behaviour studies. The systems have been implemented in an object-oriented environment in, firstly, the logic programming language Parlog++, and later, the procedural object-oriented language Java. Both systems propagate uncertainty through the agents using Baldwin's (1986) formulation [16]. The system is illustrated with interpretations of traffic intersection images.

## 6.1 Introduction

In this chapter, we describe two stages in a novel approach to high-level interpretation of images and illustrate the system with legal interpretation of traffic scenes.

Computational vision has traditionally focused on problems of low-level sensing, pattern and object recognition, and relatively little attention has been paid to interpreting an image once primitive structures have been identified. This entails fitting the image into a broader context, or equivalently, bringing to bear domain knowledge in order to produce an interpretation. Such domain knowledge is typically symbolic. It is therefore important to develop an architecture that is able to integrate the domain knowledge with the labelled image data. In the past, some image understanding approaches have been reported in the literature, of which most were based either on hypothesise-and-test or syntactic parsing, both falling into the single threaded procedural paradigm.

Neumann [228] takes 3D “geometric scene description sequences”, which contain data on the time, location and orientation of objects ideally found in real images (actually “hand coded”) of traffic scenes, and uses a failure-driven backtracking algorithm on an associative net to generate case frames. These frames are used as the interface to a more formal logic programming language. The system generates statements like “car A passes truck B and turns into the side-street”, which is comparable to but not as complex as the output from our system. Like the other systems described in this section, this system is not concurrent and object-oriented.

Bell and Pau [23] have developed an object-oriented logic programming system for picture interpretation. Their system is based on the Prolog failure-driven/backtracking hypothesise-and-test system. The object oriented component of the system is implemented with a preprocessor that translates the code into standard Prolog. This three level (feature, application dependent, and object identification) system is used to find objects (cars) in natural scenes. Note that it is an object identifying system and is not concerned with high-level output.

SIGMA developed by Matsuyama and Hwang [198] is a system that finds objects in aerial monochrome views of housing estates. It also uses an object-oriented approach, where each object instance is used to establish a concept, and the control is by a single threaded failure-driven reasoner working on the equivalents of Horn clauses (in fact the same control mechanism as that used in Prolog, and so Bell and Pau’s [23] system). The low-level processing is under the control of the higher-level reasoning. However, the system does not go beyond identifying objects in the scene, for instance, into the interpretation of higher-level concepts like “what kind of housing estate”, or “utility routings”.

Huang et al. [144] use a Bayesian belief network and inference engine

[10] in sequences of highway traffic scenes to produce high-level concepts like “lane-change” and “stalled”. Belief networks propagate values around the network as vectors, with each link having associated matrices reflecting the conditional probabilities. One problem with Bayesian inference is that it requires that each node have a set of exhaustive and mutually exclusive states, which is often difficult to obtain in vision. This point is discussed in Section 6.2.2.1. Furthermore, the message passing in their system is numeric rather than symbolic, and, in distinction to our system, does not handle concurrent hypotheses and does not deal with interactions between objects in the scene, for instance, overtaking or giving way. In fact, it is not a symbolic system in the usual sense.

Another point to note is that some of the systems described above are monolithic in structure. While this is a reasonable way to build vision systems, it does not capture the nature of high-level human conceptualisation. Since we are interested in building systems that deal with concepts humans find interesting, alternative design philosophies are required. In this chapter we explore one such approach.

The notion of a “network-of-frames” emerged in the mid 1970s and was crystallised by Minsky [211]. He outlined an approach to the fields of both vision and natural language processing based on the notion of a “frame”, which is a data structure for representing a stereotyped situation. Attached to each frame are “terminals” in which specific information about the frame is stored. Terminals can contain “demons” for calculating and validating information, or default values, or links to other frames. If sufficient terminals find mismatched data, the frame may invoke another with better matching terminal constraints. For instance, when entering an unfamiliar room, the room frame terminals contain default values like left wall, right wall, ceiling but with no details. The terminal demon for, say, the left wall actively seeks out information to instantiate its values and, if necessary, activates the wall subframe. If the wall demons cannot find good values, then the “room” frame may be replaced by, say, a “backyard” frame. This system has elements of top-down and bottom-up control: bottom-up when a new frame is activated or from the tension of a mismatched frame, top-down when a frame activates its terminal demons.

Similar ideas put forward around the same time are Carl Hewitt’s “actors” [136, 137, 375] and Roger Schank’s “scripts” [285, 301].

Draper et al. [89] recently developed SCHEMA using a frame-based structure for low-level vision applications. In this system there are concurrent processes communicating through a blackboard, with structure matching and forward chaining. This system is used to segment natural 2D images, but unlike our system is not concerned with high-level interpretation. It should be pointed out that one needs to keep the interconnectivity between frames reasonably low, or as the cardinality of the system goes up the time each frame spends handling non-productive messages becomes

untenable. Blackboard systems are subject to this, unless one introduces some kind of partitioning of the blackboard. However, in this case, the blackboard system is then logically equivalent to a network-of-frames (with appropriate connectivity).

The systems by Bell and Pau [23] and SIGMA [198] discussed above, do not qualify as networks-of-frames as failure-driven backtracking is incompatible with such an approach. That is, frames, as we define them, are message passing processes that exist in real time and cannot retract (i.e., backtrack) a message once sent.

What we describe in this chapter are variations on Minsky's frames concept involving networks of concurrent processes, each dealing with one aspect of the interpretation. A novel aspect of our system is that the network extends beyond the object recognition stage to higher-level concepts based on dynamic interactions between objects, for instance "car A gives way to car B due to road rule X". This is comparable to but more complex than the level of output from Neumann's [228] system.

Recently, network-of-frames systems have been called **agent-based** systems provided they adhere to certain conditions [368]:

- **autonomy:** agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- **social ability:** agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- **reactivity:** agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- **pro-activeness:** agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative."

In the following we will adopt this nomenclature.

In this chapter we present symbolic approaches to image interpretation that use agents and concurrency. Our first system deals with "instantaneous" image sets (3 images separated by 400 milliseconds), and in Section 6.2.3 we move on to describe our second stage of the system which interpret continuous image sequences.

Our system has the following features:

- Agent-orientation reflects, to some extent, the active and modular nature of human concepts, as Lakoff [172] pointed out, and eases the construction and modification of the system

- Each agent executes concurrently, which again allows exploitation of modularity in the domain, simplifies design and implementation, and allows for future enhancements with real-time processing and parallel computation.
- The agents are organised as a hierarchy from low-level agents dealing with simple objects in the scene to high-level domain specific relationships.
- Our system handles uncertainty through a modified form of Dempster-Shafer calculus as described by Baldwin [16].
- Most importantly, the input interpreted need not simply be static images, but can involve more complex data, for instance, image sequences. This allows the system to deal with dynamic concepts like “give way”, “collision”, etc, based on information about vehicle velocity, trajectory, and driver intentionality.

It should be noted that the system as implemented has proven robust, reliable and gives consistent interpretations. It has been tested on several scenarios [76–78] and it can be used in many other applications with dynamic object interactions, for instance, biomedical image analysis and robotic vision. In the following sections we describe in detail our staged systems.

## 6.2 The SOO-PIN System

Our first SOO-PIN (symbolic object-oriented picture interpretation) system is implemented in a logic programming language. The best known example of this kind of language is Prolog, which is based on the resolution principle [288] and failure-driven backtracking. Prolog is single-threaded, that is, each clause of a goal is tried in turn until one succeeds. Another approach is to try the clauses concurrently, committing to the first clause that satisfies guard criteria. An example of this type of language is Parlog [73].

The object-oriented programming (OOP) paradigm is a model for programming based on the the notion of computational objects which communicate via message passing. This paradigm has features in common with Minsky’s [211] frames and Hewitt’s [137] actors described above (in fact Hewitt was influenced by an early version of SMALLTALK). Shapiro and Takeuchi [310] have shown how object-orientation can be implemented in a concurrent logic programming language through the following identifications:

- An object is a process that calls itself recursively and holds its internal state in unshared arguments.

- Objects communicate with each other by instantiating shared variables.
- An object becomes active when it receives a message, otherwise it is suspended.
- An object can respond to a message either by sending the answer back in another message, or instantiating an uninstantiated answer variable in the message.
- Class hierarchies and inheritance can be formed using what Shapiro and Takeuchi call filters, which actively pass messages from a class to the superclass when the method to handle a message is not available in the class.

Parlog++ [80] is such an object-oriented extension to Parlog. The OOP features (except for hierarchies and inheritance) described by Shapiro and Takeuchi are built into the language so that the programmer can write object-oriented code in a natural manner. As Shapiro and Takeuchi pointed out, their view of objects was based on Hewitt's actor model of computation, which in turn was influenced by and influenced Minsky's frames model. Thus Parlog++ is a most suitable language in which to implement our network-of-frames (or agent-based) high-level vision system.

### 6.2.1 The SOO-PIN Architecture

In this system, each agent (which we will call a “concept-frame”) corresponds to a concept to be explored, and instances of such concepts (here called “concept-instances”) are stored as data within the concept-frame. When a concept-instance of a concept-frame wheel (see Figure 6.1) is created—for whatever reason—it sends messages `checkA` to all associated concept-frames (according to how the system has been connected to reflect the domain knowledge) informing them of its existence. These messages, in turn, will prompt receiving concept-frames to check their “existence criteria”, possibly sending inquiry messages `inquiryB` to other concept-frames to establish the criteria, and if successful, creating a concept-instance. This new creation will, again, result in `checkB` messages being sent, and thus completing the cycle.

There are four basic message types in the system: `create` messages create a concept-instance in a concept-frame, `check` messages initiate the process of determining whether a concept-instance exists, `inquiry` messages return specific information from other concept-frames, and `update` messages add properties to specific concept-instances.

This architecture embodies both data-driven and knowledge-based control, where `check` and `create` messages represent the data-driven mode, and `inquiry` messages represents the knowledge-based mode, in that these

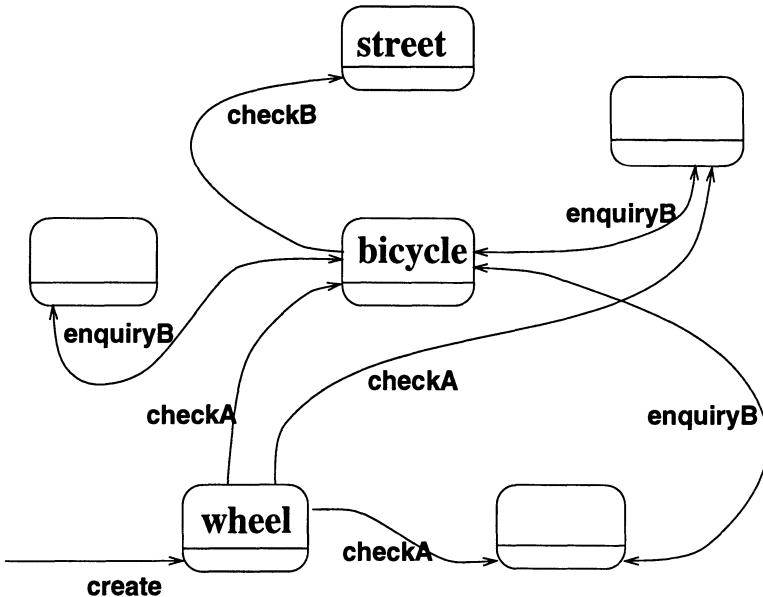


Figure 6.1: The SOO-PIN system concept: object concept-instance creations activate other associated objects through sending “check” messages. These activations in turn prompt further activations.

messages are prompted from the exploration of higher-level hypotheses. Using these two modes together avoids the large search space that results from using either alone. For instance, knowledge-based control can entail exhaustive depth-first search as the system begins searching at the top nodes of the search tree, and must check all nodes down to the data level, usually with failure-driven backtracking. On the other hand, data-driven search modes entail combining all data in all possible patterns in order to explore higher nodes in the search tree, and as Tsotsos [339] has pointed out in the area of vision, this has exponential complexity. Our claim that mixing top-down and bottom-up modes is more efficient than either alone, is confirmed in the area of linguistic parsing by Allen [4], who gives mixed-mode chart parsing as an example.

To facilitate message passing and communication with the user, the “switchboard” module spawns off concept-frames, if need be, and keeps track of their input streams. The output streams from the concept-frames are combined in a merge process, and, together with the console input are directed back to the switchboard input stream (see Figure 6.2). That is, concept-frames communicate with each other through the switchboard. The concept-frames and switchboard also write output messages directly to windows on the user console for monitoring purposes.

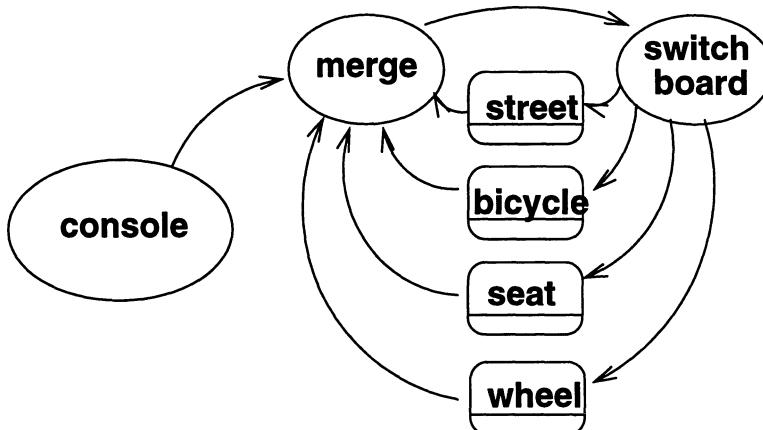


Figure 6.2: The implemented system architecture. Messages are relayed around the system by being merged into the switchboard's input channel, and then distributed to various concept-frames (shown as rectangles). Ellipses represent other concurrent processes, and arrows are message channels.

In some cases, deduced objects (things in the world) are composed of more primitive objects. Such compound objects, or simply, compounds, do not derive all relationships symbolically from their components. For an example: if a compound C is composed of A and B, then an object D may be near C but not near A or B. “Nearness” depends on the *scale* of the objects involved, which cannot be handled purely symbolically. As a consequence, it is necessary to refer back to the spatial data.

This problem illustrates our position that high-level vision is not a purely symbolic activity, rather it lies in the symbolic-spatial realm. In order to deal with both symbolic activity and spatial data reference more effectively, in our system the symbolic machinery of the agent-based system is used as a control mechanism, which is combined with procedural calls (to C routines) to instantiate certain spatial predicates. Ideally, we should extend our agent-based system down into the image segmentation and region labelling activities, freely passing messages between the highest and the lowest level agents. Here we perform the low-level processing initially, producing a spatial database of found objects with their coordinates. It is this database that is accessible from high-level concept-frames throughout the network to resolve the spatial predicates.

### 6.2.1.1 The Concept-Frame Structure

In the concept-frame, concept-instances are held as members of a list. Each concept-instance is a data-structure containing the object type (e.g., car or road), a unique identifier, and a list of properties or relationships held by the concept-instance. Below is a simplified example of a concept-frame program, in this case, that of the car.

car.

```
        Out o-stream          #output stream name
invisible InstList state, OutFile state
                #state variables

initial open(car.log,write,OutFile)-
                #activities when starting up concept-frame
clauses
    create(Id,Props)      =>
        #clause to execute upon
        #receiving a create message. Concept-instance
        #Id is added to InstList with its Props

    inquiry(Id,Prop,FoundProps) =>
        #clause for an inquiry message, the property
        #Prop is searched for in concept-instance
        #Id, and the answer bound to FoundProps

    update(Id,Props) =>
        #clause for an update message,
        #Props are added to property list of
        #concept-instance Id

    last =>
        #clause to execute upon input
        #stream close

end.           #end of concept-frame
```

In this example, the car concept-frame initially receives a `create` message from the low-level vision system at startup, whereupon a new instance is created. Later another concept-frame may send an `inquiry` message to the car concept-frame, for instance inquiring what cars exist in a given intersection. The answer is bound to a variable in the query which thus becomes available to the sender. If another concept-frame, say `traj` (see

Figure 6.4), determines a fact about the car like its velocity, it sends the information to the car frame as an update message which is appended to the car’s property list. Such facts are then available to other concept-frames through the inquiry message.

In accordance with the parallelism of Parlog++, all the clauses in the above code can execute in parallel, spawning a process for each message on the input stream. They can also run sequentially, finishing one message before starting the next. These modes are under the control of the programmer. In SOO-PIN, we execute sequentially within concept-frames, and run separate concept-frames in parallel.

### 6.2.2 Handling Uncertainty

In our agent-based system, we have attached a belief measure to the various propositions as they are conveyed through the network in order to give the final interpretations a statement of confidence. Baldwin [16] has developed a formulation for combining belief measures of two propositions in independent frames of discernment from the Dempster-Shafer [82, 187, 309] perspective. The reason for exploring this work is that such combinations fit nicely within logic programming. For instance, a typical clause might be  $P \Leftarrow A \wedge B$  ( $P$  is proven if both  $A$  and  $B$  are proven). To include uncertainty we can use Baldwin’s combination of independent propositions rule to compute the uncertainty of  $P$  given the uncertainty  $A$  and  $B$ . If  $A$  and  $B$  have belief intervals  $[\mathcal{S}(A), \mathcal{P}(A)]$  and  $[\mathcal{S}(B), \mathcal{P}(B)]$ , (where  $\mathcal{S}$  is support and  $\mathcal{P}$  is plausibility), then Baldwin’s combination rules are:

$$\begin{aligned}\mathcal{S}(A \cap B) &= \mathcal{S}(A) \cdot \mathcal{S}(B) \\ \mathcal{P}(A \cap B) &= \mathcal{P}(A) \cdot \mathcal{P}(B) \\ \mathcal{S}(A \cup B) &= \mathcal{S}(A) + \mathcal{S}(B) - \mathcal{S}(A) \cdot \mathcal{S}(B) \\ \mathcal{P}(A \cup B) &= \mathcal{P}(A) + \mathcal{P}(B) - \mathcal{P}(A) \cdot \mathcal{P}(B)\end{aligned}\tag{6.1}$$

Thus, in the example above, the support for  $P$  is  $\mathcal{S}(A) \cdot \mathcal{S}(B)$  and the plausibility is  $\mathcal{P}(A) \cdot \mathcal{P}(B)$ . These rules are analogous to the Bayesian rules, which is reasonable as Dempster-Shafer reduces to Bayes when support equals plausibility [134]. These rules allow us to propagate uncertainty values through a network, each node passing a belief pair attached to a proposition in a message to other nodes which can then calculate the belief pairs for their derived propositions. Of course, the network cannot contain any dependency loops, as this violates the independence pre-condition for the combination rules.

We now derive a new algorithm to calculate belief measures in the case where an object exists given the existence of two components and a relationship between them, out of a population of many possible components.

For instance, we may decide that a cube D exists if any two of the sides A, B and C are found. That is:

$$D \Leftarrow (A \cap B) \cup (B \cap C) \cup (C \cap A) \quad (6.2)$$

We cannot use the usual Baldwin formulae (Equation 6.1) to find the belief in D as the disjunctions are not independent, nor is there any transformation that turns them into a set of independent expressions. To calculate this type of expression, we need a more fundamental treatment.

If there are  $M$  parts there are  $N = M \cdot (M - 1)/2$  potential relationships, and our object exists if at least one such relationship is true. The probability is therefore given by Feller's [104] formula for the probability of at least one event occurring among N:

$$P = S_1 - S_2 + S_3 - S_4 + \dots - (-1)^{(N-1)} S_N. \quad (6.3)$$

where  $S_1 = \sum P_i$ ,  $S_2 = \sum P_{ij}$ ,  $S_3 = \sum P_{ijk} \dots$ , where, for instance,  $P_{ijk} = P\{R_i R_j R_k\}$  is the probability of  $R_i \cap R_j \cap R_k$  and  $R_i$  is the  $i$ th relationship (event) in the list of N. This formula is quite general as there is no assumption about the independence of the  $R_i$  within the conjunction.

To move this into the Dempster-Shafer and thus Baldwin perspective, we need to deal with support and plausibility separately. In deriving his formula, Feller [104] computes probabilities of events, say A, from the frequency of sample points being "contained" in it, i.e., from a Venn diagram model. Here we can use the same model for support and plausibility by saying A occurs when a sample point is in it (for support) and a sample point is *not* in its complement (for plausibility). Note that in this Dempster-Shafer model, we are assuming that a frame of discernment consists only of a proposition and its negation. With this alteration, Feller's argument can be used for both support and plausibility, and thus the algorithm below is identical for both. This symmetry is also visible in Equation 6.1.

We can now describe the algorithm to find the belief in an object given its parts and their relationships (in the form of a list of relationships, each relationship containing its belief and its endpoints with their belief). Note that we only need to give a description for support because, by symmetry, plausibility follows.

The algorithm is based on Equation 6.3. First, we take the power set of the list of relationships giving a series of lists of all lengths from 1 to the length of the original list. Using Equation 6.3, each  $S_n = \sum P_{ij\dots q}$  is the sum of the supports for all conjunctions of relations of length  $n$ . To calculate a specific  $P_{ij\dots q}$ , first find the union of the endpoints of the  $n$  relationships  $R_i, R_j, \dots, R_q$ , say  $O_u, u = 1 \dots k$  where the union has  $k$  endpoints. Since the  $R_i$  and  $O_u$  are independent, we can use Equation 6.1 to give the support for their conjunction:

$$P_{ij\dots q} = \mathcal{S}(R_i) \cdot \mathcal{S}(R_j) \cdots \mathcal{S}(R_q) \cdot \prod_{u=1}^k \mathcal{S}(O_u) \quad (6.4)$$

This is calculated for all such  $P_{ij\dots q}$  and summed to give  $S_n$  above. These  $S_n$ , for each  $n$ , are then summed, alternating signs in accordance with Equation 6.3, thus giving the total support for at least one relation existing among  $N$  possible. This algorithm is used extensively in SOO-PIN to calculate belief pairs.

Dubois and Prade [90], in a useful critique of Baldwin's work, point out a problem with Baldwin's Support Logic Programming, namely that the support pair for logical implication " $B \Leftarrow A [\mathcal{S}, \mathcal{P}]$ " does not have a mathematical definition, and is not equivalent to its declarative interpretation " $\neg A \cup B [\mathcal{S}, \mathcal{P}]$ ". Bearing this in mind, we have adopted Baldwin's approach to uncertainty in SOO-PIN, but with belief pairs attached to *facts* (i.e., there is a car at position  $x$  with belief  $[\mathcal{S}, \mathcal{P}]$ ) but not to *clauses*. This avoids the problem pointed out by Dubois and Prade and still allows us to use Baldwin's combination rules. This approach also allows us to use the existing Parlog++ syntax.

### 6.2.2.1 Belief and Vision

In computer vision tasks, there are a large number of potential frames, corresponding to the set of all possible image regions, which is the power set of the pixels in the image! The proposition "segment A is a motorcycle" is not independent of the proposition "segment B is a car" if the segments intersect, but they are also not in the same frame of discernment because these consist of sets of exhaustive and mutually exclusive propositions. Therefore we tend to have simple frames of discernment consisting of, for instance, "segment A is a car" and "segment A is not a car"; but with potentially complex interdependencies between frames of discernment. Consequently we need to be careful about dependency when combining evidence.

Richer frames of discernment occur when dealing with compounds of given objects. For instance, given a car A, the kind of turn it is doing can only be one of a finite list. It is this "closed world" that characterises frames of discernment.

### 6.2.2.2 Implementation of Belief Pairs

Belief pairs in SOO-PIN are conveyed through the network as a data structure `bel(S,P)`, where S is the support and P, the plausibility. This data structure is attached to all propositions of fact within the system, for instance, the proposition "car c1\_116 exists with support S and plausibility P" is expressed by the Parlog++ term `id(car,c1_116,bel(S,P))`.

Relationships (another kind of fact) are handled similarly. For instance, the proposition "car c1\_116 is on road 2" is expressed by the term `reln(in, id(car, c1_116, bel(Sc,Pc)), id(road,2, bel(Sr,Pr)), bel(Si,Pi))`, where the Sc is support for the car, Sr is support for the road, and Si is support for the relationship in (similarly, P for plausibility).

These belief data structures are conveyed around the network with the propositions in the various messages in the system—**check**, **create**, **inquiry** and **update**. The belief values are combined within the concept-frames using Equation 6.1 and the algorithm described at the end of Section 6.2.2. Finally, the high-level interpretations receive these belief pairs, which are displayed with the interpretation in the output (see Section 6.3.6).

### 6.2.3 Dynamic Image Sequences

In this section we describe a development of our system that deals with image sequences, thus introducing **dynamics**, concepts involving evolution through time under physical constraints. We are guided in our approach by van Gelder and Port [346] and their concern with the inadequacies of the **computational** model of cognition.

As van Gelder and Port point out, there are two main approaches to cognition in continuous temporal contexts, the dynamic (or connectionist) and the computational. The former approach is exemplified by neural nets and Kosko's [168] fuzzy cognitive maps, in which nodes (objects, agents, processes or modules) are connected together by arcs (links, synapses, relationships, channels) which convey continuous time-varying real values. The nature of the cognition is determined by the network structure and the object structure. For instance, in neural nets the incoming weighted link values are summed, normalised with the sigmoidal function and output to other links. In the computational approach, as articulated in Newell and Simon's "physical symbol system hypothesis" [229], the links between modules carry not time-varying values but symbols, hence the actions of the modules are batch-oriented rather than a continuous reaction to the input values.

The symbolic approach is exemplified by the work of [11, 228, 302] in which dynamic image sequences of traffic scenes or soccer games are analysed using event recognisers. Here at each instant the geometric relationships are described by a geometric scene description (GSD), which is updated frame by frame. This is input into the event recogniser, which is a transition network [11, 302] (working like a linguistic parser on a stream of tokens), or a set of logical clauses [228] with control based on unification and backtracking, i.e., the same control structure as Prolog. We feel that this symbolic approach does not take full advantage of the constraints offered by dynamics, and it requires a lot of computer resources having to roll forward the GSD from frame to frame, nor is this work based on an agent-oriented approach such as the one we espouse.

Work which is an amalgam of the connectionist and symbolic approaches is that of [284] in which a behaviour net (similar to a spreading activation network) conveys an "energy" value between modules. However, needing to convey some symbolic information, they use "pronomes"—places where

symbols are stored and accessed by other modules. Thus in this system, any symbolic information is either conveyed as dedicated channels between modules, or is stored in special modules which are updated through another set of dedicated channels, one channel or module for each possible value of all the symbols. It becomes apparent that this system scales badly as the domain complexity increases.

#### 6.2.3.1 Situatedness and Intentionality

In recent years, it has become clear that computer systems dealing with (modelling) dynamic real situations need to be “embedded” in that situation, that is, interacting with their environment, and in fact such interaction, in the case of computational systems, “grounds” or provides meaning to the symbols used. This is particularly true of systems for processing spatial relationships, for instance, the analysis of vehicle interactions. Situationists have made radical claims that cognitive systems use *only* context for their representations, and that there are no internal representations [68]. However, Slezak [316] has pointed out the need to distinguish the representation used internally to implement cognitive systems from that used for external communication, and shows how this clarifies the situationists’ claims.

Cognitive systems for processing complex dynamic spatial information need to perform at the intentional or semantic level. It is clear, for example, that understanding the movements of traffic under the control of humans is not going to successfully predict behaviour without in some way modelling the intentional states of the drivers. Specifically, to understand why a car is slowing down, it is necessary to model the give-way traffic rules which indicate that the car must give-way to another, as well as the driver’s intention to adhere to that rule. This can be generalised to any complex dynamic system involving “rational” agents, where each agent runs models of any other agent they are interacting with [367].

#### 6.2.3.2 A Dynamic Symbolic Interpreter

In Section 6.4, we describe a system which models the intentional level with a symbolic network. The pure dynamic approach is not currently viable due to the lack of a high-level description language and top-down construction paradigm. We employ a computational system which emulates a dynamic system. Here agents actively forward messages through the network provided the change from the previous state exceeds a threshold. Thus the symbolic system reacts to any changes while avoiding performing the same calculation repeatedly. Agents dealing with spatial concepts also have a dynamic aspect in that they interact with real-world sensors or an active spatial database.

In our dynamic network, we can distinguish between a number of agent types:

- **Instantaneous:** an agent dealing with concepts concerned with a single instant (frame), i.e., dealing with a particular segment in an image. These agents have a lifetime of a few frames only.
- **Continuous:** an agent dealing with a concept that has continuously time-varying parameters. For instance, a car with parameters position and velocity. This agent is constantly updated with the current values for the object involved, and is removed when the object disappears. When any change in its parameters occurs (over a given threshold), this agent sends messages to other relevant agents in the network.
- **Event:** an agent that deals with an event in space-time (i.e., a vehicle coming to a stop), which would carry as a parameter the time of the event. Such agents would have lifetime parameters as well, and upon expiry, would disappear.
- **Durational:** an agent dealing with extended events, for instance the entire turn sequence of a car. These agents would carry the start and stop times of the event.

These agents can deal with single objects in the scene, or more complex concepts based on relationships or interactions between objects.

Output from this system derives from top-level scenario agents which tell the story of an interaction between two or more objects in the scene, i.e., an event involving cars approaching each other, realising they are in a give-way relationship, one car slowing, then stopping, and eventual disappearance of the cars from the scene. As well, the system provides a short term story generator that is able to tell the instantaneous picture upon the user's request.

### 6.3 Interpretation of Traffic Scenes

We have implemented the first stage of the SOO-PIN system for a number of scenarios [76–78]. Here we describe an application which takes images of road intersections (see Figure 6.3), interprets vehicle activities (i.e., vehicle A is turning right from the west) and produces legal analyses of the scene such as:

- whether the car is on the wrong side of road or intersection,
- when a car should give way to another. For instance:
  - give way to right at intersection,

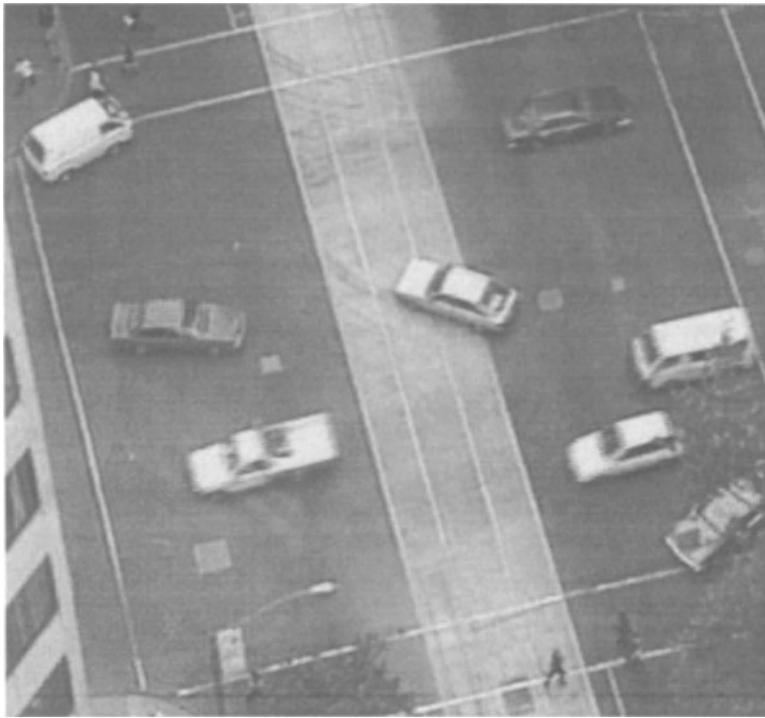


Figure 6.3: Typical traffic scene processed by the SOO-PIN system

- give way to oncoming when turning right,
- at T-intersections cars in ending road give way to those on through road,
- at traffic lights or give way signs,
- traffic jams (i.e., give-way deadlocks).

We chose to explore the traffic scenario because it is a complex domain with well-defined rules based on spatio-temporal relationships. It is a good test domain for SOO-PIN because the concepts operate at various levels of abstraction, that is, cars are found through low-level image processing, car velocities involve token correspondence processes, car activities involve working with geometric constraints in a phase space, and give way rules involve complex relationships between cars. It would be difficult to perform processing at these various abstraction levels without at least an object-oriented approach. Here we take this one step further into concurrency which allows us not only to deal with each object as a separate entity, but

also positions the system, in future enhancements, to deal with continuous time.

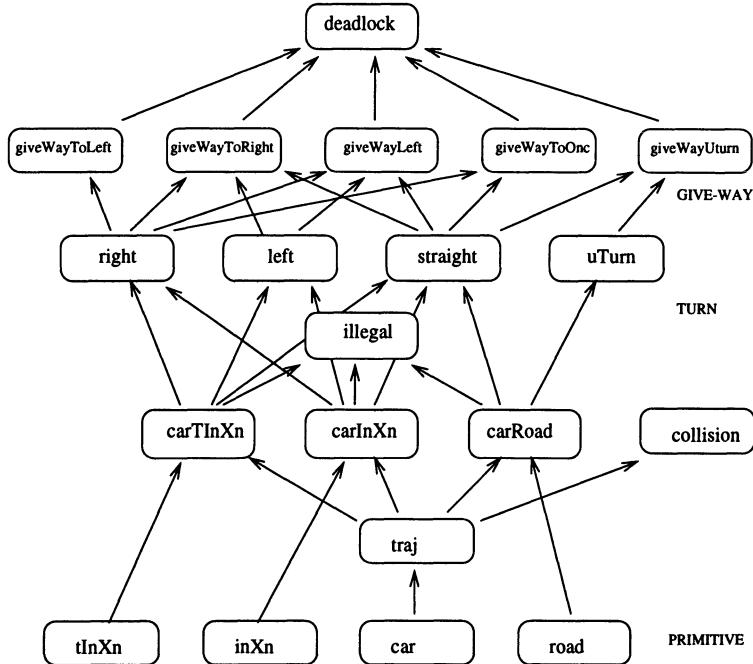


Figure 6.4: The traffic scenario network. The arrows refer to check or create messages, inquiry and update messages are not shown. *inXn* refers to “intersection”, *tInXn* to “T-intersection”, *carInXn* to the concept of a car in an intersection, *carTInXn* refers to a car in a T-intersection, and *carRoad* to a car in a road. *traj* determines car velocities.

The traffic domain has been incorporated into the network in Figure 6.4. We discuss the concept-frames in more detail below.

### 6.3.1 Primitive Concept-Frames

The concept-frames *car*, *inXn*, *tInXn* and *road* correspond to objects that are either found by low-level processing (see Section 6.3.4) or are given (i.e., intersection, T-intersection and road are constant). These concept-frames simply send check messages higher in the network, store information about the concept-instances, and respond to queries about them. The *traj* and *collision* concept-frames, which are discussed further in Section 6.3.5, calculate car velocities and determine whether they are likely to collide.

### 6.3.2 Turn Concept-Frames

On this level, `carInXn`, `carTInXn` and `carRoad` determine the containment of cars in road structures (roads and intersections), i.e., “car A is in intersection B”. Upon receiving a message from a car or road structure concept-frame, these concept-frames activate a procedural routine written in the C language that reads the spatial database and returns a list of containment predicates involving the sender. This is done because the high-level interactions of a car are dependent on the road structure in which it is contained.

In accordance with the basic philosophy of our approach (in which symbolic and numeric procedures are inter-related), when a car concept-instance is created, a call is made to a C routine to determine what the car is doing in the road structure. If car velocity is available, it is used to determine the heading angle of the car, otherwise car orientation is used (as explained below in Section 6.3.5.1). This routine first uses an affine transformation (to compensate for the camera view) that maps the intersection coordinates onto a square, thus normalising the car positions and heading angles. The various car activities, i.e., right turn, left turn, straight and illegal from the north, south, east and west, are unambiguously defined by regions within a 3D product space of the normalised car positions and heading angles. The turn activity found by this routine is sent in a `create` message to the appropriate concept-frame, i.e., `right`, `left`, `straight`, `uTurn` or `illegal`.

When the car position and heading angle fall close to the boundary of such regions within the 3D product space, confidence in the deduced activity decreases. To reflect this drop in confidence, or belief, we generate a belief pair (see Section 6.2.2) attached to the output predicate that decrease to  $[0.5, 0.5]$  as the boundary is approached (see Figure 6.5). Note that both support and plausibility drop to 0.5, reflecting the buildup of evidence against the predicate (i.e., *for* the predicate corresponding to the other side of the boundary). This way of dealing with confidence near numerical thresholds or boundaries is used throughout SOO-PIN.

### 6.3.3 Give-Way Concept-Frames

When concept-instances of turn concept-frames are created, they send `check` messages to the appropriate give-way concept-frames. These concept-frames check the context of the car given in the message to see if any other vehicle is in a give-way relationship with it.

Upon a give-way concept-instance creation, a `create` message is sent to the `deadlock` concept-frame, together with the identities of the two cars involved. This concept-frame checks for cycles in give-way chains: a legal deadlock or traffic jam.

Descriptive high-level output from the network is generated by `deadlock`,

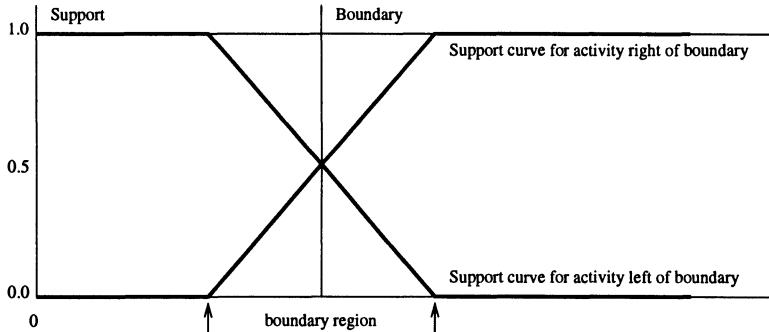


Figure 6.5: The support and plausibility values associated with assigning car activities near boundaries in a 3D product space. Support and plausibility in one activity drops to 0.5 as the boundary is approached because the activity is ambiguous at the boundary itself.

the give-way concept-frames, **illegal** and **collision**. See Section 6.3.6 for runtime examples of this output.

### 6.3.4 Low-level Processing: Extracting Attributes

The system runs on images recorded on videotape from a camera mounted above a city intersection (see Figure 6.3). These images were digitised using an Abekas Digital Video system (in 720x576, 24 bit RGB format). Background subtraction was used in the detection of vehicles by subtracting an image of the empty intersection from target views. The resultant images were then median filtered (using a  $3 \times 3$  window) to smooth regions and enhance edges. The resulting regions were then thresholded to remove regions too small to be vehicles, and then labelled. Assuming that each residual region corresponded to candidate vehicles, we then computed region (vehicle) attributes corresponding to centroids and best-fitting major axes (orientations).

These values, for each image and region, were stored in a “spatial database” together with the car identities and frame numbers. The (constant) intersection and road coordinates were also stored in this database.

### 6.3.5 Computing Velocities

Token correspondence was used to track cars in the image sequence where velocities of individual vehicles (selected regions) were found by comparing three successive video frames separated by 400 msec, as shown in Figure 6.6.

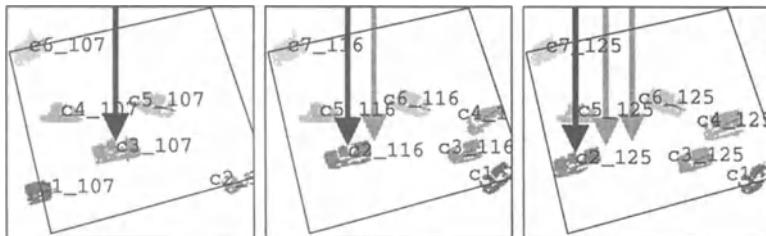


Figure 6.6: Example of 3 successive frames with the movement of one car indicated by the arrows, faint arrows referring to previous car positions. The intersection boundary is shown by the skew rectangle, the car labels are shown next to the cars.

The matching was based upon the computed centroids and orientations of the labelled regions (cars). The middle frame was considered the master frame, and each car found in this frame was compared with cars in the other two frames. A pairing was successful if the computed velocity required to move the first car to the second fell within bounds, and if the velocity was consistent with the orientations of the two cars as determined from their shape (see Figure 6.7). The set of possible pairings from the previous to middle was compared with that from the middle to successor frames, and the best pair of pairs was chosen as the most likely trajectory of the car.

This technique resolves a number of potential ambiguities discussed below. For example, in Figure 6.8 three cars are shown turning right. The system has a choice in dealing with car c2\_20. It could correctly identify its trajectory as a right turner, or join it with c1\_10 and c3\_30, producing a car going straight. This option was rejected because of the skew orientations with respect to the computed velocity vector, and the concordant rotation rates of the real trajectories. If, however, the input is limited to the cars c1\_10, c2\_20 and c3\_30 then the system does accept the straight trajectory because the algorithm finds the best available, within bounds.

Velocity estimation was done by the concept-frame `traj`. This concept-frame processes all car concept-instances from the middle frame, and calls a C procedure which returns lists of pairings from successor and previous frames. The lists were then compared in the Parlog++ language to determine the most likely trajectory of the car. This division of labour between C and Parlog is an example of our basic design philosophy in which spatial processing was handled in C, and symbolic processing was handled by Parlog++.

The chosen trajectory was then stored as a concept-instance of the `traj` concept-frame, the corresponding velocity was sent to the originating `car` concept-frame with an `update` message, and `check` messages were sent to higher level processes.

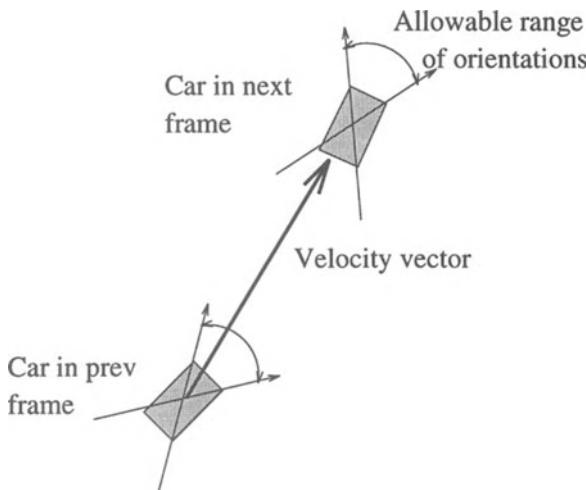


Figure 6.7: A pair of cars in successive frames, showing how the velocity vector between the cars must be compatible with the orientations of each car.

### 6.3.5.1 Using Velocities

As can be seen from Figure 6.4, a message is also sent from `traj` to `collision`. This concept-frame determines whether the car in the message is on a collision course with any other car. This is performed by calling a C routine that uses the velocities and positions of each pair of cars to calculate if the trajectories (nearly) intersect within a period corresponding to driver reaction time. If so, an output message is generated warning of the collision.

The concept-frames `carInXn`, `carTInXn` and `carRoad` obtain velocity data from the `car` concept-frame, and use this velocity to calculate the turn activity of the car in the intersection (or road). The support for velocity derived from the presence of car instances in three frames is computed by applying Equation 6.1 for disjunction to the car support from each frame. This increases the reliability of the resulting activity type over using the car orientation alone to calculate activity, as, for instance, if the cars from each frame have support of 0.8, then the total support for the sequence of three is 0.992. Using velocity also means that if a car is travelling on the wrong side of the road, the system can detect this, whereas if using only orientation, there is no way of knowing if the car is travelling in the wrong direction.

If the concept-frames `carInXn`, `carTInXn` and `carRoad` do not obtain car velocity (i.e., the car is not matched with cars in other frames) then

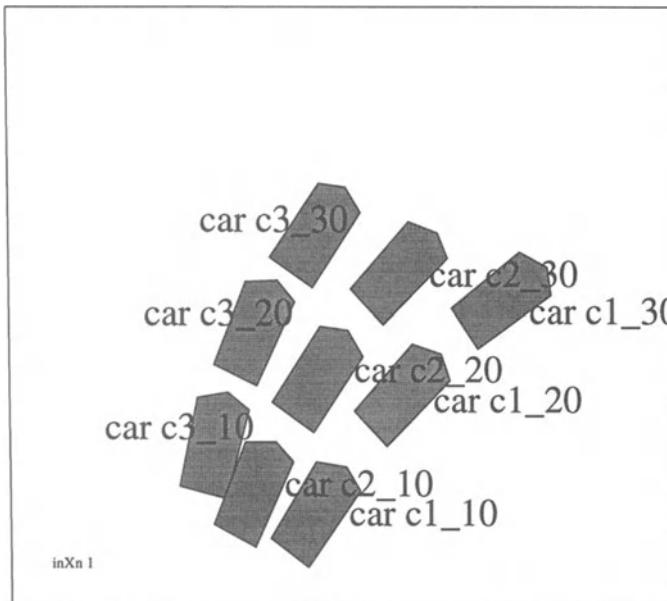


Figure 6.8: Diagram showing 3 cars turning right, over 3 frames. The system successfully found the 3 trajectories.

they can respond in one of two ways:

- ignore the car (since it was not confirmed in other frames it may not exist), or,
- use its orientation alone and continue, albeit with a lower “belief value” for subsequent deductions (due to the lack of support from the other frames). This option also generates less precise interpretations, as described below.

We have chosen to use the second option, as it produces more opportunities for further interpretation.

Finally, velocities are useful in generating the English language interpretations. For instance, without velocity the system would only produce vague output like:

**\*Give Way to oncoming: car c5\_116 turning right from west gives way to car c2\_116 from east.**

That is, it could detect there was a “give-way situation”, but it has no way of determining the tense, i.e., whether car c5\_116 has given way, or has not given way. With velocity this distinction is clarified, producing output like:

\*Give Way to oncoming: car c5\_116 turning right from west has given way to car c3\_116 from east.

and

\*Give Way to left-turner: car c6\_116 turning right from west should have (but hasn't) given way to car c4\_116 from east.

Given acceleration, the system could also generate phrases like “is giving way to”; moreover, acceleration is potentially available from the traj concept-frame since it has two velocities from comparing previous and successor frames. However, it has not been used in this study.

### 6.3.6 Results

Figure 6.9 shows a typical input image together with an intermediate result showing positions of cars found and their velocities, if any, within the intersection boundary. Note that, in this figure, the two cars with high velocity are c2\_116 and c3\_116. Other cars were found to be stopped as indicated by arrows without tails. Results from this intersection were all quite reliable due to the nearly vertical view. SOO-PIN generated the following interpretation:

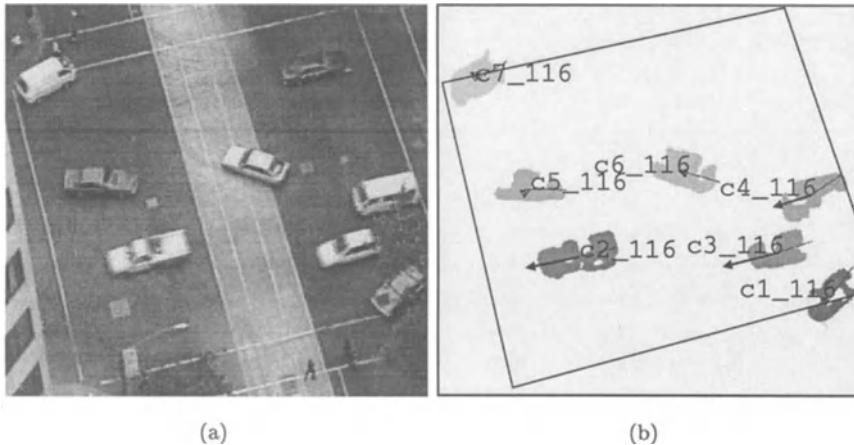


Figure 6.9: Collins & Exhibition Sts., frame 116. (a) Cars in intersection, original image. (b) Cars found by system, with orientations given by lines through cars, velocity given by arrows, and their labels. Intersection boundary input manually, north is up.

\*Give Way to oncoming: id(car,c5\_116,bel(1,1)) turning right from west has given way to id(car,c4\_116,bel(1,1)) from east : bel(0.60,0.98)

\*Give Way to oncoming: id(car,c5\_116,bel(1,1)) turning right from west has given way to id(car,c3\_116,bel(1,1)) from east : bel(0.62,0.98)

\*Give Way to oncoming: id(car,c5\_116,bel(1,1)) turning right from west has given way to id(car,c2\_116,bel(1,1)) from east : bel(0.90,0.98)

\*Give Way to left-turner: id(car,c5\_116,bel(1,1)) turning right from west has given way to id(car,c1\_116,bel(1,1)) from east : bel(0.93,0.98)

\*Give Way to left-turner: id(car,c6\_116,bel(1,1)) turning right from east has given way to id(car,c7\_116,bel(1,1)) from west : bel(0.93,0.99)

The next case (Figure 6.10) is from another intersection, with the camera considerably lower and hence giving an oblique view. The following interpretation was generated:

\*Give Way to left-turner: id(car,c4\_150,bel(1,1)) turning right from west has given way to id(car,c1\_150,bel(1,1)) from east : bel(0.62,0.98)

\*Give Way to left-turner: id(car,c5\_150,bel(1,1)) turning right from west has given way to id(car,c1\_150,bel(1,1)) from east : bel(0.62,0.98)

In this case every car activity has been correctly identified, and both the interactions in the interpretation are correct. One can see how two cars are speeding through the intersection, and the turners are creeping forward with short but correctly oriented velocity arrows. An enhancement for the future would be to find the reason why a car is stopped. In this case, cars c4\_150 and c3\_150 are waiting for the car in front.

Finally, we show a case where an illegal turn occurred (Figure 6.11). This is a T-intersection, with the straight-through traffic travelling east-west. SOO-PIN generated the following interpretation:

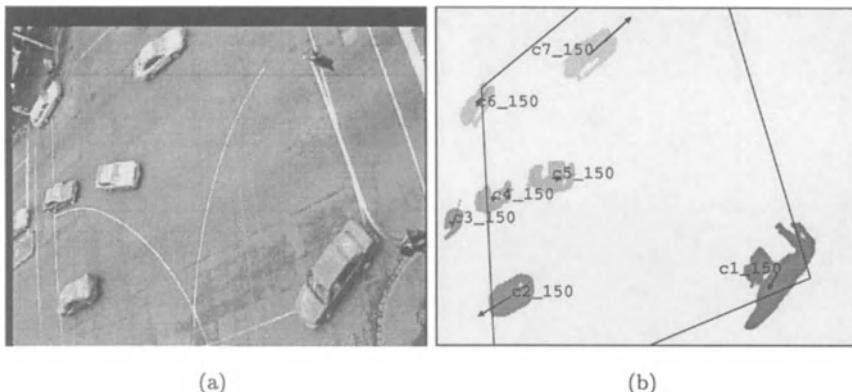


Figure 6.10: Lygon & Queensberry Sts., frame 150. (a) Cars in intersection, original image. (b) Cars found by system, with orientations given by lines through cars, velocity given by arrows, and their labels. Intersection boundary input manually, north is up.

\*Illegal: car id(car,c5\_316,bel(1,1)) from east is doing a right turn on the wrong side of id(tInXn,1,bel) : bel(0.72,0.78)

A car is doing an illegal U-turn across the intersection from the east and back to the east, currently facing north. The system detects that the car is at an unusual angle in the T-intersection, and deduces that it is doing a right-turn illegally, i.e., into a no-road.

### 6.3.7 Summary

In this experiment, the system was run on a sample of 10 scenes, involving 30 images from 3 different intersections. This resulted in 20 correct interpretations and 5 erroneous interpretations, as judged by a human. However, if the interpretations are weighted by the support given in their belief pair, the accuracy is 87%. This indicates that wrong interpretations have lower than average support, in other words, that our belief calculus is giving reasonable results.

Three of these wrong interpretations involved cars that were split in two due to a section of the car merging with the background (see Section 6.3.4). There were consequently two redundant interpretations, and one erroneous collision warning between the car halves. The rest of the errors was due to cars that were not assigned a velocity, resulting in the system using car orientations to deduce car activity, which, being derived from only one

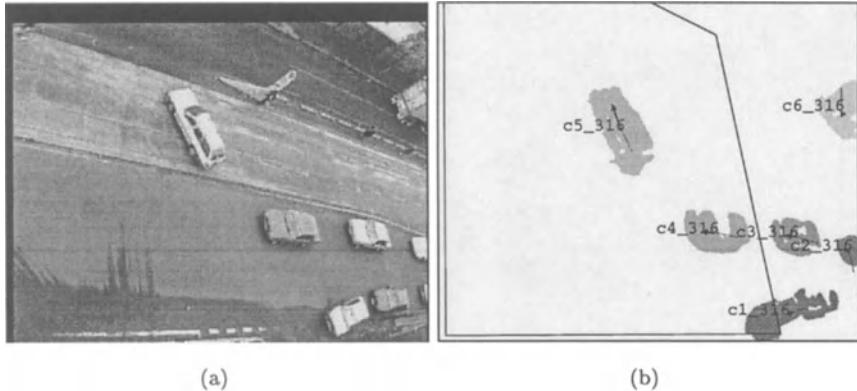


Figure 6.11: Swanston & Faraday Sts., frame 316. (a) Cars in intersection, original image. (b) Cars found by system, with orientations given by lines through cars, velocity given by arrows, and their labels. Intersection boundary input manually, north is up.

image, is unreliable. This kind of error could be averted by using the more conservative approach of ignoring cars without velocity.

These 10 scenes took 1763 seconds of CPU time on a SGI Personal Iris (with a 33 MHz processor) for low-level processing, and 76 seconds of CPU on a Sun SparcStation II for the high-level (Parlog++) processing, i.e., just over 3 minutes per scene. This was achieved without fine tuning the system for speed. It can be seen that far more time was spent on the low-level processing. This was due to the large size of the images, typically 720 by 575 pixels, together with the multi-pass processing required to perform median filtering and isolation and labelling of connected regions. Further, symbolic processing is inherently faster than image processing, provided the search spaces are small, which is the case here. For off-line processing, this speed is acceptable, but for real-time processing, i.e., responses of the order of 1 second, the low-level processing needs a speedup of about 180, and the symbolic processing about 8. The latter can be achieved with faster hardware. The low-level processing needs a combination of smaller images, redesign to reduce the number of passes, and faster hardware to achieve real-time capability.

## 6.4 Interpretation of Traffic Sequences

In this section we deal with the implementation of stage two of our work, extending the basic SOO-PIN concept to dynamic image sequences, espe-

cially of **traffic scenes**. Here we are aided by the work of Bruton et al. [45] whose recursive 3D filter provides us with a system to convert traffic image sequences into a file of positions and velocities of the trajectories of the moving segments in the sequence. This allows us to perform the high-level analysis of the scene using a set of agents derived from those employed in the static SOO-PIN (stage one) described in Section 6.3. This enhancement consists of converting the previous agents to **instantaneous** and **continuous** agents, and adding new **durational** agents. These latter embody the historical development of the various events, for instance, an individual car's turn history. The continuous agents are directly evolved from stage one SOO-PIN, with continuous update of data from the image sequence (i.e., cars found), together with destruction of the agent when the associated concept disappears (i.e., a car leaves the scene), as illustrated in Figure 6.12. There are two durational agents. The agent `whyStopped` is activated when a car stops, and actively seeks to determine the reason the car starts or disappears. The agent `history` accumulates the activities of the cars from the various levels in the hierarchy, and reports back when requested. Note that this hierarchy is sparser than that in stage one, as it is still under development.

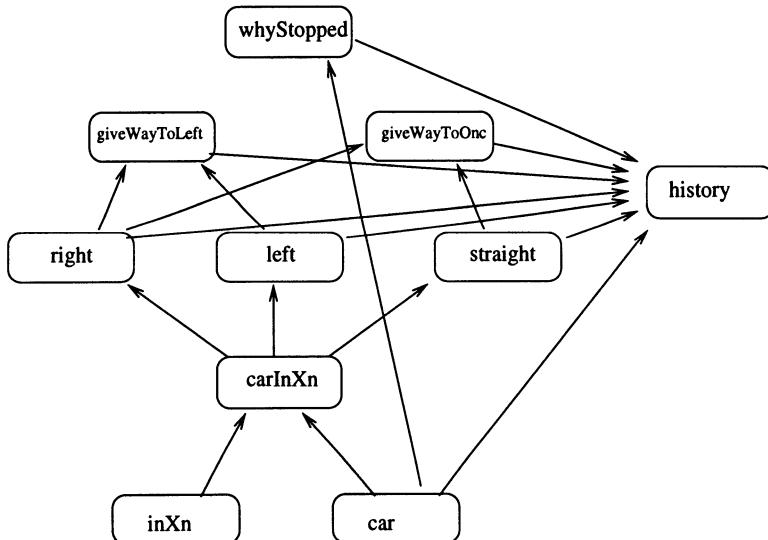


Figure 6.12: The dynamic traffic scenario network. The arrows refer to check or create messages, inquiry and update messages are not shown. `inXn` refers to “intersection”, `carInXn` to the concept of a car in an intersection. The `whyStopped` agent is at the top because it sends enquiries down to the `giveWay` agents. The `history` agent receives input from all other agents.

#### 6.4.1 Dynamic Network Implementation and Results

The dynamic network has been implemented in the language Java—a high-level procedural object-oriented language that incorporates threads (concurrency). Using a recursive 3D filter eliminated the problem of determining velocity through token correspondence as in stage one, but because the trajectories returned were inconsistent (they jumped around a lot), processing was required in the car agent to create reasonable car trajectories. For instance, a trajectory could shift from one location to another in one frame, or alternatively, terminate and be replaced by another with a different name. In the car agent, we have incorporated an analogue of the old traj agent to tie these inconsistent trajectories together, creating two new trajectories from one input, or splitting one trajectory into two.

As in stage one SOO-PIN, uncertainty was handled using the Dempster-Shafer (DS) evidential reasoning, with belief values being passed around the network as pairs of support and plausibility. DS works very well in the dynamic context, as consistent data over several frames rapidly builds confidence in the interpretation using the usual evidence combination formulas.

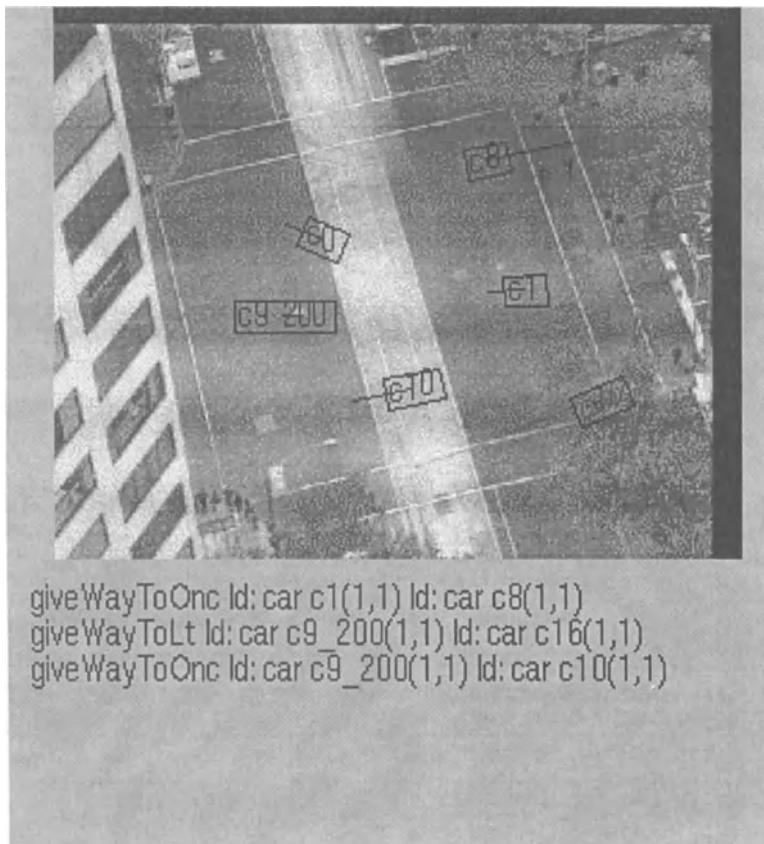
One frame from our system when running the output of the 3D recursive filter is shown in Figure 6.13, where the Java generated window shows the output data overlaid on an image of the empty intersection. The data is shown as rectangles containing car identities with lines indicating the car velocities. Below the image is the instantaneous give-way relationships found by the system for that frame.

The output from the history agent for one car (car c9\_200) is shown below:

```

Car c9_200 (1,1)
Frame 200: Car entering scene(1,1)
Frame 202: Car going right(1,1)
Frame 202: Giving way to oncoming car c10(1,1)
Frame 202: Giving way to oncoming car c9(1,1)
Frame 212: Giving way to left-turning car c16(1,1)
Frame 233: car stopped, giveWay To Left car c16(1,1)
Frame 245: car stopped, blocked by c1(1,1)
Frame 247: car stopped, giveWay To Oncoming car c11(1,1)
Frame 254: car stopped, giveWay To Oncoming car c2(1,1)
Frame 273: car stopped, giveWay To Oncoming car c12(1,1)
Frame 297: car stopped, giveWay To Oncoming car c13(1,1)
Frame 315: car stopped, giveWay To Oncoming car c14(1,1)
Frame 318: car stopped, giveWay To Oncoming car c3(1,1)
Frame 344: car stopped, giveWay To Oncoming car c15(1,1)
Frame 372: car stopped, giveWay To Oncoming car c4(1,1)

```



```
giveWayToOnc Id: car c1(1,1) Id: car c8(1,1)  
giveWayToLt Id: car c9_200(1,1) Id: car c16(1,1)  
giveWayToOnc Id: car c9_200(1,1) Id: car c10(1,1)
```

Figure 6.13: One frame's output from Java processed velocity data, showing cars represented as rectangles containing identity codes, with velocity lines. Below the image is the instantaneous give-way relationships found by the system for that frame.

This shows the car entering the scene and beginning to turn right. When it stops moving, the `whyStopped` agent becomes active, which then attempts to establish why it is stopped as the scenario unfolds. The sequence ends before the car moves away. This output can be viewed at URL [http://www.cs.mu.OZ.AU/~sandy/inXn\\_out4.mpg](http://www.cs.mu.OZ.AU/~sandy/inXn_out4.mpg).

Not counting the time taken to perform the low-level trajectory determination, the high-level interpretation took 96.25 cpu seconds and 326 elapsed seconds on a Sun SparcStation II to analyse the 450 frame sequence, i.e., about 0.21 cpu seconds per frame. It can be seen that the Java system runs much faster than the earlier stage one of SOO-PIN.

## 6.5 Conclusion

We have presented two stages of our work in concurrent, hierarchical agent-based systems for high-level vision based on concepts outlined by Minsky, Hewitt and Schank. It has been implemented in Parlog++ and demonstrated in a real world traffic scenario. The experiments show that this architecture is a natural and successful approach to high-level vision, not only as demonstrated by the speed of the system at run time, but also from our experience in building the system – the high-level modular object-oriented approach simplified maintenance and enhancement, and logic programming naturally handled symbolic processing and pattern matching. Moreover, this approach facilitated our philosophy that many spatial predicates cannot be resolved purely symbolically, but need to be dealt with in a combined symbolic-numeric fashion.

Further, we have demonstrated our initial work dealing with interpreting long image sequences—which necessitated extending the nature and typology of the our concept-frames to deal with **time**, especially distinguishing between **instantaneous**, **continuous** and **durational** agents. This work has been based partly on the concerns raised by van Gelder and Port on the adequacy of the computational paradigm to deal with dynamic systems, and our need to work with the practical and theoretical devices at our disposal—computers and symbolic processing. The resulting system, written in Java, successfully interpreted the significant events in a 450 frame sequence of traffic images. The interpretations were based on:

- car activities,
- give-way events between cars,
- explanations for why they were stopped.

The Dempster-Shafer belief values passed around the network, while not being used as control mechanisms within the system, vindicated themselves in that interpretations that humans regarded as correct tended to have higher support, as shown in Section 6.3.7. Future work involves extending the dynamic interpretation to deal with more concepts and more explanatory power, and is also concerned with extending the agent-based architecture to incorporate a low-level movement detection system.

# Chapter 7

## Invariance Signatures for Two-Dimensional Contours

David McG. Squire and Terry Caelli

### Abstract

Invariant pattern recognition is an important problem in many areas of computer vision. In this chapter, a new invariant feature of two-dimensional contours is introduced: the Invariance Signature (IS). The IS is a measure of the degree to which a contour is invariant under a variety of transformations, derived from the theory of Lie transformation groups. It is shown that a Model-Based Neural Network (MBNN) [54, 321] can be constructed which computes the IS of a contour, and classifies patterns on this basis. MBNNs, whilst retaining the structure and advantages of traditional neural networks (TNNs), enable explicit modelling of the target system. This can result in greatly improved generalisation, and representation in lower-dimensional state spaces. MBNNs can be trained with much smaller training sets than are required by TNNs. This means that MBNNs are much less computationally-expensive to train than TNNs. Experiments demonstrate that such Invariance Signature networks can be employed successfully for shift-, rotation- and scale-invariant optical character recognition.

## 7.1 Introduction

The aim of any invariant pattern recognition technique is to obtain a representation of the pattern in a form that is invariant under some specified transformations of the original image. Ideally, such a technique would produce a representation of the pattern which was not only invariant, but which also uniquely characterised the input pattern.

This goal is not necessarily realisable, nor is it always as desirable as it might seem. In many pattern recognition problems, the aim is to produce a system which classifies input patterns as belonging to a particular *class*, rather than to identify every input pattern presented uniquely. In such cases a unique representation for each possible input pattern can actually be a disadvantage. What is required is an invariant representation that retains enough information for distinct classes to be distinguished. It is by no means necessary that members of the same class be distinguishable in the invariant representation used. Indeed, if all members of a class are identical in the invariant representation, this can greatly reduce the size of the training set required by many recognition algorithms. The Invariance Signature technique presented in this chapter provides a means of realizing this goal. Before describing it in detail, some of the major alternative techniques for invariant pattern recognition will be reviewed.

### 7.1.1 Methods for Invariant Pattern Recognition

The ability to perceive the permanent features of the visual environment is something which humans take for granted. Indeed, it is hard to imagine a world in which we could not do so. We do not even notice that we recognise objects and patterns independently of changes in lighting conditions, shifts of the object or observer, or changes in orientation and scale. Gibson [121] stated this rather well:

It can be shown that the easily measured variables of stimulus energy, the intensity of light, sound, odour, and touch, for example, vary from place to place and from time to time as the individual goes about his business in the environment. The stimulation of receptors and the presumed sensations, therefore, are variable and changing in the extreme, unless they are experimentally controlled in a laboratory. The unanswered question of sense perception is how an observer, animal or human, can obtain constant perceptions in everyday life on the basis of these continually changing sensations. For the fact is that animals and men do perceive and respond to the permanent features of the environment as well as to the changes in it . . . The hypothesis is that constant perception depends on the ability of the individual to detect the invariants, and that he ordinarily pays no attention whatever to the flux of changing sensations. [355, p. 96]

In this chapter we are concerned with only a small subset of the problems outlined above: the invariant perception of two-dimensional patterns under shift, rotation and scaling in the plane. This corresponds to the ability of humans to recognise patterns such as typed or handwritten characters independently of their size, orientation or position, which they do unthinkingly when reading a document such as an architectural drawing.

#### 7.1.1.1 Integral Transforms

An approach that has long been used in invariant pattern recognition is to seek an integral transform of the image such that the image representation in the transform domain is invariant under the some specified transformations. Such transforms frequently have two components: one which is invariant under the application of a specified transformation to the image, and another which encodes the transformation. Most such techniques are based upon the Fourier transform. In the complex Fourier transform domain, the amplitude spectrum is invariant under shifts of the image, and the shift is encoded in the phase spectrum. The Fourier and Mellin transforms, in various guises and combinations, form the basis of many invariant pattern recognition techniques [7, 53, 105, 106, 183, 297, 306].

#### 7.1.1.2 Moments

Rather than transforming the image, integrals can also be used to compute geometrical moments of the image. Certain combinations of the moments are invariant under transformations of the original image [155]. Wechsler [355], however, notes that these moments are not fault-tolerant, and in general produce disappointing results when used in pattern recognition experiments. Better results have been obtained using alternative sets of moments, such as the Zernike moments [166].

#### 7.1.1.3 Cross-Correlation

Matched Filtering is one of the oldest techniques for pattern recognition [18, 293]. Combining it with convolution allows it to be used for invariant pattern recognition. Also known as cross-correlation, this effectively involves exhaustively-searching the image for all possible transformed versions of the pattern that is to be recognised. Some version of cross-correlation plays a role in many invariant pattern recognition techniques [7, 53, 179, 183, 233, 257, 379].

#### 7.1.1.4 Parts and Relationships

In the above techniques, an attempt is made to find an invariant representation of the image as a whole, and to match images to images (or

transformed images) on a pixel-by-pixel basis. A different approach to the invariant pattern recognition problem is to view an image as a group of component *parts*, and *relationships* between those parts. As a trivial example, if an image consists of two straight lines, then the angle between those lines is invariant under shifts, rotations and scalings of the image. The lines are the parts, and the angle between them is their relationship. In “parts and relationships” techniques, matching is done between these abstracted properties of the image, rather than between pixels. These techniques require some form of sub-graph matching [181], which is known to be an NP-complete problem [18]. The challenge is thus to find an algorithm that can obtain an acceptable solution in reasonable time. Explicit search may be satisfactory for sufficiently small graphs. Another approach is relaxation labelling, which is employed by Li [181].

### 7.1.1.5 Contour-Based Methods

An important class of two-dimensional invariant pattern recognition techniques, especially in the context of this chapter, are those designed for the recognition of contours. Contours may be extracted through some form of edge detection, or may be a natural pattern representation, such as in character recognition. Broadly speaking, approaches to contour recognition may be divided into two classes: those which represent the contour by a parameterised algebraic expression fitted to the image, and those which treat a contour as a group of pixels. Contour recognition is of particular interest because contours corresponding to object boundaries are frequently used in three-dimensional object recognition, and also because there are many applications in which patterns naturally consist of line drawings (*e.g.* character recognition, circuit diagrams, engineering drawings, *etc.*). Moreover, there is evidence that the human visual system applies a contour-based approach to pattern recognition even when no contour exists in the image: an implied contour is interpolated [51].

**7.1.1.5.1 The Hough Transform** Perhaps the simplest contour-based technique is the Hough transform, which is described in standard computer vision references [18]. Variants of the Hough transform occur frequently in the invariant pattern recognition literature [65, 182, 257]. The Hough transform and its generalisations can be interpreted as cross-correlation techniques, where the template is a parametric curve rather than an image [257]. It can be used to discover the set of parameters that best explain the image data, given that the parametric form of the curve producing the data is known *a priori*. Its disadvantage is that it is restricted in practice to a reasonably small set of template curves and allowable transformations, since its computational expense increases exponentially with the number of parameters.

**7.1.1.5.2 Algebraic and Differential Invariants** Another approach to invariant contour matching involves calculating invariants of the contour. This offers the chance to avoid the computation time and space expenses of methods such as cross-correlation or the Hough transform, in which the contour is matched against all possible transformed versions of the template. Algebraic invariants are well-suited for use with algebraic contours: contours which can be expressed by an implicit polynomial  $f(x, y) = 0$ . Conic sections form a family of such contours. The general equation for a conic section can be written succinctly in matrix form,  $\mathbf{X}^T \mathbf{A} \mathbf{X} = 0$ . The shape of a particular conic is determined by the matrix  $\mathbf{A}$ , which is real symmetric. The coordinates can be transformed using a similarity transform so that  $\mathbf{A}$  is diagonal. This implies that properties of the matrix  $\mathbf{A}$  which are invariant under similarity transforms will be invariant descriptors of the shape of the conic under translation and rotation. One such feature is the determinant  $D$ , another is the trace  $T$ . In fact, any symmetric function of the eigenvalues of  $\mathbf{A}$  is an invariant. The matrix  $\mathbf{A}$  must be obtained by fitting a polynomial to the image data – in itself a far from trivial problem, and potentially computationally-expensive. A high resolution image of the curve is required if the coefficients are to be estimated sufficiently well to be useful for recognition. Once this is done, however, the matching process is very cheap.

Differential Invariants arise most naturally when the coordinates of points on a curve,  $\mathbf{x}$ , are expressed explicitly as a function of some local parameter  $t$ ,  $\mathbf{x} = \mathbf{x}(t)$ , rather than by an implicit function such as that above. The natural shape descriptors in such a representation are the derivatives  $\frac{d^n x_i}{dt^n}$ . These descriptors are *local*, since they depend on the derivatives at a particular value of  $t$ , unlike the global descriptors derived from the coefficients of the implicit function in the case of algebraic invariants. A differential invariant is a function of the derivatives  $\frac{d^n x_i}{dt^n}$  which does not change under a transformation of the coordinates  $\mathbf{x}$  and the parameter  $t$ . Various differential invariants have been widely applied in computer vision: curvature, torsion and Gaussian curvature, for instance, are all invariant under Euclidean transformations [115].

One advantage of differential invariants is that they are complete – a small set of invariants contains all the essential information about the curve. They are also local, in that differential properties at one point determine the entire curve. This means that differential invariants are invulnerable to occlusion. Whilst mathematically elegant, this approach has one great disadvantage. Its application to digital images requires the computation of extremely high-order derivatives of the contour in the image (as high as the eighth). This process is well-known to be error-prone. Moreover, these derivatives are raised to high powers, magnifying the estimation error.

## 7.2 Lie Transformation Groups and Invariance

One approach to invariant pattern recognition is to study the ways in which local features change under the action of global transformations of the image. Such considerations lead naturally to the study of Lie transformation groups, and many, varied invariant pattern recognition techniques make use of the Lie theory [72, 105, 139, 140, 297, 306, 322].

In this section the theory of Lie transformation groups will be developed, and their significance for invariant pattern recognition made clear. A new shift-, rotation- and scale- invariant function of a two-dimensional contour with respect to a given Lie transformation group, the *Invariance Measure Density Function*, will then be derived. It will be shown that several such functions can be combined to yield an *Invariance Signature* for the contour. This Invariance Signature has several properties that make it attractive for implementation in an MBNN: it is based on local properties of the contour, so initial calculations are inherently parallel; it is statistical in nature, and its resolution can be chosen at the designer's discretion, allowing direct control over the dimensionality of the network implementation. However, the use of the Invariance Signature is by no means limited to neural network implementations. Whilst patterns are not uniquely represented by the Invariance Signature, it will be shown in Section 7.4 that *classes* of patterns are represented sufficiently differently for optical character recognition applications.

### 7.2.1 Definition of a Group

A group is a set of elements and an associated operation which maps elements of the group into each other. Consider a set of elements  $G$  and an associated operation  $\oplus$ . Let the elements of  $G$  be denoted by  $\epsilon_1, \epsilon_2, \dots$ . For  $G$  to be a group, it must have the following properties:

1. Closure:

$$\forall \epsilon_i, \epsilon_j \in G, \quad \epsilon_i \oplus \epsilon_j \in G \quad (7.1)$$

2. Associativity:

$$\forall i, j, k, \quad (\epsilon_i \oplus \epsilon_j) \oplus \epsilon_k = \epsilon_i \oplus (\epsilon_j \oplus \epsilon_k) \quad (7.2)$$

3. Identity Element  $I$ :

$$\exists I : \forall \epsilon \in G, \quad \epsilon \oplus I = \epsilon \quad (7.3)$$

4. Inverse Elements:

$$\forall \epsilon \in G, \quad \exists \epsilon^{-1} : \quad \epsilon \oplus \epsilon^{-1} = I \quad (7.4)$$

### 7.2.2 One Parameter Lie Groups in Two Dimensions

A Lie group is a continuous transformation group with a differentiable structure. For two-dimensional invariant pattern recognition, the most interesting groups are the one-parameter Lie transformation groups defined on the plane. These transformation groups include rotation, dilation and translation. These are smooth transformations of the form

$$\begin{aligned} x' &= \mu(x, y, \alpha) \\ y' &= \nu(x, y, \alpha). \end{aligned} \quad (7.5)$$

The parameter  $\alpha$  determines which element of the group the transformation is. For instance, if  $\alpha_0$  corresponds to the identity element, we have

$$\begin{aligned} x' &= \mu(x, y, \alpha_0) = x \\ y' &= \nu(x, y, \alpha_0) = y. \end{aligned} \quad (7.6)$$

There is a vector field  $\vec{g} = [ g_x \ g_y ]^T$  associated with each Lie group  $G$ . This vector field gives the direction in which a point  $(x, y)$  is “dragged” by an infinitesimal transformation under the action of the group. It is given by

$$\begin{aligned} g_x(x, y) &= \frac{\partial \mu}{\partial \alpha} \Big|_{\alpha=\alpha_0} \\ g_y(x, y) &= \frac{\partial \nu}{\partial \alpha} \Big|_{\alpha=\alpha_0}. \end{aligned} \quad (7.7)$$

This vector field  $\vec{g}$  allows an operator  $\mathcal{L}_G$  to be defined,

$$\mathcal{L}_G = g_x \frac{\partial}{\partial x} + g_y \frac{\partial}{\partial y}. \quad (7.8)$$

The operator  $\mathcal{L}_G$  is called the *generator* of the transformation group  $G$ , because it can be used to construct the finite transformation corresponding to the infinitesimal dragging described in Equations 7.7. This is done by “summing” the repeated applications of the infinitesimal transformation.

### 7.2.3 From Infinitesimal to Finite Transformations

Let us consider the case in which we know the direction of the vector field specifying the infinitesimal transformation at each point, as given by Equations 7.7, and we wish to construct the Equations 7.5 specifying the finite transformation. We will consider the transformation of  $x$  in detail. For a small change in the group parameter from the identity element,  $\alpha = \alpha_0 + \Delta\alpha$ , we can approximate the change in  $x$  by

$$x' = x + \Delta x \approx x + \Delta\alpha \frac{\partial \mu}{\partial \alpha} \Big|_{\alpha=\alpha_0}. \quad (7.9)$$

We now wish to find a finite transformation corresponding to  $n$  applications of the  $\Delta\alpha$  transformation. This will approximate the finite transformation corresponding to the group element specified by parameter  $\alpha = n\Delta\alpha$ . Let  $x_i$  correspond to the value of  $x'$  after  $i$  applications of the transformation specified by  $\Delta\alpha$ . We obtain

$$\begin{aligned}x_0 &= x \\x_1 &= x_0 + \frac{\alpha}{n} \mathcal{L}_G x_0 = \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right) x \\x_2 &= x_1 + \frac{\alpha}{n} \mathcal{L}_G x_1 \\&= \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right) x_1 \\&= \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right)^2 x\end{aligned}\tag{7.10}$$

and thus

$$x_n = \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right)^n x.$$

In the limit as  $n \rightarrow \infty$ , the approximation becomes exact, and the finite transformation is given by

$$\mu(x, y, \alpha) = \lim_{n \rightarrow \infty} \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right)^n x.\tag{7.11}$$

Similarly,

$$\nu(x, y, \alpha) = \lim_{n \rightarrow \infty} \left(1 + \frac{\alpha}{n} \mathcal{L}_G\right)^n y.\tag{7.12}$$

#### 7.2.4 Derivation of the Rotation Transformation

As an example, we will derive the rotation transformation from the knowledge only that each point  $P$  should be dragged in a direction at right angles to the line from the origin to  $P$  by the infinitesimal transformation, as shown in Figure 7.1. We will denote the rotation group  $R$ . The parameter of the group is  $\theta$ . From Figure 7.1, we see that

$$\begin{aligned}dx &= -rd\theta \sin \phi \\&= -\sqrt{x^2 + y^2} d\theta \frac{y}{\sqrt{x^2 + y^2}} \\&= -yd\theta.\end{aligned}\tag{7.13}$$

Similarly,

$$dy = xd\theta.\tag{7.14}$$

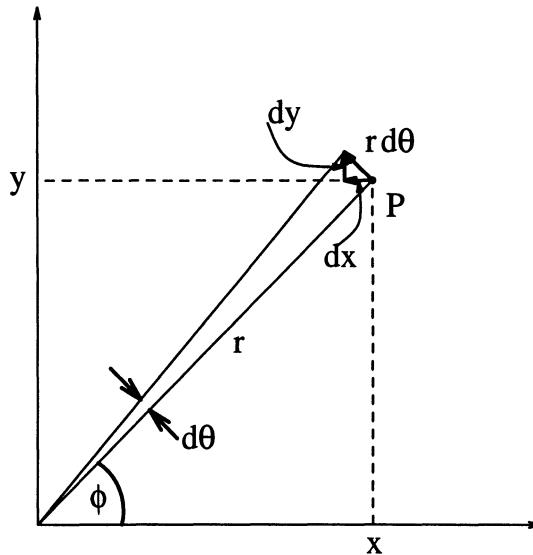


Figure 7.1: Infinitesimal transformation leading to rotation.

The generator of the rotation group  $R$  is thus

$$\mathcal{L}_R = -y \frac{\partial}{\partial x} + x \frac{\partial}{\partial y}. \quad (7.15)$$

Consequently, the corresponding finite transformation for  $x$  is

$$\mu(x, y, \theta) = \lim_{n \rightarrow \infty} \left( 1 + \frac{\theta}{n} \mathcal{L}_R \right)^n x. \quad (7.16)$$

Using the binomial expansion,

$$\begin{aligned} \mu(x, y, \theta) &= \lim_{n \rightarrow \infty} \left[ 1 + n \frac{\theta}{n} \mathcal{L}_R + \frac{n(n-1)}{2!} \frac{\theta^2}{n^2} \mathcal{L}_R^2 \right. \\ &\quad \left. + \frac{n(n-1)(n-2)}{3!} \frac{\theta^3}{n^3} \mathcal{L}_R^3 + \dots \right] x. \end{aligned} \quad (7.17)$$

We note that

$$\lim_{n \rightarrow \infty} \frac{n!}{(n-k)! n^k} = 1, \quad (7.18)$$

and

$$\begin{aligned} \mathcal{L}_R x &= -y \\ \mathcal{L}_R^2 x &= -x \\ \mathcal{L}_R^3 x &= y \\ \mathcal{L}_R^4 x &= x, \end{aligned} \quad (7.19)$$

after which the values repeat. We can therefore write

$$\begin{aligned}\mu(x, y, \theta) &= \left[ x - \theta y - \frac{\theta^2}{2!} x + \frac{\theta^3}{3!} y + \frac{\theta^4}{4!} x + \dots \right] \\ &= x \left( 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \dots \right) - y \left( \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots \right) \quad (7.20) \\ &= x \cos \theta - y \sin \theta.\end{aligned}$$

Similarly,

$$\nu(x, y, \theta) = x \sin \theta + y \cos \theta. \quad (7.21)$$

These are the well-known equations describing a rotational transformation by an angle  $\theta$ . The procedure above can be used to generate the finite transformation corresponding to any Lie operator.

### 7.2.5 Functions Invariant Under Lie Transformations

A function is said to be invariant under the action of a transformation if all points of the function are mapped into other points of the function by the action of the transformation on the coordinate system. Consider a function  $F(x, y)$ . We wish to determine its invariance with respect to a Lie transformation group  $G$ . Let

$$\vec{g}(x, y) = [ g_x(x, y) \quad g_y(x, y) ]^T \quad (7.22)$$

be the vector field corresponding to the generator of the Lie group,  $\mathcal{L}_G$ .  $F$  is constant with respect to the action of the generator if

$$\mathcal{L}_G F = 0. \quad (7.23)$$

This can be written in terms of the vector field as

$$\nabla F \cdot \vec{g}(x, y) = 0, \quad (7.24)$$

that is

$$\frac{\frac{\partial F}{\partial x}}{\frac{\partial F}{\partial y}} = -\frac{g_y}{g_x}. \quad (7.25)$$

Now consider a contour  $C$  parameterised by  $t$  specified by the implicit function

$$\forall t \quad F(x(t), y(t)) = K. \quad (7.26)$$

Since  $F$  is constant on the contour, we have

$$\frac{dF}{dt} = \frac{\partial F}{\partial x} \frac{dx}{dt} + \frac{\partial F}{\partial y} \frac{dy}{dt} = 0, \quad (7.27)$$

and consequently

$$\frac{\frac{\partial F}{\partial x}}{\frac{\partial F}{\partial y}} = - \frac{dy}{dx}. \quad (7.28)$$

Thus the condition for invariance with respect to the Lie transformation group generated by  $\mathcal{L}_G$ , given in Equation 7.24, holds if

$$\frac{dy}{dx} = \frac{g_y}{g_x}. \quad (7.29)$$

on the contour.

The condition derived in Equation 7.29 has a very natural interpretation. It says that a contour is invariant under the action of a group  $G$  if the tangent to the contour at each point is in the same direction as the vector field  $\vec{g}$  corresponding to the infinitesimal transformation that generates the group.

### 7.2.6 From Local Invariance Measures to Global Invariance

We now propose a new shift-, rotation- and dilation-invariant signature for contours. We call this an *Invariance Signature*, since it is derived from the degree to which a given contour is consistent with invariance under a set of Lie transformation groups.

### 7.2.7 The Local Measure of Consistency

We have seen in Equation 7.29 that in order for a contour  $C$  to be invariant under a transformation group  $G$  the tangent to the contour must be everywhere parallel to the vector field defined by the generator of the group. We now define the *Local Measure of Consistency* with invariance under a transformation group  $G$  at a point  $(x, y)$  on  $C$ ,  $\iota_G(x, y)$ :

$$\iota_G(x, y) = \left| \hat{\theta}(x, y) \cdot \hat{g}_G(x, y) \right|. \quad (7.30)$$

The absolute value is used because only the orientation of the tangent vector is significant, not the direction. At each point both the tangent vector to the contour,  $\vec{\theta}(x, y)$  and the vector field  $\vec{g}_G(x, y)$  are normalised:

$$\hat{g}_G(x, y) = \frac{g_x(x, y)\hat{i} + g_y(x, y)\hat{j}}{\sqrt{g_x^2(x, y) + g_y^2(x, y)}} \quad (7.31)$$

and

$$\hat{\theta}(x, y) = \frac{\hat{i} + \frac{dy}{dx}\hat{j}}{\sqrt{1 + \left(\frac{dy}{dx}\right)^2}}, \quad (7.32)$$

where  $\hat{i}$  and  $\hat{j}$  are unit vectors in the  $x$  and  $y$  directions respectively. Substituting Equations 7.31 and 7.32 in Equation 7.30, we obtain

$$\iota_G(x, y) = \frac{1}{\sqrt[+]{1 + \left[ \frac{g_y(x, y) - g_x(x, y) \frac{dy}{dx}}{g_x(x, y) + g_y(x, y) \frac{dy}{dx}} \right]^2}}. \quad (7.33)$$

### 7.2.8 The Invariance Measure Density Function

Equation 7.30 is a mapping  $C \mapsto [0, 1]$ , which gives a measure of the degree to which the tangent at each point is consistent with invariance under  $G$ . We wish to find a function which characterises the degree to which the entire contour  $C$  is consistent with invariance under  $G$ . Such a function is the density function for the value of  $\iota_G$  in  $[0, 1]$ ,  $I(\iota_G)$ , which we will call the *Invariance Measure Density Function*. The more points from  $C$  that are mapped to values close to 1 by Equation 7.30, the more consistent  $C$  is with invariance under  $G$ . The shape of  $I(\iota_G)$  is a descriptor of  $C$ , and we will show that  $I(\iota_G)$  is invariant under rotations and dilations of  $C$  (see Theorem 7.1). Translation invariance is obtained if the origin of coordinates in which  $\iota_G$  is calculated is chosen to be the centroid of  $C$ .

It is interesting to note that there is evidence from psychophysical experiments that a measure of the *degree of invariance* of a pattern with respect to the similarity group of transformations (rotations, translations and dilations) is important in human pattern recognition [52]. The measure proposed here might be seen as a mathematical formalisation of this notion. Moreover, its implementation in a neural network architecture is consistent with Caelli and Dodwell's statement [52, p. 159] of a proposal due to Hoffman [139, 140]:

Hoffman's fundamental postulate was that the coding of orientation at various positions of the retinotopic map by the visual system, discovered by Hubel and Wiesel [145] and others, actually provides the visual system with "vector field" information. That is, the visual system, on detecting specific orientation and position states ("Φ/P codes"), spontaneously extracts the path curves (interpreted as visual contours) of which the local vectors are tangential elements.

First, however, we must establish the form of  $I(\iota_G)$ . Without loss of generality, consider  $C$  to be parameterised by  $t : t \in [t_0, T]$ . The arc length  $s$  along  $C$  to a given value of the parameter  $t$  is:

$$s(t) = \int_{t_0}^t \sqrt{\left( \frac{dx}{dt} \Big|_\tau \right)^2 + \left( \frac{dy}{dt} \Big|_\tau \right)^2} d\tau. \quad (7.34)$$

The total length of  $C$  is thus  $S = s(T) = \oint_C ds$ . For well-behaved functions  $F(x, y)$ , we can construct  $s(t)$  such that we can reparameterise  $C$  in terms of

$s$ . Thus we can rewrite Equation 7.30 to give  $\iota_G$  in terms of  $s$ . For simplicity, we will first consider the case in which  $\iota_G$  is a monotonic function of  $s$ , as shown in Figure 7.2.

The Invariance Measure Density is:

$$\begin{aligned} I(\iota_G) &= \lim_{\Delta \iota_G \rightarrow 0} \left| \frac{\Delta s}{S \Delta \iota_G} \right| \\ &= \frac{1}{S} \left| \frac{ds}{d\iota_G} \right|. \end{aligned} \quad (7.35)$$

$I(\iota_G)$  can be interpreted as the probability density function for  $\iota_G$  at points  $(x(s), y(s))$ , where  $s$  is a random variable uniformly distributed on  $[0, S]$ . It is clear that for the general case, in which  $\iota_G$  is not a monotonic function of  $s$ , the function could be broken up into piecewise monotonic intervals, and the contribution to the density from each interval summed. The general form for a specific value  $\iota_G'$  is therefore

$$I(\iota_G') = \frac{1}{S} \sum_{s \in [0, S] : \iota_G(s) = \iota_G'} \left| \frac{ds}{d\iota_G} \right|. \quad (7.36)$$

**Theorem 7.1** *The Invariance Measure Density,  $I(\iota_G)$ , is invariant under translations, rotations and dilations of the contour  $C$  with respect to which it is calculated.*

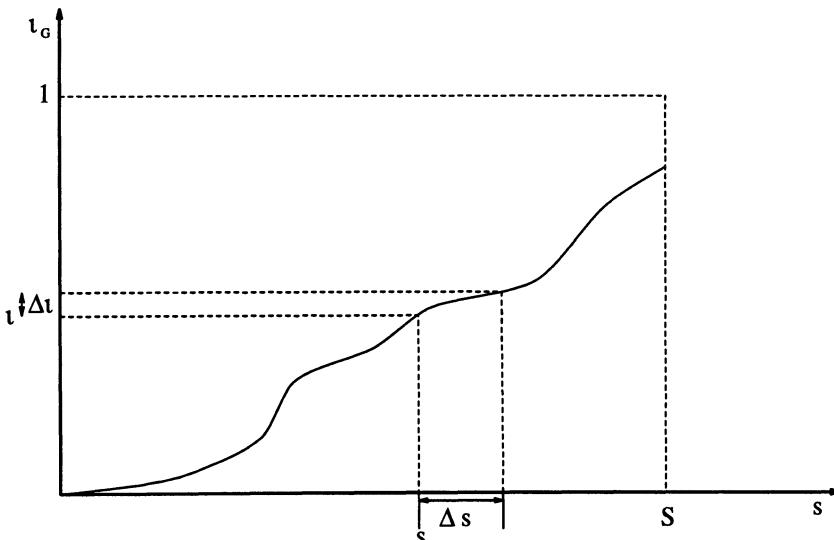


Figure 7.2: Local Measure of Consistency as a function of arc length.

**Proof** That  $I(\iota_G)$ , defined in Equation 7.36, is invariant under translations of the contour  $C$  is trivial, since, as defined in Section 7.2.8,  $\iota_G$  is calculated with the origin at the centroid of the contour  $C$ . Rotation and dilation invariance can be proved by construction. Since the transformations for rotation and dilation in the plane commute,<sup>1</sup> we can consider a two-parameter Abelian group corresponding to a rotation by an angle  $\theta$  and a dilation by a positive factor  $\beta$ . The coordinates  $x$  and  $y$  are transformed according to

$$\begin{aligned}x' &= \beta(x \cos \theta - y \sin \theta) \\y' &= \beta(x \sin \theta + y \cos \theta).\end{aligned}\tag{7.37}$$

We are interested in the relationship between the arc length function  $s(t)$  defined in Equation 7.34 for the original parameterised contour  $(x(t), y(t))$  and the new function  $s'(t)$  when the coordinates are transformed according to Equation 7.37. We find that

$$\begin{aligned}\frac{dx'}{dt} &= \beta \cos \theta \frac{dx}{dt} - \beta \sin \theta \frac{dy}{dt} \\ \frac{dy'}{dt} &= \beta \sin \theta \frac{dx}{dt} + \beta \cos \theta \frac{dy}{dt}.\end{aligned}\tag{7.38}$$

Combining these, we find that

$$\begin{aligned}\left(\frac{dx'}{dt}\right)^2 + \left(\frac{dy'}{dt}\right)^2 &= \beta^2 \left[ \cos^2 \theta \left(\frac{dx}{dt}\right)^2 \right. \\ &\quad - 2 \cos \theta \sin \theta \frac{dx}{dt} \frac{dy}{dt} + \sin^2 \theta \left(\frac{dy}{dt}\right)^2 \\ &\quad \left. + \sin^2 \theta \left(\frac{dx}{dt}\right)^2 + 2 \cos \theta \sin \theta \frac{dx}{dt} \frac{dy}{dt} \right. \\ &\quad \left. + \cos^2 \theta \left(\frac{dy}{dt}\right)^2 \right] \\ &= \beta^2 \left[ \left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 \right].\end{aligned}\tag{7.39}$$

This result can be substituted into Equation 7.34 to obtain

$$s'(t) = \beta s(t).\tag{7.40}$$

---

<sup>1</sup>As a consequence of Clairaut's theorem, they commute when the functions to which the transformations are applied are both defined and continuous in a neighbourhood around each point of the function.

This indicates that the total arc length is  $S' = \beta S$ , and the derivative of  $s(t)$  is also scaled. Substituting into Equation 7.36, we obtain

$$\begin{aligned}
I'(\iota_{G'}) &= \frac{1}{S'} \sum_{\iota_G(s)=\iota_{G'}} \left| \frac{ds'}{d\iota_G} \right| \\
&= \frac{1}{\beta S} \sum_{\iota_G(s)=\iota_{G'}} \beta \left| \frac{ds}{d\iota_G} \right| \\
&= \frac{1}{S} \sum_{\iota_G(s)=\iota_{G'}} \left| \frac{ds}{d\iota_G} \right| \\
&= I(\iota_{G'}) \\
&\quad Q.E.D.
\end{aligned} \tag{7.41}$$

Thus we have demonstrated that  $I(\iota_G')$  is invariant under rotations and dilations of the contour  $C$ .

#### 7.2.8.1 Invariance Measure Densities For Specific Contours

In order to demonstrate how the Invariance Measure Density defined above can be applied, we will evaluate  $I(\iota_G)$  for a specific contour.

**7.2.8.1.1 The Square** Let the contour  $C$  be a square of side  $2L$  centred at the origin, as shown in Figure 7.3. We will find the Invariance Measure Density for  $C$  with respect to rotation,  $I_{C_{rot}}(\iota)$ . By symmetry, we need only find  $I_{C_{rot}}$  for one side of the square. On the side indicated by the dashed line in Figure 7.3,  $x$  and  $y$  can be expressed in terms of a parameter  $t$  as

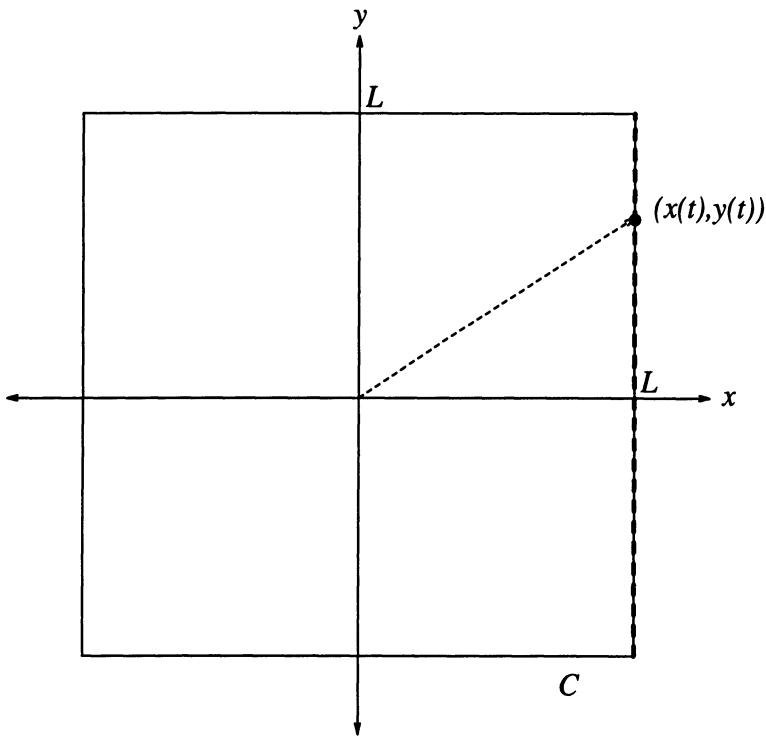
$$\begin{aligned}
x &= L \\
y &= t, \quad -L \leq t \leq L.
\end{aligned} \tag{7.42}$$

Here the arc length  $s(t) = t + L$ , and the total is  $S = 2L$ . If  $\dot{x}(t)$  and  $\dot{y}(t)$  are the derivatives of  $x$  and  $y$  with respect to  $t$ , Equation 7.33 can be rewritten as

$$\iota_{C_{rot}}(s) = \frac{1}{\sqrt{1 + \left[ \frac{g_y(s)\dot{x}(s) - g_x(s)\dot{y}(s)}{g_x(s)\dot{x}(s) + g_y(s)\dot{y}(s)} \right]^2}}. \tag{7.43}$$

Here we have  $\dot{x}(t) = 0$  and  $\dot{y}(t) = 1$ . Equations 7.15 and 7.42 can be substituted into Equation 7.43 to give

$$\iota_{C_{rot}}(s) = \frac{1}{\sqrt{1 + \left( \frac{s-L}{L} \right)^2}}, \tag{7.44}$$

Figure 7.3: Square of side  $2L$ .

which can be inverted to yield

$$s = L \left( 1 + \sqrt{\frac{1}{\iota_{C_{rot}}^2} - 1} \right); \quad (7.45)$$

differentiating,

$$\frac{ds}{d\iota_{C_{rot}}} = \frac{-L}{\iota_{C_{rot}}^2 (1 - \iota_{C_{rot}}^2)}. \quad (7.46)$$

Using Equation 7.36, we arrive at our final result:

$$\begin{aligned} I_{C_{rot}}(\iota_{C_{rot}}) &= \frac{1}{2L} \sum_{\iota'_{C_{rot}}=\iota_{C_{rot}}} \left| \frac{ds}{d\iota_{C_{rot}}} \right|_{\iota'_{C_{rot}}} \\ &= \frac{1}{2L} \times 2 \times \frac{L}{\iota_{C_{rot}}^2 (1 - \iota_{C_{rot}}^2)} \\ &= \frac{1}{\iota_{C_{rot}}^2 (1 - \iota_{C_{rot}}^2)}, \quad \iota_{C_{rot}} \in \left[ \frac{1}{\sqrt{2}}, 1 \right]. \end{aligned} \quad (7.47)$$

Note that, as required, the scale of the square  $L$  does not appear in this result. The factor of 2 arises because  $\iota_{C_{rot}}$  is a symmetric function of  $s$ , so the sum has two terms. This function, shown in Figure 7.4, is characteristic of the square.

### 7.2.9 Invariance Space: Combining Invariance Measure Densities

We have seen how to compute the Invariance Measure Density Function for a contour  $C$  with respect to invariance under a Lie transformation group  $G$ . We now consider the case in which the Invariance Measure Density Function is calculated with respect to a number of groups, and the results combined to provide a more complete characterisation of the transformational properties of a contour  $C$ . This operation can be considered to be a mapping of each point from the two dimensional image space to the interior of a unit hypercube in an  $n$ -dimensional *invariance space*, where each of the  $n$  dimensions corresponds to a particular sort of invariance. Equation 7.48 shows this for the case of a three-dimensional invariance space, where the dimensions correspond to the Local Measure of Consistency  $\iota$  with respect

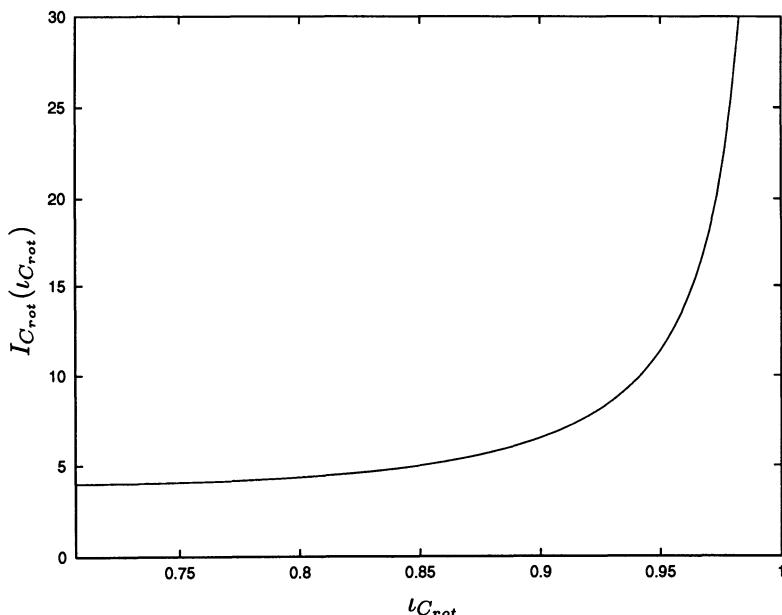


Figure 7.4: Invariance Density Measure with respect to rotation for a square.

to the transformations of rotation, dilation and translation:

$$(x, y) \mapsto [ \begin{array}{ccc} \iota_{rot} & \iota_{dil} & \iota_{trans} \end{array} ]^T. \quad (7.48)$$

The distribution of points in this invariance space is characteristic of the contour  $C$ . This particular three-dimensional invariance space is the one that will be used for the experimental application of Invariance Signatures. Since each of the component invariance measure densities is invariant, this  $n$ -dimensional Invariance Signature is invariant under rotations, dilations, translations and reflections of the input image.

#### 7.2.9.1 Vector Fields Corresponding to Rotation, Dilation and Translation Groups

The vector fields for the generators of the transformation groups for rotation, dilation and translation are given in normalised form. All can be derived using Equation 7.7: for rotation invariance

$$\vec{g}_{rot}(x, y) = \frac{1}{\sqrt{x^2 + y^2}} [ \begin{array}{cc} -y & x \end{array} ]^T, \quad (7.49)$$

and for dilation invariance

$$\vec{g}_{dil}(x, y) = \frac{1}{\sqrt{x^2 + y^2}} [ \begin{array}{cc} x & y \end{array} ]^T. \quad (7.50)$$

The translation invariance case is somewhat different. In fact, the term “translation invariance” is something of a misnomer. What is in fact measured in this case is the degree to which the contour is “linear”. The vector field used is constant for all  $(x, y)$ , and the direction is determined by finding the eigenvector corresponding to the dominant eigenvalue,  $\vec{e}_1$ , of the coordinate covariance matrix of all the points in the contour.<sup>2</sup> The direction of this eigenvector is the principal direction of the contour.

To place this in the same context as the previous two cases, the “translation invariance” factor is a measure of the degree to which the tangent at each point in the contour is consistent with a function invariant under translation in the direction of the vector  $\vec{e}_1$ . Since  $\vec{e}_1$  is calculated from the image each time it is required, this measure is invariant under rotations, dilations and translations of the image. The vector field for the translation invariance case is thus:

$$\vec{g}_{trans}(x, y) = [ \begin{array}{cc} e_{1x} & e_{1y} \end{array} ]^T \quad (7.51)$$

---

<sup>2</sup> $\vec{e}_1$  is chosen to be a unit vector.

### 7.2.9.2 Uniqueness

It should be noted that this representation of the image is not unique, unlike some previous integral transform representations [106]. The combination of individual Invariance Measure Densities into an Invariance Space does, however, increase the discriminant properties. As an example, removing two opposite sides of a square will not alter its rotation and dilation Invariance Signatures, but it will change the translation Invariance Signature. Likewise, a single straight line has the same translation Invariance Signature as any number of parallel straight lines, however they are spaced. The rotation and dilation Invariance Signatures, however, are sensitive to these changes.

### 7.2.10 Discrete Invariance Signatures

For any computer application of Invariance Signatures, a discrete version must be developed. The natural choice is the frequency histogram of the value of  $\iota_G$ . For a continuous contour, this is obtained by dividing the interval  $[0, 1]$  into  $n$  “bins” and integrating  $I(\iota_G)$  over each bin. For bins numbered from  $b_0$  to  $b_{n-1}$ , the value in bin  $k$  is thus

$$b_k = \int_{\frac{k}{n}}^{\frac{k+1}{n}} I(\iota_G) d\iota_G. \quad (7.52)$$

Since  $I(\iota_G)$  is a probability density function, the sum of the values of the bins must be one.

For a system using images of sampled contours, a true frequency histogram of the estimated local measures of consistency may be used. The designer of a system using these Invariance Signatures must choose the number of bins,  $n$ , into which the data is grouped. It will be seen in Section 7.4 that this choice is not arbitrary for real systems.

An example of a sampled contour and the estimated tangent vector at each point is shown in Figure 7.5. The circle indicates the centroid of the contour, and the dashed line shows the direction of  $\vec{e}_1$ . The estimated discrete Invariance Signatures are shown in Figure 7.6, for 20 bins.

It would be expected that this “flower”-shaped contour would have Invariance Signatures which reflect a quite strong dilation-invariant component corresponding to the approximately radial edges of the “petals”, and also a significant rotation-invariant component due to the ends of the petals which are approximately tangential to the radial edges. This is indeed what is observed in Figure 7.6.

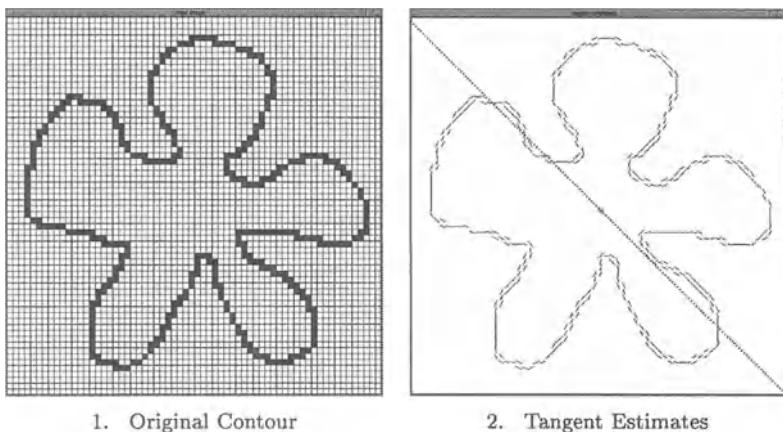


Figure 7.5: Example of a sampled contour and its estimated tangents.

### 7.3 The Invariance Signature Neural Network Classifier

We propose a Model-Based Neural Network to compute the discrete Invariance Signature of an input pattern, as described in Section 7.2.10, and to classify the pattern on that basis. This MBNN consists of a system of neural network modules, some hand-coded and some trained on sub-tasks. A schematic diagram is shown in Figure 7.7. This system will be referred to as the Invariance Signature Neural Network Classifier (ISNNC).

Whilst the ISNNC appears complex, it retains the basic characteristics

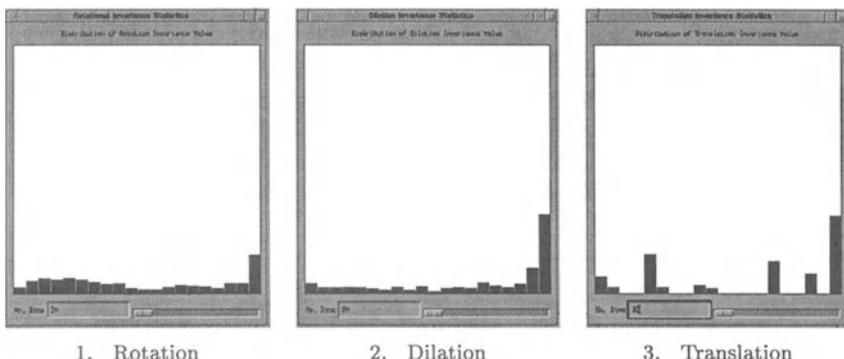


Figure 7.6: 20 bin discrete Invariance Signatures for the contour in Figure 7.5.

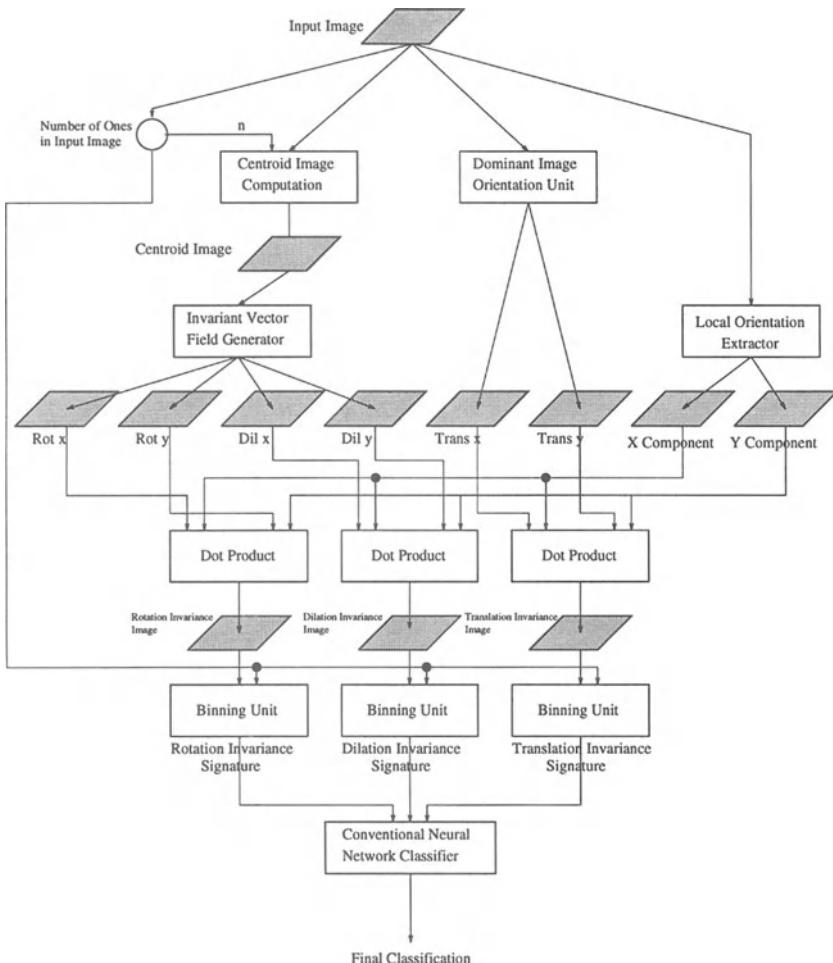


Figure 7.7: Invariance Signature-based contour recognition system.

of a traditional feed-forward neural network. It consists entirely of simple nodes joined by weighted connections.<sup>3</sup> Each node  $i$  in the network computes the sum of its  $j$  weighted inputs,  $\text{net}_i$ ,

$$\text{net}_i = \sum_j w_{ij} x_j. \quad (7.53)$$

This is then used as the input to the transfer function  $f$  of the node, which is either linear,  $f(\text{net}_i) = \text{net}_i$ , or the standard sigmoid,

---

<sup>3</sup>with the exception of the *Dominant Image Orientation Unit* for which a neural network solution is still to be developed.

$$f(\text{net}_i) = \frac{1}{1 + e^{-\text{net}_i}}. \quad (7.54)$$

The only departure from a traditional neural network at runtime is that some of the connection weights are calculated by nodes higher up in the network. We call these *dynamic weights*. The presence of dynamic weights allows the ISNNC to compute dot products,<sup>4</sup> and also for some nodes to act as gates controlling whether or not the output of a node is transmitted. Since connection weights in any implementation of a neural network are no more than references to some stored value, this should not present any difficulty.<sup>5</sup> Alternatively, the same functionality can be achieved by allowing a new class of node which multiplies its inputs. Nodes of this type are a component of Higher Order Neural Networks [253, 320].

The ISNNC consists of a number of distinct levels. Data is processed in a feed-forward manner, and computation at a given level must be completed before computation at the next level can begin. Each of the following subsections describes computation at a particular level.

### 7.3.1 Lie Vector Field Generation

#### 7.3.1.1 Calculation of a Centroid Image

The first step in creating the vector fields corresponding to the generators of the rotation and dilation groups is to compute the coordinates of the centroid of the image, since, as seen in Section 7.2.8, this must be the origin of coordinates. This is done using a neural module which takes as its input a binary image, and outputs an image of the same size which is zero everywhere except at the centroid, where it is one. In Figure 7.7 this is labelled “Centroid Image”. The weights of the Centroid Image Computation module are entirely hand-coded.

A quantity needed for this operation, and also later in the ISNNC, is  $N_{\text{on}}$ , the total number of “on” pixels in the input binary image  $\mathcal{I}(x_i, y_j)$ . This is given by

$$N_{\text{on}} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j). \quad (7.55)$$

This can easily be computed by a single node with a linear transfer function which has inputs from all input nodes with weights of 1. In fact, the value used in later levels of the ISNNC is  $\frac{1}{N_{\text{on}}}$ . Since the size of the

---

<sup>4</sup>The calculation of dot products, for example, is achieved by having the outputs of one layer provide the *weights* on connections to another layer.

<sup>5</sup>The aim of our research with MBNNs has never been biological plausibility. Perhaps this runtime weight calculation can be thought of as rapid, accurate training.

input image is fixed, it is trivial to build a neural module which interpolates the function  $f(x) = \frac{1}{x}$  with any desired accuracy for the integer values  $\{x : x \in [0, N_x N_y]\}$ . Bulsari [46], for instance, demonstrates how to construct piecewise constant approximations to problems such as this with neural networks. From now on it will be assumed that the value  $\frac{1}{N_{\text{on}}}$  is available.

The equations for the coordinates,  $(\bar{x}, \bar{y})$ , of the centroid of  $\mathcal{I}(x_i, y_j)$  are

$$\begin{aligned}\bar{x} &= \frac{1}{N_{\text{on}}} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j) x_i \\ \bar{y} &= \frac{1}{N_{\text{on}}} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j) y_i\end{aligned}\tag{7.56}$$

A simple system of two linear neurons can compute each of these quantities. Such a module is shown in Figure 7.8 (note that the “/” in the right-hand side of the node indicates that the transfer function is linear; an “S” indicates a sigmoid transfer function). This illustrates the notion of dynamic weights mentioned above. The weight on the single connection between Node A and Node B is dynamic, being computed from the number of ones in the input image. The weights on the connections to Node A are the  $x$ -coordinates of each of the input nodes.<sup>6</sup> These weights are fixed during the design process, so there is no sense in which a node needs to “know” what its coordinates are. An identical system is used to compute  $\bar{y}$ .

Once  $\bar{x}$  and  $\bar{y}$  have been calculated, the Centroid Image can be generated. This is done using a pair of nodes for each input node, as shown in Figure 7.9.<sup>7</sup> The weights and inputs to Nodes A and B are set so that they both “fire” only if their coordinate is within  $\theta$  of the centroid coordinate. Here  $\theta = 0.5$  was used, since the input nodes were assigned unit spacing. A neural AND of these nodes is then calculated by Node C.

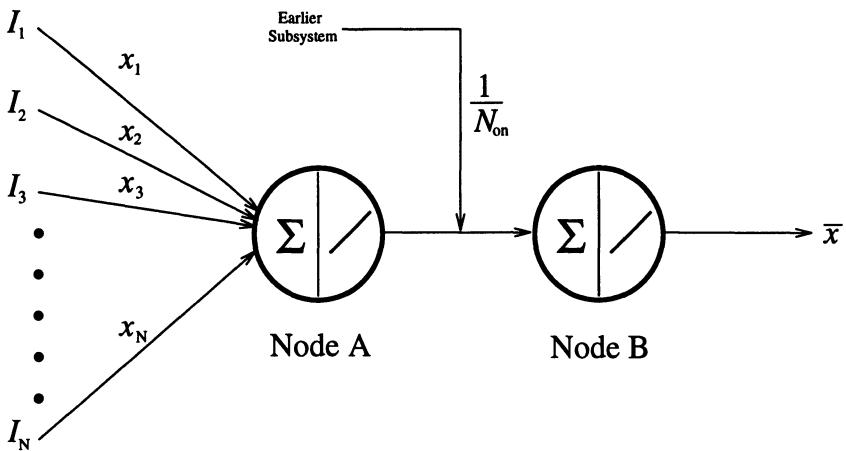
This results in two images, one for the  $x$  coordinate and one for  $y$ . Each has a straight line of activation corresponding to the centroid value for that coordinate. A neural AND of these images is computed, giving the final output Centroid Image,  $\mathcal{I}_C$ , in which only the centroid node is one.

This admittedly complicated neural procedure for computing the centroid is of course unnecessary if the system is being simulated on a digital computer. A hybrid system in which some calculations are not neural is a more efficient solution in that case. The goal in this section, however, is to show that parallel, neural solutions to these problems *are* possible.

---

<sup>6</sup>Not to be confused with their output values, which are also often denoted as  $x_i$  or  $y_i$ .

<sup>7</sup>The parameter  $\alpha$  in Figure 7.8 (and subsequent similar figures) determines the gain of the sigmoids. In this study it was set to 100.

Figure 7.8:  $\bar{x}$  module.

### 7.3.1.2 Gating of Vector Fields

Once the Centroid Image has been calculated, it is used to produce the rotation and dilation vector fields, four images in all. The weights from the Centroid Image are derived from Equations 7.49 and 7.50. Each centroid image node has weights going to all the (linear) nodes in each vector field component layer. Each weight has the value appropriate for the vector field at the destination node if the source node is the centroid. Since all nodes of the Centroid Image are zero except for the centroid node which is one, only the weights from the centroid node contribute to the vector

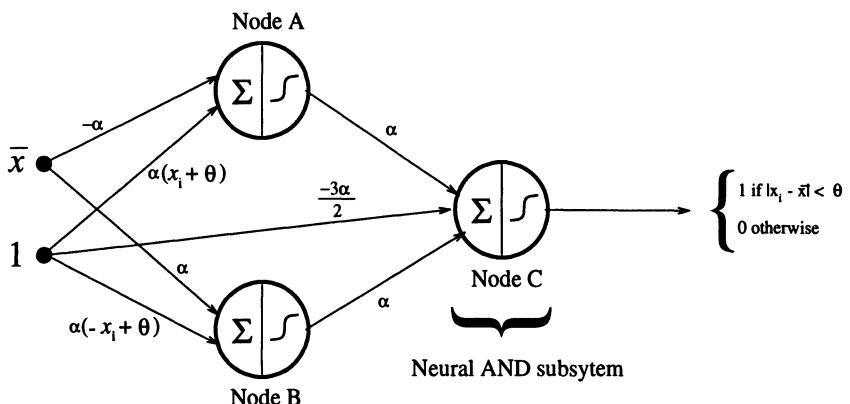


Figure 7.9: Coordinate matching module for node  $x_i$ . Output is 1 when  $|x_i - \bar{x}| < \theta$ .

field component images. Thus, weights corresponding to the vector fields for all possible centroid positions are present in the network, but only the appropriate ones contribute. Equation 7.57 shows the function computed by a node  $(k, l)$  in the rotation invariance vector field  $x$ -component image:

$$\text{Rot}_{x_{(k,l)}} = \sum_{i=0}^{x_{\max}} \sum_{j=0}^{y_{\max}} w_{(i,j)(k,l)} I_{C_{(i,j)}} \quad (7.57)$$

$$w_{(i,j)(k,l)} = \frac{-(j-l)}{\sqrt{(j-l)^2 + (i-k)^2}}.$$

Note that it does not matter which node is chosen as the origin of coordinates in each layer when the weights are set, only that it is consistent between layers. Similar functions for setting weight values, derived from Equations 7.49 and 7.50, are used for the other three vector component images.

### 7.3.2 Local Orientation Extraction

#### 7.3.2.1 A Tangent Estimate

In order to compute the Invariance Signature of an image, it is necessary to estimate the tangent vector at each point of the contour, since the mapping to invariance space requires the dot product of this with the three vector fields described in Section 7.2.6. A simple and robust estimate of the tangent vector at a point is the eigenvector corresponding to the largest eigenvalue of the covariance matrix of a square window centred on that point.

The size of the window chosen defines the number of orientations possible. Figure 7.10 shows the orientation of the tangent estimates for an image of a circle for both  $3 \times 3$  and  $5 \times 5$  windows, calculated exactly. It is clear that the tangent estimates calculated using the  $5 \times 5$  window are better than those from the  $3 \times 3$ , but this comes at a computational cost, since the number of calculations is proportional to the square of the window size. Moreover, the “best” choice of window size depends upon the scale of the input contour. If the window size becomes large compared to the scale of the contour, the dominant eigenvector of the window’s covariance matrix becomes a poor estimator of the tangent at a point, since the window is likely to include extraneous parts of the contour. This is clear from the tangent estimates shown in Figure 7.11, which was calculated with a window size of  $35 \times 35$ . Consequently, we choose to use a  $3 \times 3$  window, both for its computational advantage and because it makes no assumptions about the scale of the input contour.

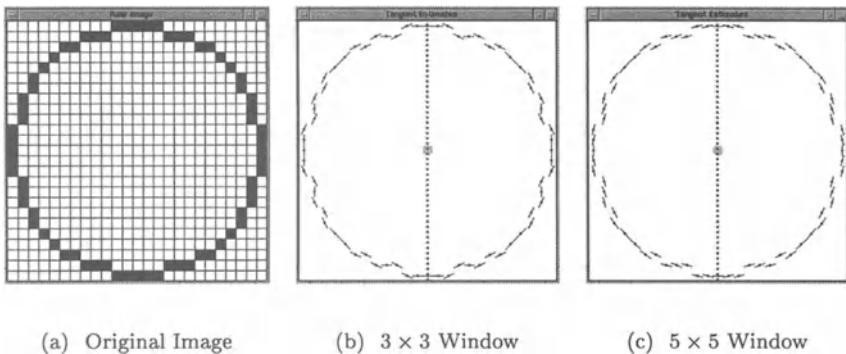


Figure 7.10: Tangent estimation with varying window sizes, using the eigenvector corresponding to the largest eigenvalue of the covariance matrix.

### 7.3.2.2 Discontinuities

Estimating this mapping presents some difficulties for a neural network, as it is not continuous. There are two discontinuities. The first arises when the dominant eigenvalue changes, and the orientation of the tangent estimate jumps by  $\frac{\pi}{2}$  radians. The second is due to the flip in sign of the tangent vector when the orientation goes from  $\pi$  back to 0.

The first discontinuity can be avoided by using a weighted tangent estimate. Rather than being unity, we let the magnitude of the estimated tangent vector corresponds to strength of the orientation. Let the eigenvalues of the covariance matrix be  $\lambda_1$  and  $\lambda_2$ , where  $\lambda_1 \geq \lambda_2$ . The corresponding unit eigenvectors are  $\hat{e}_1$  and  $\hat{e}_2$ . The weighted tangent vector estimate  $s$  is

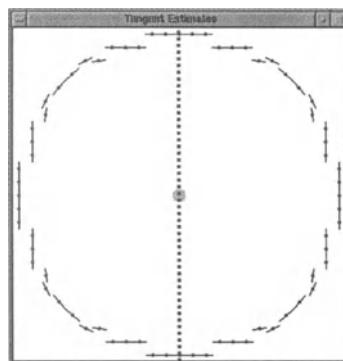


Figure 7.11: Tangents estimated with a window size of  $35 \times 35$ .

given by

$$\mathbf{s} = \begin{cases} \left(1 - \left|\frac{\lambda_2}{\lambda_1}\right|\right) \hat{\mathbf{e}}_1 & \text{if } \lambda_1 \neq 0, \\ 0 & \text{if } \lambda_1 = 0. \end{cases} \quad (7.58)$$

This weighting cause the magnitude of the estimated tangent vector to go to zero as  $\left|\frac{\lambda_2}{\lambda_1}\right| \rightarrow 1$ , and thus the  $\frac{\pi}{2}$  jump is avoided. There is, however, still a discontinuity in the derivative.

### 7.3.2.3 Training a Neural Orientation Extraction Module

We now aim to produce a neural orientation extraction module (NOEM) which approximates this mapping. This module is to be replicated across the input layer so that the  $x$  and  $y$  components of the tangent vector are estimated everywhere. The result is two images of the same size as the input layer. The values are gated by the input image (used as dynamic weights), so that values are only transmitted for nodes corresponding to ones in the input image. It was decided to produce this module by training a network on this task.

A training set was produced consisting of all 256 possible binary  $3 \times 3$  input windows with a centre value of 1, and, for each, the two outputs given by Equations 7.58. A variety of networks was produced and trained on this task. The performance measure used,  $E$ , was the ratio of the sum squared error to a variance measure for the training set,

$$E = \frac{\sum_{c=1}^N \sum_{j=1}^M (t_{cj} - y_{cj})^2}{\sum_{c=1}^N \sum_{j=1}^M (\bar{t}_{cj} - t_{cj})^2} \quad (7.59)$$

where  $N$  is the number of training exemplars,  $M$  the number of nodes in the output layer,  $t_{cj}$  the desired output value of the node and  $y_{cj}$  the actual output value.

A variety of standard, single hidden layer networks with different hidden layer sizes was trained using backpropagation [298]. It was found that the task could not be learnt without a significant number of hidden nodes, which is not unexpected for such a highly nonlinear function. The networks proved very unstable during training, so it was necessary to use an extremely small learning rate. As would be expected, the accuracy of the mapping improved with increasing hidden layer size, and the training time increased. The residual error did not stabilise, but continued to decrease steadily, albeit very slowly, as training was continued. The final accuracy achieved was thus ultimately determined by how long one was prepared to wait.

For the work reported in [322], a three layer neural network with a  $3 \times 3$  input layer, a 20 node hidden layer, and 2 output nodes was trained using backpropagation. After  $6 \times 10^6$  iterations with a learning rate of 0.0005, a

value of  $E = 3.11\%$  was reached. This was considered sufficiently accurate for this module. Further investigations, as will be seen in Section 7.4.1, have indicated that even such seemingly small residual errors in the Local Orientation Extraction module can adversely affect final classification performance. Consequently, means of improving the accuracy of this module were sought.

Despite the universality of single hidden layer neural networks, several authors have indicated that networks with multiple hidden layers can be useful for extremely nonlinear problems [300, 319]. This suggests that a more compact solution might be obtained using a network with two hidden layers. Several standard four-layer networks were trained using backpropagation on this problem. It did indeed seem that they converged more rapidly, and to a smaller residual error. However, as with the single hidden layer networks, they were very sensitive to the learning rate, suggesting that perhaps a more sophisticated learning algorithm, such as QuickProp [100], would be more appropriate. After an initial large reduction, the error continued to decrease very slowly throughout training. The limitation on the final accuracy of the approximation appeared to be how long one was prepared to wait as much as the number of hidden nodes.

It is noted that this is a similar problem to edge extraction, the difference being that edge extraction is usually performed on images containing grey-scale gradients rather than on a binary, thin contour. Srinivasan, Bhatia and Ong [323] have developed a neural network edge detector which produces a weighted vector output very much like the one described in Equation 7.58. They used an encoder stage which was trained competitively, followed by a backpropagation-trained stage. The encoder produced weight distributions closely resembling Gabor filters of various orientations and phases. A more compact and accurate tangent estimator might be developed using an MBNN incorporating a stage with Gabor weighting functions, as used in [54].

### 7.3.3 Calculation of the Local Measure of Consistency

The next stage of the network computes the Local Measure of Consistency of the tangent estimate at each point of the contour for each of the Lie vector fields. The output required is an *Invariance Image* for each Lie transformation. This is an image of the same size as the input image, where the value at each point is given by the absolute value of the dot product of the estimated tangent vector to the contour and the Lie vector field at that point, as specified in Equation 7.33.

The neural implementation is simple. With reference to Figure 7.7, the  $x$ -component image from the Local Orientation Extractor provides dynamic weights to be combined with the  $x$ -component of each of the Lie vector fields. The same is done for the  $y$ -components. For each Lie transformation,

there is thus layer of neurons which each have two inputs. The inputs come from the vector field images at the same coordinates, and are weighted by the Local Orientation images, again at the same coordinates. Note that points corresponding to zeros in the input image have tangent estimates of zero magnitude and thus make no contribution.

As an example, consider the subsystem corresponding to the point in the input image at coordinates  $(i, j)$ . Figure 7.12 shows the neural implementation of the first stage of the calculation. Modules of this type are replicated for each point in the input image, and for each Lie vector field image. All that then remains is to pass the output of these modules through modules which calculate the absolute value of their input.

Figure 7.13 shows a neural module which calculates a good approximation of the absolute value of its input. It is less accurate for inputs very close to zero, but in this application, where possible orientations are coarsely quantised, this does not cause any problems. This completes the neural calculation of the Local Measure of Consistency for each contour point with respect to each Lie vector field. All that remains is to combine these into an Invariance Signature.

### 7.3.4 Calculation of the Invariance Signature

The ability of the MBNN approach to produce modules that perform tasks not usually associated with neural networks is illustrated by the binning unit shown in Figure 7.14. There is one of these modules for each of the  $n$  bins comprising the Invariance Signature histogram for each invariance class. Each binning module is connected to all the nodes in the Invariance Image, and inputs to it are gated by the binary input image, so that only the  $N$  nodes corresponding to ones in the input image contribute.

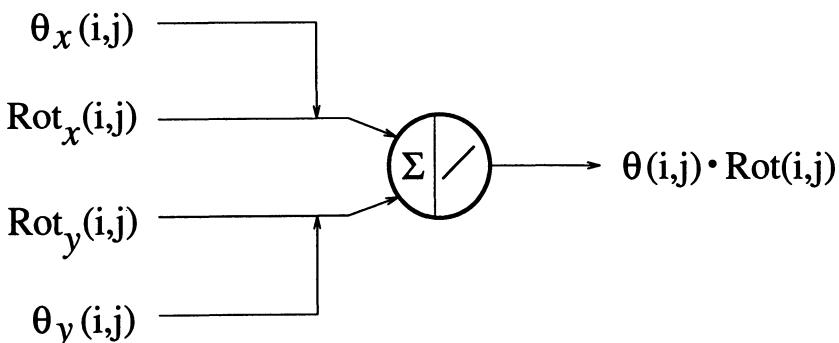


Figure 7.12: Calculation of the dot product of the tangent estimate  $\theta$  and the vector field corresponding to rotation, for the image point at coordinates  $(i, j)$ .

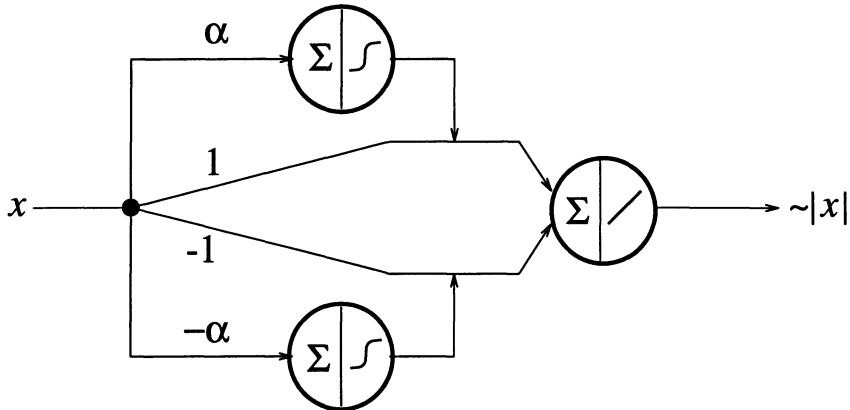


Figure 7.13: Neural module which calculates the absolute value of its input.

The  $n$  bins have width  $\frac{1}{n-1}$ , since the first bin is centred on 0, and the last on 1.<sup>8</sup> The system is designed so that Nodes A and B only have an output of 1 when the input  $x$  is within bin  $i$ . The condition for  $x$  to be in bin  $i$  is:

$$\frac{2i-1}{2(n-1)} < x < \frac{2i+1}{2(n-1)}. \quad (7.60)$$

In order to detect this condition, the activations of Nodes A and B are set to:

$$net_A = \alpha x - \frac{\alpha(2i-1)}{2(n-1)} \quad (7.61)$$

$$net_B = -\alpha x + \frac{\alpha(2i+1)}{2(n-1)}. \quad (7.62)$$

The outputs of these nodes go to Node C, which computes a neural AND. Node D sums the contributions to bin  $i$  from all N nodes.

This concludes the calculation of the Invariance Signature. The end result is a signature which consists of  $3n$  values, where  $n$  is the number of bins in the invariance signature histogram for each invariance class. This calculation has been achieved in the framework of a modular MBNN, where module functions are strictly defined, and are completely independent of any training data. The building blocks of the individual modules are restricted to simple summation artificial neurons, with either linear or sigmoidal transfer functions. The sole departure from standard neural network components is the introduction of dynamic weights, but, as already stated,

<sup>8</sup>This is because a bin ending at an extreme of the range would miss contributions from the extreme value since the edge of the neural bin is not vertical.

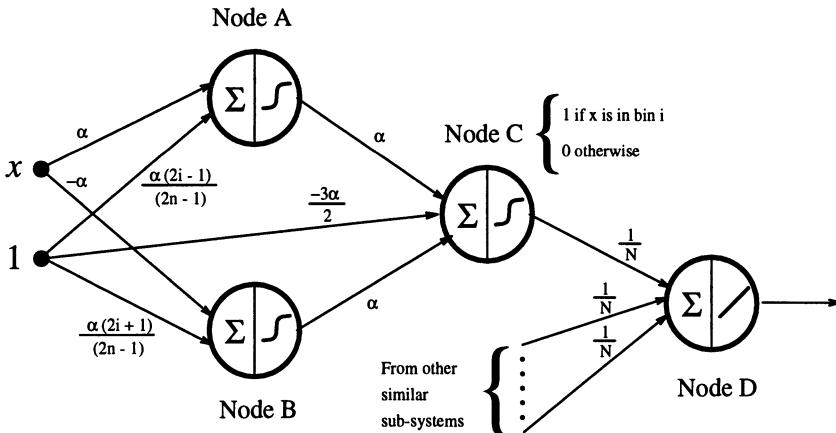


Figure 7.14: Neural binning module.

these can be eliminated if product neurons are used, as in Higher Order Neural Networks [253, 320]. This shows the power of the modular approach to neural network design, and demonstrates that it is possible to design modules which perform tasks that are not often considered to be part of the neural networks domain, such as binning data.

A final module can be added to this network which is trained to classify patterns on the basis of the  $3n$ -dimensional Invariance Signature, rather than on the input patterns themselves. Classification performance will thus be guaranteed to be invariant under shift, rotation and scaling. The advantages of this approach are demonstrated in Section 7.4.

## 7.4 Character Recognition with Invariance Signature Networks

In Section 7.2 a new class of invariant signature for two dimensional contours was derived. It was established that these Invariance Signatures are themselves invariant under rotations, dilations and translations of the contour. In Section 7.3 it was shown that an MBNN can be constructed which computes these signatures and classifies contours on that basis.

It remains now to demonstrate that these Invariance Signatures retain enough information to be usable for pattern recognition, and that they are not unduly sensitive to noisy data such as that encountered in real applications. In order to do this, the system is applied to the classification of letters of the Roman alphabet, both for “perfect”<sup>9</sup> machine-generated

<sup>9</sup>Throughout this section, the term “perfect” will be used to describe data which is both noise-free and unambiguous.

training and test data, and for data gathered using physical sensors.

### 7.4.1 Perfect Data

#### 7.4.1.1 Departures from Exact Invariance

Despite the fact that Invariance Signatures are invariant under rotations, dilations and shifts in the plane when calculated for a continuous contour, departures from invariance occur in real applications in several ways. Scanned data contains random noise from the sensor, though the quality of scanners now available renders this negligible for this application. More important sources of error are discussed below.

**7.4.1.1.1 Quantisation Noise** Noise is introduced into the tangent estimation procedure by the sampling of the contour. Since the estimated tangent orientation is quantised,<sup>10</sup> the value of the Local Measure of Consistency  $\iota_G$  can be changed when a contour is quantised at a new orientation. It is possible to compensate partially for this effect by using sufficiently wide bins when calculating the Invariance Signature  $I(\iota_G)$ , but there will still be errors when the change in estimated orientation moves the resultant  $\iota_G$  across bin boundaries.

**7.4.1.1.2 Ambiguous Characters** In many fonts some letters are rotated or reflected versions of other letters, such as {b, d, p, q} and {n, u}. In fonts with serifs, the serifs are often so small that they make a negligible contribution to the Invariance Signature. Consequently, it is impossible to classify isolated characters into 26 classes if shift, rotation, scale and reflection invariance is desired. Admittedly, reflection invariance is not usually desired in a character recognition system, but it is an unavoidable characteristic of the ISNNC. In commercial optical character recognition systems, context information is used to resolve ambiguous letters,<sup>11</sup> which occur even in systems without inherent invariances. Such an approach would be equally applicable as a post-processing stage for the ISNNC system.

#### 7.4.1.2 The Data Set

These effects can be avoided by using a computer to produce a “perfect” data set, which is free from quantisation noise and contains no ambiguous characters. Such a data set can be used to demonstrate that Invariance Signatures retain sufficient information for classification in the absence of noise, and these results can be used as a basis for assessing the performance of the system on real data.

---

<sup>10</sup>See Section 7.3.2.

<sup>11</sup>i.e. surrounding letter classifications and a dictionary of acceptable words.

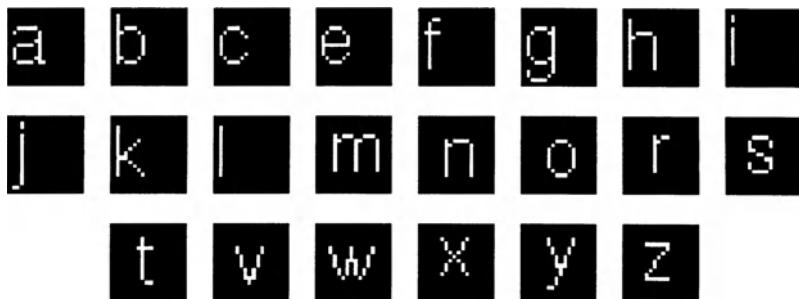


Figure 7.15: Training set of canonical examples of unambiguous characters.

A training data set was created using a screen version of the Helvetica font. Only the letters  $\{a, b, c, e, f, g, h, i, j, k, l, m, n, o, r, s, t, v, w, x, y, z\}$  were used, so that ambiguity was avoided. An  $18 \times 18$  binary image of each letter was produced. This training data set is shown in Figure 7.15.

A “perfect” test data set was created by computing reflected and rotated versions of the training data, where all rotations were by multiples of  $\frac{\pi}{2}$  radians, so that there was no quantisation error. This test data set is shown in Figure 7.16.

#### 7.4.1.3 Selected Networks Applied to this Problem

Simulations indicated that the training set shown in Figure 7.15 could be learnt by a neural network with no hidden layer: it is a linearly separable problem [212]. It was assumed that this would also be true of the Invariance Signatures calculated from these data.<sup>12</sup> Consequently, two different network architectures were constructed for comparison. The first was a fully-connected MLP (Multilayer Perceptron), or TNN, with a  $18 \times 18$  input layer, no hidden layers, and a  $1 \times 22$  output layer. The other was an ISNN, with an  $18 \times 18$  input layer for the Invariance Signature calculation stage. The Invariance Signature was calculated using 5 bins for each Lie transformation, meaning that the Invariance Signature layer had  $3 \times 5$  nodes. This was connected to a  $1 \times 22$  output layer, with no intervening hidden layers, forming an linear classifier sub-network.

#### 7.4.1.4 Reduction of Data Dimensionality

It should be noted that although the Invariance Signature Calculation stage of the ISNN has to be run every time a new pattern is to be classified at run time, it is only necessary to calculate the Invariance Signatures of the training and test data once. The final classification stage of the ISNN

---

<sup>12</sup>This was confirmed by subsequent experiments.

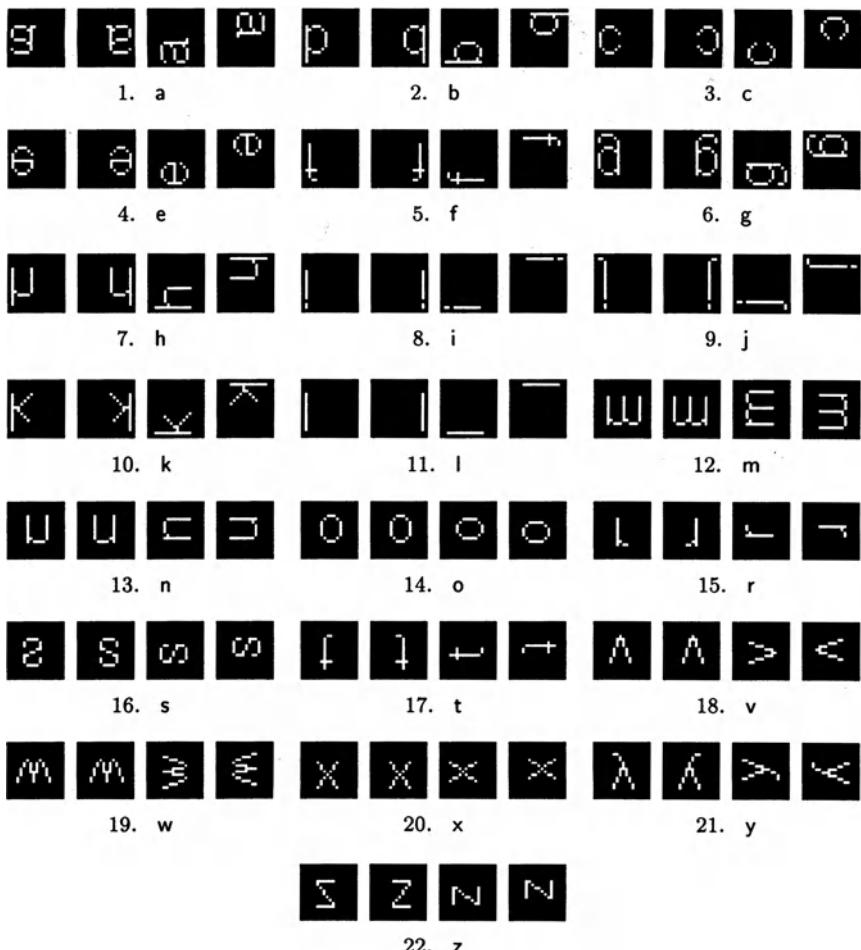


Figure 7.16: Test set of ideally shifted, rotated and reflected letters.

can then be trained as a separate module. This can lead to an enormous reduction in the time taken to train a network, since the number of training passes required to learn a large and complex data set is typically order  $10^2$  to  $10^3$ , and the total number of parameters  $n_p$  in a TNN is:

$$n_p = \sum_{i=1}^{N-1} (\text{nodes in layer})_{i-1} \times (\text{nodes in layer})_i \quad (7.63)$$

where  $N$  is the total number of layers, and  $i$  is the layer number, where the input layer is labelled 0. The iteration time during training is proportional to  $n_p$ , so, for this example, each training iteration for the Invariance Signa-

ture classification stage will be  $\frac{18 \times 18}{3 \times 5} = 21.6$  times faster than that for the TNN.

The calculation of the Invariance Signatures is admittedly time-consuming, but the time is made up many times over during training when the input image size is large. In real applications, the image size is typically larger than the  $18 \times 18$  image used in this demonstration. ISNNC systems can therefore provide an extremely significant reduction in the dimensionality of the training data, and a corresponding reduction in the time taken to train the classification network. Moreover, the simulation of the ISNNC on a sequential computer is unable to take advantage of the inherently parallel, local computations that characterise many of the ISNNC modules. A parallel implementation would largely alleviate the time-consuming nature of the Invariance Signature calculation.

#### 7.4.1.5 Perfect and Network-Estimated Local Orientation

Since the neural network implementation of the Local Orientation Unit described in Section 7.3.2 still had some residual error, two versions of the Invariance Signature training and test data were created, one with the local orientation at each point calculated directly from the covariance matrix eigenvectors, and the other using the neural network module. Results from these data sets will be compared to evaluate the importance of accurate orientation extraction in the neural module.

Ten examples of each network were made, each with a different initialisation of parameters, so that it could be ensured that the results were reproducible. The TNNs and the ISNNC linear classifier modules were all trained using the backpropagation algorithm. Training was continued for 1000 iterations. A network was deemed to have classified a pattern as belonging to the class corresponding to the output node which had the highest output value, so no patterns were rejected.

**7.4.1.5.1 Results for Traditional Neural Networks** The results obtained with TNNs are summarised in Table 7.1. It is clear that the TNNs do not exhibit transformation invariance. They could not be expected to do so, since no transformed versions of the patterns were included in the training data.

It might be expected that the fact that the final performance of the TNNs is better than chance ( $\frac{1}{22} = 4.5454\%$ ) is because some of the transformations of the training data resulted in patterns almost identical to the untransformed training pattern for some of the highly symmetrical letters in the training set (*e.g.* o, s, x and z). Analysis of which test patterns were classified correctly, however, shows that this is not the case. The 12 correctly classified test patterns are shown in Figure 7.17. No reason for these particular patterns being classified correctly is obvious. It is well-

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	100.00	14.78	100.00	13.64
Network 2	100.00	14.78	100.00	13.64
Network 3	100.00	15.91	100.00	13.64
Network 4	100.00	14.78	100.00	13.64
Network 5	100.00	14.78	100.00	13.64
Network 6	100.00	15.91	100.00	13.64
Network 7	100.00	14.78	100.00	13.64
Network 8	100.00	14.78	100.00	13.64
Network 9	100.00	14.78	100.00	13.64
Network 10	100.00	14.78	100.00	13.64
$\mu \pm \sigma$	100.00 $\pm$ 0.00	15.00 $\pm$ 0.45	100.00 $\pm$ 0.00	13.64 $\pm$ 0.00

Table 7.1: Classification performance (%) correct) of traditional neural network classifiers trained for 1000 iterations with the data shown in Figure 7.15 and tested with the “perfect” data set in Figure 7.16.

known that the “rules” generated by TNNs are hard to extract, although progress has been made in that area [360, 361]. What is clear, however, is that chance must play a part in some of these classifications, for instance  $i_4$ , which shares no “on” pixels with the training example of  $i$  (see Figures 7.15 and 7.17).

Table 7.2 shows that marginally better performance on the test data could have been achieved by employing an early-stopping scheme in which training was stopped when classification performance on the test set started to deteriorate. This would, however, have been at the cost of less than 100% correct performance on the training data. It should be noted that the sum squared-error on the test set decreased throughout the entire 1000 iterations. This indicates that an early-stopping scheme based on the test set error would fail to improve classification performance in this case. Figure 7.18 shows how the classification performances and errors typically varied during training.

It should be acknowledged that the TNNs could not really be expected

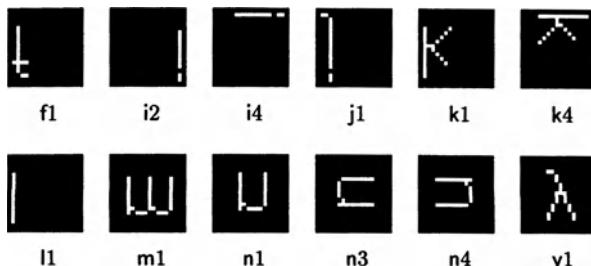


Figure 7.17: Test patterns classified correctly by the TNNs.

Iteration	% Correctly Classified		
	Training Data	Test Data	
Network 1	86	100.00	14.773
Network 2	33	95.455	14.773
Network 3	1	90.909	15.909
Network 4	85	100.00	14.773
Network 5	94	100.00	14.773
Network 6	1	90.909	15.909
Network 7	95	100.00	14.773
Network 8	1	95.455	14.773
Network 9	86	100.00	14.773
Network 10	19	95.455	14.773
$\mu \pm \sigma$	50 $\pm$ 40	96.818 $\pm$ 3.550	15.000 $\pm$ 0.454

Table 7.2: Performance at iteration at which best performance on test data occurred for traditional neural network classifiers trained with the data shown in Figure 7.15 and tested with the “perfect” data set in Figure 7.16.

to perform better than chance on this test set given the training set used. Their architecture provides no invariances, and generalisation cannot be expected unless multiple transformed versions of the patterns are included in the training set. This argument can be used against all the comparisons between TNNs and MBNNs in this chapter: they are not really fair.

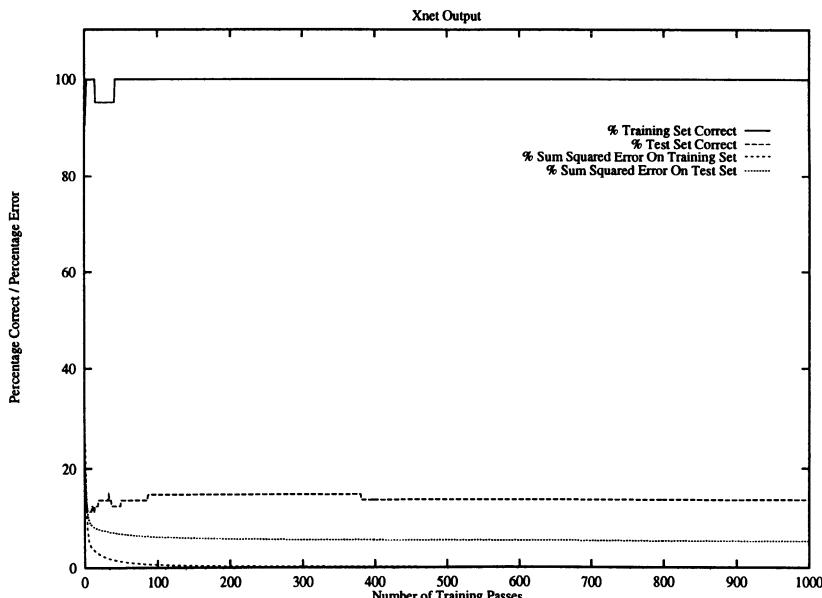


Figure 7.18: Classification performance and errors of TNN No. 2 during training.

Nevertheless, these comparisons between naive applications of TNNs and specifically-designed MBNNs demonstrate that MBNNs can perform successfully using training sets completely inadequate for TNNs. Moreover, these MBNNs are of lower dimensionality than the TNNs. Providing TNNs with sufficiently large training sets would only make their training still more computationally-expensive, with no *guarantee* of invariant performance.

**7.4.1.5.2 Results with Perfect Local Orientation** The results for the ten ISNNCs which used perfect Local Orientation Extraction are shown in Table 7.3. The average number of iterations for 100% correct classification to be achieved was 220. Since the problem is linearly-separable, it could in fact be solved directly, using a linear equations technique such as singular-valued decomposition [269]. Since weights may set by any method at all in the MBNN paradigm, this makes the comparison of convergence times somewhat irrelevant. Nevertheless, the MBNN modules, although taking, on average, 4.4 times as many iterations to converge as the TNNs, were still 4.9 times faster to train, due to their lower dimensionality, as discussed in Section 7.4.1.4.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	100.00	100.00	100.00	100.00
Network 2	100.00	100.00	100.00	100.00
Network 3	100.00	100.00	100.00	100.00
Network 4	100.00	100.00	100.00	100.00
Network 5	100.00	100.00	100.00	100.00
Network 6	100.00	100.00	100.00	100.00
Network 7	100.00	100.00	100.00	100.00
Network 8	100.00	100.00	100.00	100.00
Network 9	100.00	100.00	100.00	100.00
Network 10	100.00	100.00	100.00	100.00
$\mu \pm \sigma$	$100.00 \pm 0.00$	$100.00 \pm 0.00$	$100.00 \pm 0.00$	$100.00 \pm 0.00$

Table 7.3: Classification performance (% correct) of Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 7.15 and tested with the “perfect” data set in Figure 7.16.

As expected, the ISNNCs generalise perfectly to the transformed images in the test set. The network architecture constrains the system to be shift-, rotation-, scale- and reflection-invariant in the absence of quantisation noise, so this is no surprise. Importantly, the result indicates that sufficient information is retained in the 5 bin Invariance Signatures for all 22 unambiguous letters of the alphabet to be distinguished. Inspection of the percentage sum squared error values after each iteration indicated that the error on the test set was indeed identical to that on the training set: for perfect data, the ISNNC produces perfect results.

**7.4.1.5.3 Results using the NOEM** The perfect results above were obtained using a hybrid system, which used a module which was not a neural network to calculate the local orientation at each point. The neural module for local orientation extraction used here was trained using back-propagation until 98.93% of the variance in the training data was explained (see Equation 7.59). The Invariance Signatures for the test and training data were calculated using the system incorporating this module, and the classification module was then trained separately using these Invariance Signatures. Systems were produced with both 5 and 10 bin Invariance Signatures. The results obtained are shown in Tables 7.4 and 7.5.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	100.00	96.591	100.00	96.591
Network 2	100.00	96.591	100.00	96.591
Network 3	100.00	96.591	100.00	96.591
Network 4	100.00	96.591	100.00	96.591
Network 5	100.00	96.591	100.00	96.591
Network 6	100.00	96.591	100.00	96.591
Network 7	100.00	96.591	100.00	96.591
Network 8	100.00	96.591	100.00	96.591
Network 9	100.00	96.591	100.00	96.591
Network 10	100.00	96.591	100.00	96.591
$\mu \pm \sigma$	100.00 $\pm$ 0.00	96.591 $\pm$ 0.00	100.00 $\pm$ 0.00	96.591 $\pm$ 0.00

Table 7.4: Classification performance (% correct) of 5 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 7.15 and tested with the “perfect” data set in Figure 7.16.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	100.00	95.455	100.00	93.182
Network 2	100.00	95.455	100.00	93.182
Network 3	100.00	95.455	100.00	93.182
Network 4	100.00	95.455	100.00	93.182
Network 5	100.00	95.455	100.00	93.182
Network 6	100.00	95.455	100.00	93.182
Network 7	100.00	95.455	100.00	93.182
Network 8	100.00	95.455	100.00	93.182
Network 9	100.00	95.455	100.00	93.182
Network 10	100.00	95.455	100.00	93.182
$\mu \pm \sigma$	100.00 $\pm$ 0.00	95.455 $\pm$ 0.00	100.00 $\pm$ 0.00	93.182 $\pm$ 0.00

Table 7.5: Classification performance (% correct) of 10 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 7.15 and tested with the “perfect” data set in Figure 7.16.

The misclassified patterns for the 5 bin ISNNCs were those shown in Figure 7.19. t1 and t2 were classified as f, which is understandable, since there is very little difference indeed between the patterns. t4 was misclassified as a. All ten networks had these same misclassifications. These data show that the small residual error in the NOEM does cause a degradation in the transformation invariant classification performance of the ISNNCs. These results are, however, still far superior to those for the TNNs. Moreover, there is no reason that the NOEM could not be trained further. It is to be expected that the invariant classification performance would continue to approach 100% as the accuracy of the module was improved.

The results for the 10 bin system in Table 7.5 show that the effects of inaccuracies in the NOEM are greater when the number of bins is increased. This is because the more numerous bins are necessarily narrower. Thus a small difference in the estimated tangent at a point can more easily cause the Local Measure of Consistency to fall in a different bin, altering the Invariance Signature.

#### 7.4.2 Optical Character Recognition

Having demonstrated that Invariance Signatures retain sufficient information for the classification of “perfect” character data, it is now necessary to show that the system can be used to achieve transformation invariant recognition of data in the presence of sensor and quantisation noise. To this end, it was decided to apply ISNNCs to the classification of scanned images of shifted and scaled printed alphabetic characters.

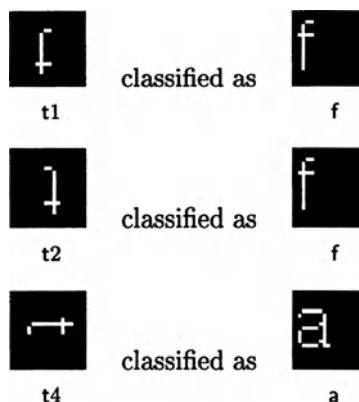


Figure 7.19: Test Patterns Misclassified by the 5 Bin Invariance Signature Neural Network Classifiers, and the training examples as which they were incorrectly classified.

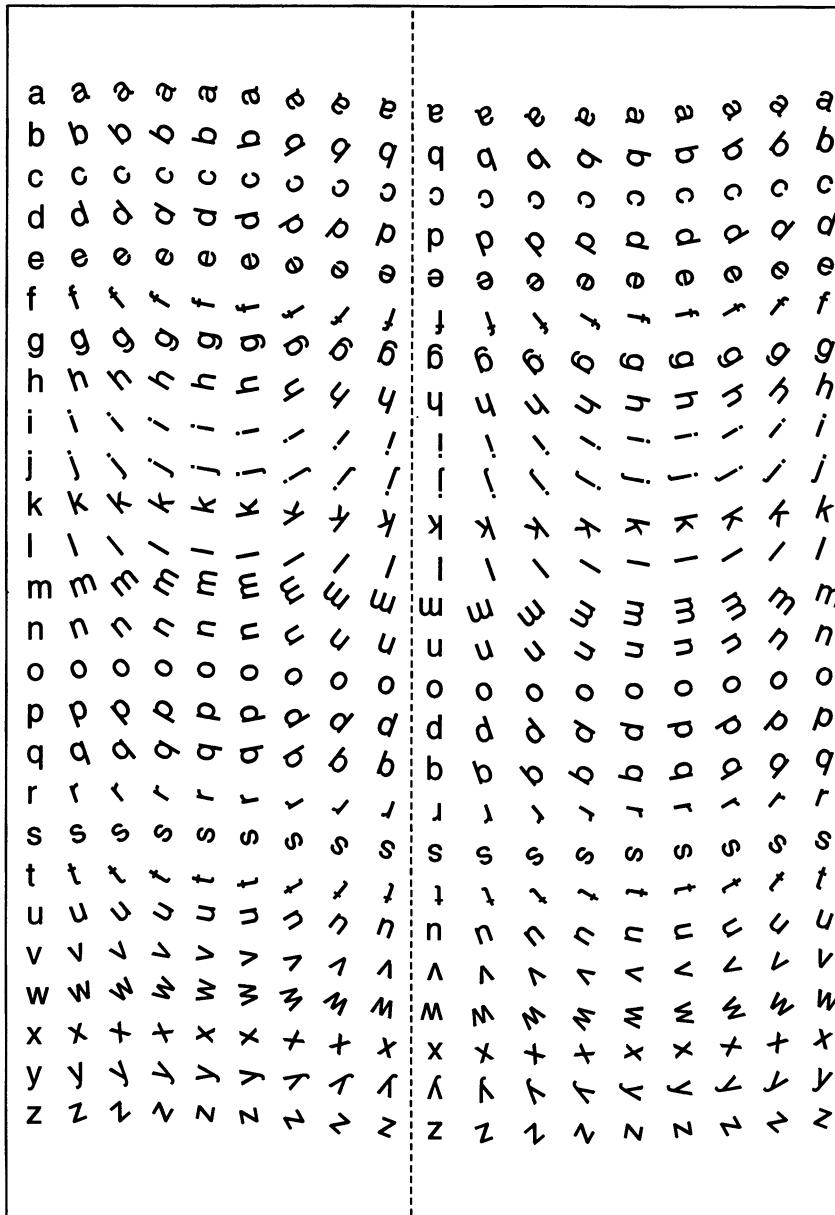


Figure 7.20: The characters scanned to generate the real data set. The box shows the border of an A4 page (210mm × 297mm) so that the size of the original characters may be judged. The dashed line shown the partition into training and test data.

#### 7.4.2.1 The Data Set

The data set used was derived from the set of shifted and rotated versions of the lowercase Roman alphabet shown in Figure 7.20. Each character appears in 18 different orientations, at rotation increments of 20 degrees. The shifts arise due to the fact that characters are extracted using a bounding-box technique which takes no account of the centroid position.

An A4 page with these characters printed onto it from a laser printer at 300 dots per inch was scanned at 75 dots per inch using a UMAX Vista-S6 Scanner. The connected regions in this image were then detected, and the minimum bounding-box which could contain the largest of the characters was calculated ( $56 \times 57$  pixels). The image was then segmented into separate characters, and each character was thinned using an algorithm due to Chen and Hsu [62]. The thinning was necessary since the Invariance Signature method is only applicable to thin contours. This set of 468 characters was partitioned into a training set and a test set. The training set consisted of the characters rotated by angles in the range  $[0^\circ, 160^\circ]$  relative to the upright characters, and the test set of those in the range  $[180^\circ, 340^\circ]$ . In Figure 7.20, those characters to the left of the dashed line were used for training, and those to the right for testing. Some examples of the resultant scanned, extracted and thinned training set is shown in Figure 7.21, and examples of the test set in Figure 7.22.

It is interesting to compare the subjective visual similarity between the Invariance Signatures both within and between classes for these extracted data. Figure 7.23 shows the first four training examples of the letter “a”, accompanied by images showing the tangent estimates at each point of the contours. The tangent estimate is represented by a line segment with the orientation of the estimated tangent. These images do not indicate the weight assigned to each estimated tangent.

It is not easy to interpret the similarity between these tangent representations of the contours. For this, it is necessary to see the Invariance Signatures. Figure 7.24 shows the Invariance Signatures for the patterns in Figure 7.23 with respect to rotation, dilation and translation. The Invariance Signatures are represented by the Invariance Measure Density histograms. It is immediately apparent that there is a great deal of (subjective) similarity between these representations of the shifted and rotated patterns. They are not identical, as a result of the various sources of noise discussed in Section 7.4.1. For classification purposes, however, all that is required is that these signatures be more similar to each other within a letter class than between letter classes. This must be determined by experiment.

Figures 7.25 and 7.26 show the equivalent data for the first four training patterns for the letter “x”. These again show the marked within-class similarity, and are also distinctly different to the signatures for the letter “a”: these letters were deliberately chosen since “a” is “quite rotationally-invariant”, whereas “x” is “quite dilationally-invariant”.

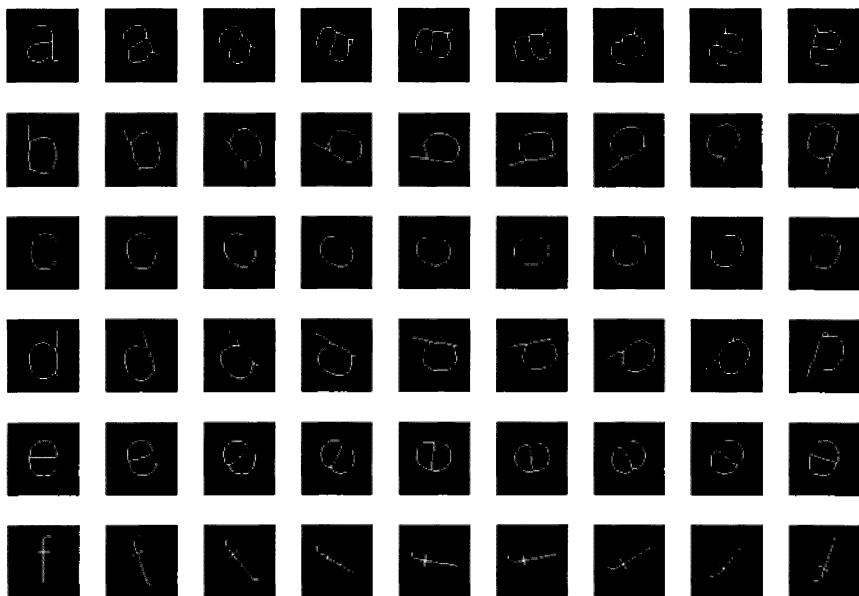


Figure 7.21: Examples from the training set of thinned, scanned characters.

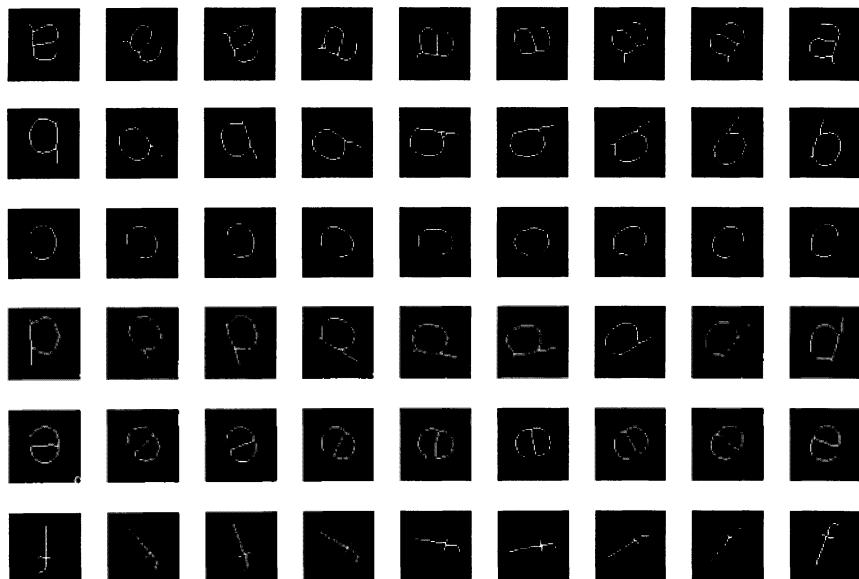


Figure 7.22: Examples from the test set of thinned, scanned characters.

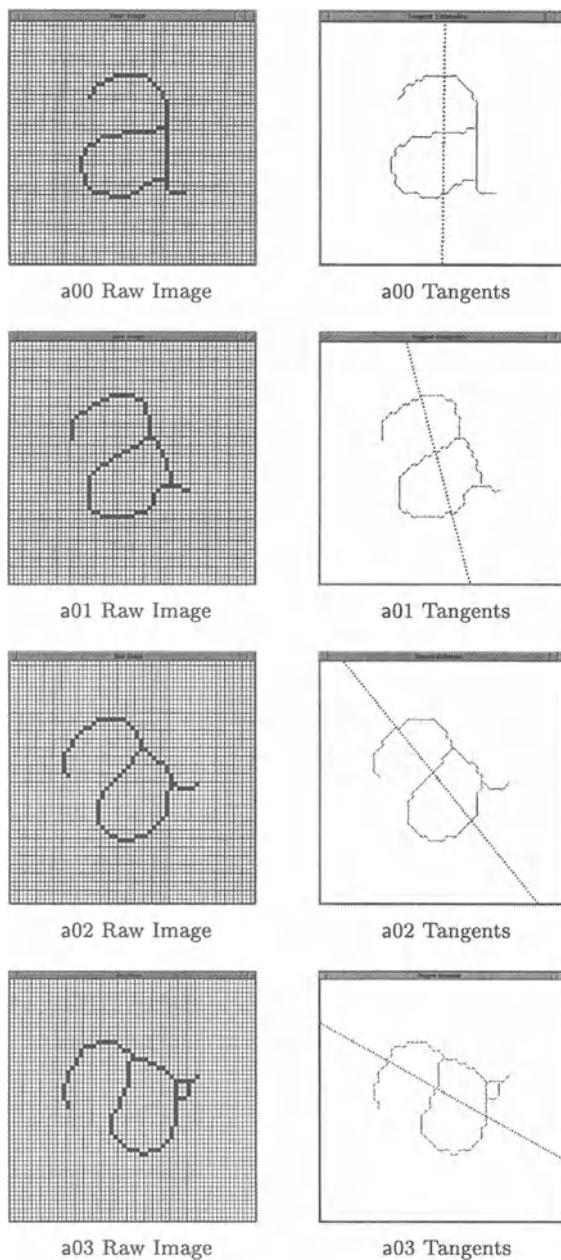


Figure 7.23: Tangents estimated for training examples of the letter “a”.

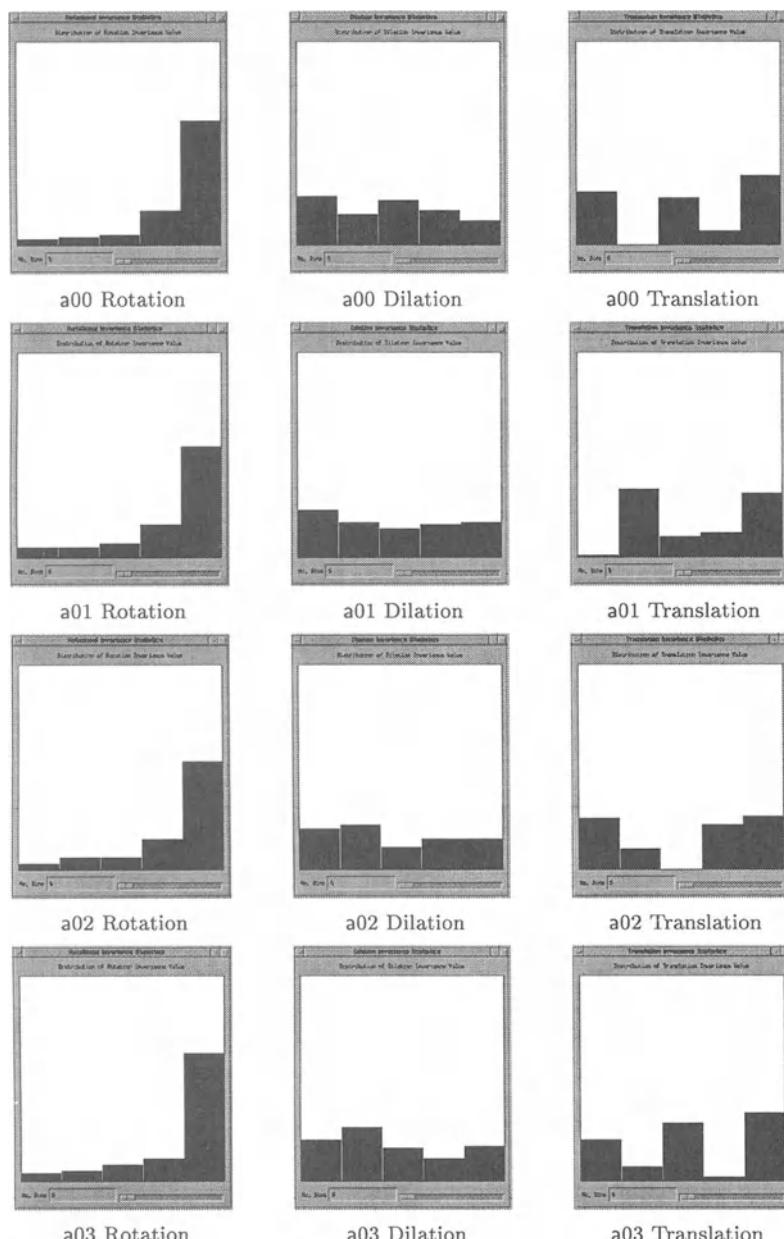


Figure 7.24: 5 bin Invariance Signatures for training examples of the letter "a".

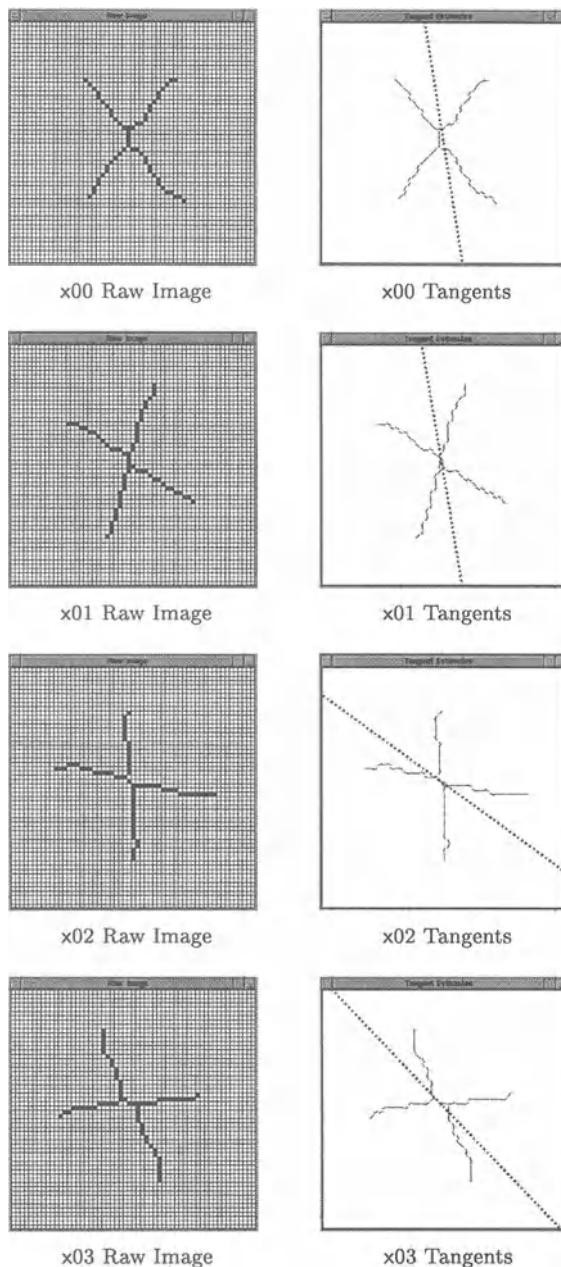


Figure 7.25: Tangents estimated for training examples of the letter "x".

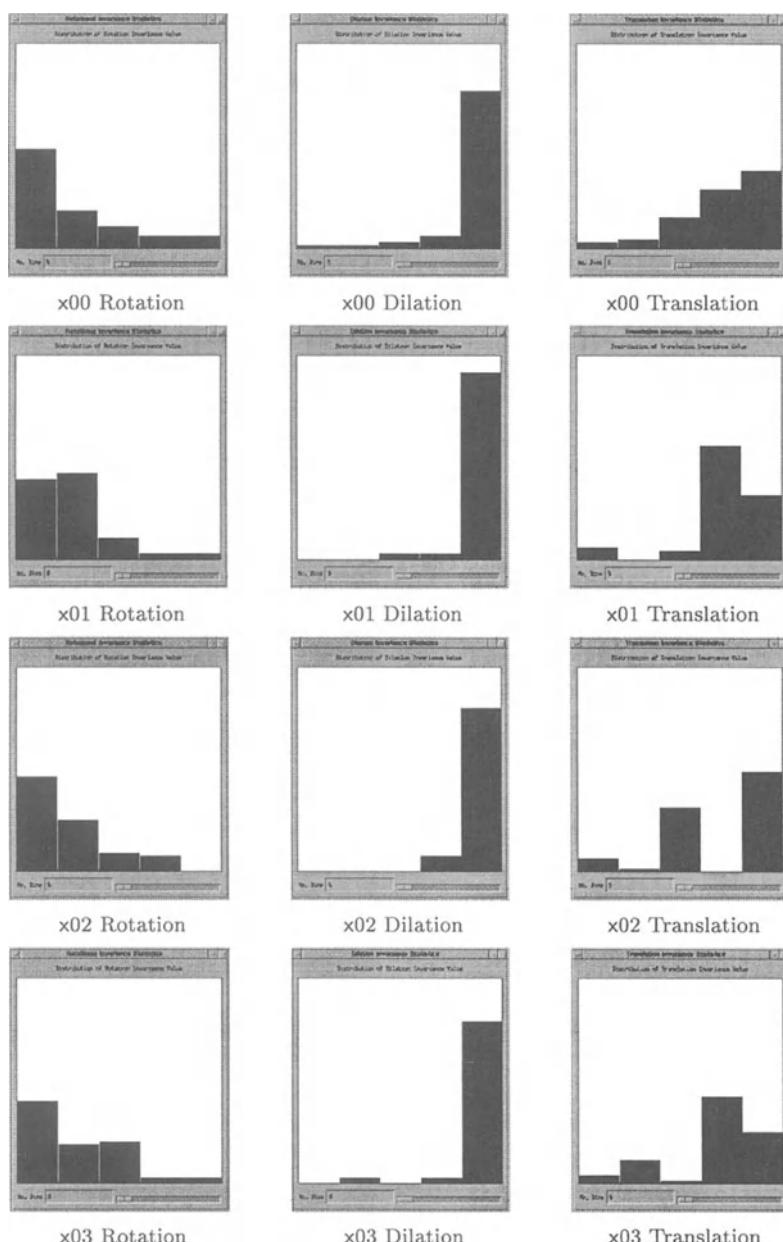


Figure 7.26: 5 bin Invariance Signatures for training examples of the letter "x".

#### 7.4.2.2 Selected Networks Employed for this Problem

The methodology employed for this experiment was identical to that used for the synthetic data, described in Section 7.4.1.3. The TNN used in this case had a  $56 \times 57$  node input layer, and a  $1 \times 26$  output layer, giving a massive 83018 independent parameters to be estimated. With only 234 training patterns and 83018 parameters, this problem is almost certain to be linearly separable. Simulation verified that this problem was indeed learnable by a network with no hidden layer.

A variety of different classification modules was tried, for ISNNCs with both 5 and 10 bin Invariance Signatures. These included the simple linear classifier, and a variety of MLP classifiers with differing numbers of nodes in their hidden layers. These experiments indicated that 5 bin Invariance Signatures were insufficient for this problem, and that it was not linearly separable. It also became clear that the errors introduced by the slightly inaccurate network-estimated Local Orientation Extraction caused a noticeable departure from invariance (see Section 7.4.1.5.3). For these reasons, the results presented are for 10 bin ISNNCs with directly-calculated (“perfect”) Local Orientation and a MLP classification module. The MLP classifier had a  $3 \times 10$  node input layer, a 15 node hidden layer, and a 26 node output layer. This classifier has only 881 parameters to be estimated, a reduction of 99% compared to the TNN linear classifier. This translates to a dramatic reduction in both the storage space and the training time required. Only five TNNs were trained, partly because of the training time needed, and partly because the results were so consistent.

#### 7.4.2.3 Results for Traditional Neural Networks

It might have been expected that the TNNs would perform better on this task than on the task with the synthetic data described in Section 7.4.1.3, since this training set does indeed contain differently transformed versions of the canonical untransformed characters. As can be seen from Table 7.6, the results are in fact similar, and slightly worse in this case (average best percent correct of  $13.932 \pm 0.300$  compared with  $15.000 \pm 0.454$ ). Although better than chance (3.85% correct), these results are no where near usable for an optical character recognition system.

#### 7.4.2.4 Results for Invariance Signature Neural Network Classifiers

The results obtained with ISNNCs attempting to classify the characters into 26 classes appear in Table 7.7. These results show that the ISNNCs achieve a much higher correct classification rate on the test set than the TNNs. The failure of the ISNNCs to achieve 100% correct classification of the training set is in fact not surprising. The test and training sets used for this problem

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	100.00	13.675	100.00	11.111
Network 2	100.00	14.103	100.00	11.111
Network 3	100.00	14.103	100.00	10.684
Network 4	100.00	14.103	100.00	10.684
Network 5	100.00	13.675	100.00	11.111
$\mu \pm \sigma$	$100.00 \pm 0.00$	$13.932 \pm 0.300$	$100.00 \pm 0.00$	$10.940 \pm 0.209$

Table 7.6: Classification performance (% correct) of Traditional Neural Network Classifiers trained for 200 iterations with the data described in Section 7.4.2.1.

have each individual character of the alphabet mapped to a separate class. Yet, as was discussed in Section 7.4.1, the sets of characters  $\{b, d, p, q\}$  and  $\{n, u\}$  are identical under rotations and reflections: transformations under which the ISNNC output is invariant. The expected training set performance for noise-free data is thus  $100 \times (20 + 0.25 \times 4 + 0.5 \times 2) / 26 = 85\%$  correct. Any performance on the training data better than this must be the result of the fitting of noise in the training data.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	95.726	72.650	95.726	71.795
Network 2	96.154	72.222	96.154	70.940
Network 3	96.154	73.077	96.154	69.658
Network 4	95.299	73.932	95.299	70.513
Network 5	94.872	71.795	94.872	71.368
Network 6	95.726	72.650	95.726	68.803
Network 7	96.154	73.504	96.154	69.658
Network 8	97.436	72.222	97.436	70.513
Network 9	94.444	74.359	94.444	69.658
Network 10	96.154	70.940	96.154	70.940
$\mu \pm \sigma$	$95.812 \pm 0.783$	$72.735 \pm 0.971$	$95.812 \pm 0.783$	$70.385 \pm 0.877$

Table 7.7: Classification performance (% correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 40000 iterations with the data described in Section 7.4.2.1.

In order to assess the magnitude of this effect, training and test sets were created in which  $\{b, d, p, q\}$  were assigned the same label, as were  $\{n, u\}$ . The results are shown in Table 7.8. These results show that the average final test set performance was improved by 14.8% by this re-labelling, which is very close to the maximum possible 15.4% achievable if this were the only source of error.

The residual difference between training and test set error is generalisation error, rather than invariance error. These networks were trained with only 9 examples of each character, and these examples are quite noisy.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	98.718	87.179	98.718	87.179
Network 2	99.145	83.761	99.145	82.906
Network 3	99.573	86.752	99.145	85.897
Network 4	99.573	86.752	99.145	85.043
Network 5	98.718	88.034	98.718	88.034
Network 6	99.573	85.897	99.145	85.470
Network 7	100.00	85.897	100.00	83.761
Network 8	98.291	86.325	98.291	85.043
Network 9	97.863	86.752	97.863	85.043
Network 10	99.145	84.615	99.145	83.761
$\mu \pm \sigma$	$99.056 \pm 0.628$	$86.196 \pm 1.179$	$99.056 \pm 0.628$	$85.214 \pm 1.483$

Table 7.8: Classification performance (%) correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 10000 iterations with the data described in Section 7.4.2.1, modified to label characters which can be transformed into each other as the same character.

There is the unavoidable quantisation noise, but there are also some quite marked artifacts, such as loops introduced by the thinning algorithm, one of which can be seen in pattern a03 in Figure 7.23. There are also two erroneous test patterns: the result of clipping in the segmentation process. These were left in, as such errors can and do occur in practical applications.

Misclassifications							
$d \rightarrow n$	$f \rightarrow t$	$f \rightarrow j$	$i \rightarrow l$				
$i \rightarrow l$	$i \rightarrow l$	$j \rightarrow r$	$k \rightarrow f$	$k \rightarrow r$	$k \rightarrow f$	$l \rightarrow i$	
$m \rightarrow h$	$n \rightarrow b$	$n \rightarrow b$	$q \rightarrow n$	$r \rightarrow f$	$r \rightarrow y$	$r \rightarrow k$	
$r \rightarrow i$	$t \rightarrow f$	$t \rightarrow f$	$t \rightarrow f$	$t \rightarrow f$	$u \rightarrow h$	$w \rightarrow m$	

Table 7.9: Failure analysis for Network 5 from Table 7.8.

The errors made by the ISNNCs are not random. To illustrate this, a failure analysis is presented for Network 5 from Table 7.8, showing how the test patterns were misclassified. Patterns which are perceptually similar are responsible for many of the misclassifications. This means that prior information about likely errors could be used in conjunction with these classifications to aid error correction. This failure analysis indicates that the ISNNCs often appear to make errors that are very “human”, which is a promising indication that the Invariance Signature measures contour similarity in a way similar to the (unknown) measure used by humans. Inspection of the actual thinned patterns used for training and testing the networks indicates that the patterns for the letters {i, j, l} are little more than straight lines. If these were to be re-labelled as the same character, final test set performance for Network 5 would improve to 91.026% correct.

If the same were done for  $\{f, t\}$ , which are also extremely similar, test set performance would be 93.162% correct.

The misclassifications of patterns to the classes  $b$  and  $n$  may be due to the fact that the re-labelled dataset implies a non-uniform prior probability distribution for these classes:  $b$  was used as the target class for input patterns corresponding to  $\{b, d, p, q\}$ , and thus occurs four times more frequently in the training data than any other class. Similarly,  $n$  was used as the target for inputs  $\{n, u\}$ . Networks will tend to “guess” these classes more frequently than the others [20].

In the context of the extremely small training set, this is a remarkable result, comparable with character recognition results achieved by others with thousands, rather than tens, of training patterns. Such re-relabelling is in any case an essential feature of a *truly* invariant optical character recognition system, since some characters are inherently ambiguous under rotation and reflection. Others are extremely similar, and noise can render them indistinguishable. In any real optical character recognition system, a dictionary is used to verify recognised words, and context is used to correct erroneously labelled patterns. Another possibility is that the orientation of correctly classified unambiguous characters could be used to infer the correct labelling of ambiguous characters.

Perhaps most importantly, these results show that ISNNCs can classify correctly patterns which have been transformed by arbitrarily large amounts. The shifts and rotations of the input images are not restricted to small perturbations of the example patterns. This indicates that the ISNNCs are performing truly invariant pattern recognition, rather than interpolation-based generalisation.

This study should be considered to be a “proof of concept”, both for the Invariance Signature as a contour descriptor, and for MBNNs which classify on that basis. It is not intended to be a large-scale experiment which corresponds to a real application. Such a study would require much greater quantities of data than are used here. We believe, however, that the experiments presented here are sufficient to demonstrate that the theoretical work of Section 7.2 and Section 7.3 has resulted in a system that is genuinely useful for robust invariant pattern recognition. The quality of the results obtained is in fact remarkable when the size of the training sets is compared to those used in other neural network approaches to the optical character recognition problem, which frequently consist of many thousands of input-output examples.

## 7.5 Conclusion

In Section 7.2, a new invariant feature of two-dimensional contours was developed: the Invariance Signature. We believe that the Invariance Signature is a powerful descriptor of contour shape, which is closely-related to

measures employed in human perception. Its application is by no means limited to neural networks.

Nevertheless, the development of the Invariance Signature was inspired by the desire to find an invariant contour descriptor which was suitable for calculation in a neural network, and which corresponded well to theories of human contour perception. Since Lie group theory provides the link between the local changes in the positions of points under the action of a transformation and the global specification of the transformation, it provides the natural starting point. The Invariance Signature is a global measure of the degree of invariance of a given contour with respect to a set of Lie transformations, which, however, is constructed from *local* calculations. It is this that makes the Invariance Signature so attractive for use in an MBNN.

The core of the Invariance Signature approach is this: rather than seeking individual invariant features of a contour, the Invariance Signature measures the *degree to which the contour is invariant* under a transformation. The statistics of these departures from invariance are themselves an invariant descriptor of the contour.

In Section 7.3, it was shown that an MBNN could be constructed which calculated a discrete version of the Invariance Signature, and used this as the basis for the classification of contours presented at the input layer. Since the Invariance Signature is shift-, rotation-, scale- and reflection-invariant, the output of this network is *guaranteed* to be invariant under the application of these transformations to the input layer. This network is called the Invariance Signature Neural Network Classifier. The ISNNC consists of a wide variety of modules. These modules perform tasks as diverse as tangent vector estimation and the binning of data. The weights of all modules in the ISNNC except the final MLP classifier are specified independently of the training set; some are specified directly, others are determined by training the module on a sub-task. As with the weighting-function networks, the final result is just a collection of simple nodes joined by weighted connections, exactly as for a traditional neural network.

In order to be useful, the Invariance Signature must not only be invariant, but it must retain sufficient information for contour classes to be distinguished. That this is so is demonstrated in Section 7.4. The application chosen for the demonstration of the efficacy of ISNNCs was optical character recognition, the task being the classification of arbitrarily shifted and rotated lower-case alphabetic characters from a sans serif font. Two groups of experiments were performed.

The first group of experiments used noise-free, unambiguous computer-generated characters: perfect data. Networks were trained with only one upright example of each of the 22 unambiguous characters. The test set consisted of 4 differently shifted and rotated examples of each character. The best test classification performance obtained by TNNs on these data was

$15.00 \pm 0.45$  percent correct. Using NOEM (see Section 7.4.1.5), the ISNNC achieved  $96.60 \pm 0.00$  percent correct classification, and with eigenvector-based local orientation extraction,  $100.00 \pm 0.00$  percent correct: with perfect data, the ISNNC produces perfect results.

This preliminary experiment with “perfect” data demonstrated that the Invariance Signature was indeed sufficient to distinguish all the unambiguous letters of the alphabet. The second group of experiments used data obtained by scanning characters printed at 18 different orientations. Shifts were introduced by the segmentation process. These data were then thinned. The resulting data were far from perfect, the dominant problems being quantisation noise and thinning artifacts. The data set was divided into a training set and a test set, each containing 9 examples of each character. Trained with this data, the TNNs achieved a best performance of  $13.93 \pm 0.30$  percent correct. The ISNNCs obtained  $72.74 \pm 0.97$  percent correct on ambiguous data, and  $86.20 \pm 1.18$  percent correct on unambiguous data. Further inspection of the training and test data indicated other ambiguities. When corrected for these, ISNNC performance improved to 93.16 percent correct.

In all these cases, both for perfect data and for the scanned character task, the dimensionality of the classification module was very much lower for the ISNNCs than it was for the TNNs. For the scanned data task, the (minimal) TNN had 83018 parameters, whereas the ISNNC had only 881 parameters to be determined during training. Moreover, the dimensionality of the ISNNC classifier can be varied by the designer, by changing the number of bins for the discrete Invariance Signatures. The training time for these ISNNCs was correspondingly reduced.

These experiments indicate that the Invariance Signature can be successfully employed for the recognition of scanned characters independent of rotations and shifts, and that this technique can be implemented in a MBNN. The test set performance obtained is comparable to that obtained by others using thousands of training examples, and is remarkable considering the amount of noise present. Since the ISNNCs are guaranteed to be invariant under shift, scaling, rotation and reflection, and training set performance on the noisy data was  $99.06 \pm 0.63$  percent correct, it can be concluded that test set performance below this level is due to failure to generalise in the presence of noise, not failure to generalise in the sense of invariance. If the size of the training set were increased, test set performance could be expected to approach 100 percent correct.

It might be said that the comparisons were unfair, since the training sets used were inadequate for good TNN performance to be expected. That only emphasises one of the key advantages of the MBNN approach: large training sets are not required. Also, in every case the number of parameters required to specify the MBNN was smaller than that for the TNN, often dramatically so. The training time required was correspondingly shorter.

The MBNN approach to invariant pattern recognition has been shown to provide a framework for a network designer to compile expert *a priori* knowledge of the problem domain into a neural network. These MBNNs have guaranteed invariances, require less training data, have fewer parameters, are faster to train, and out-perform their traditional counterparts.

# Chapter 8

## ABC: Biologically Motivated Image Understanding

Garry Briscoe and Terry Caelli

### Abstract

In contrast to all previous chapters, this final discussion is more focused on mechanisms for image interpretation which are biologically-based. It naturally follows then that the processes of interest involve investigations into just how arrays of neurons, and their hierarchical and recurrent interactions in general, can be adapted to produce temporal structures which encode spatial information. Of specific interest is the Adaptive Behavioural Cognition (ABC) model. The model comprises a series of recurrent self-organising topological maps, which form a general proposal for the understanding of cognition, and the visual component of this model is the subject of this chapter.

## 8.1 Basic Physiology And Anatomy Of The Visual System

The human eye contains some 130 million rods and approximately 8 million cones. Rods, which are monochrome receptors of light, are generally found to be in the majority in creatures, such as the owl, which are active at night. Cones, on the other hand, are sensitive to colour. There are three types of cones, each sensitive to a different part of the spectrum. Their absorption curves indicate a peak wavelength of approximately 420 nanometers for the short-wavelength sensitive S cones, around 530 nanometers for the medium-wavelength sensitive M cones, and 560 nanometers for the L cones which are sensitive to long-wavelength light. Animals with a preponderance of cones tend to be active during daylight [307].

The distribution of rods and cones in the retina is very irregular. At the centre of the macula, only cones are found, with no rods. In the periphery, mainly rods are found, with only a few cones. Further, the number and distribution of cones within the retina is also very non-uniform. It is thought that of the 8 million cones, only about one million are S cones, with approximately 2.3 million M and 4.7 million L cones. S cones are scarce at the fovea, are at a maximum concentration just outside the fovea, then taper off in numbers with increasing eccentricity. M and L cones, on the other hand, show the highest concentration at the fovea before gradually decreasing towards the periphery [307].

The 130 million photoreceptors converge onto about 1 million retinal ganglion cells. Near the centre of the macula, the convergence ratio is about unity, while in the periphery, the ratio is several hundred to one. For cones, a ratio of one-to-one in the fovea decreases to about one ganglion cell for every two cones beyond about  $10^\circ$  eccentricity.

The photoreceptor responses are transferred via bipolar cells to retinal ganglion cells (RGC), whose axons project to the *lateral geniculate nucleus* (LGN). From the LGN they synapse onto relay cells, and then project to area V1 of striate cortex.

The rods and cones are combined in such a way that the ganglion cells are responsive to particular *receptive fields*—small regions of light that exhibit a central region of excitation (or conversely inhibition) surrounded by an annulus of inhibition (excitation). Receptive field sizes vary systematically with retinal location—within the macula some are of the order of only 0.01 millimetres, whereas only 10 millimetres away from the fovea the receptive field centres may be up to 50 times larger. At each retinal eccentricity there are multiple receptive field sizes.

Two major pathways originate from the retinal ganglion cells, producing a rich set of data pathways that interconnect a multitude of higher visual areas. These are the parvocellular (P) and magnocellular (M) streams. P cells carry chromatic information by virtue of the spectral opponency of

their receptive fields. Because of the relative numbers of cone types, most P cells have red-green opponency, but a small minority have blue-yellow opponency. The M cells exhibit only minor spectral opponency, but behave non-linearly to a spectral contrast.

Lesion studies confirm the physiological data. P-layer lesions lead to a large deficit in colour discrimination, as well as moderate deficits in form and depth discrimination. M-layer lesions produce deficits in motion, depth and form perception. The two streams contribute jointly to many aspects of perception, but provide separate contributions to the perception of colour and motion.

Of the axons of the ganglion cells that originate from the retina, about 80% project to the LGN while the remaining 20% of the optic tract projects to several structures in the midbrain, the most prominent of which is the *superior colliculus*. The superior colliculus appears to be used to detect objects away from the point of fixation, and to guide orienting movements of the eyes and head towards an observed object. It is not thought to contribute to a detailed visual analysis of objects.

At the cortical stage there is a vast expansion in the numbers of neurons, with hundreds of cortical neurons for each LGN neuron [344]. The visual cortex accounts for about 50% of the total cortex in the macaque monkey, with area V1 (striate cortex, or primary visual cortex) accounting for about 15% alone. For humans, V1 is about twice as large, but because of the ten-fold increase in cortical size over the macaque, it accounts for only about 3% of the total cortex.

The volume of the cortex dedicated to the macula, the region of highest visual acuity, has been estimated to be 35 times greater than the area of the cortex dedicated to an equal amount of retinal input from the periphery. There are approximately 1.3 billion visual cortical neurons (counting both hemispheres), which corresponds to nearly 600 cells for each LGN input [345].

As many as 32 separate visual areas have been isolated in the macaque which are largely or exclusively visual. An additional half dozen or so areas receive strong visual inputs, but also receive major inputs from other modalities. The total number of corticocortical connections may be as high as 305, nearly a third of the total possible number of pathways interconnecting these areas. A more detailed description of the currently postulated areas and connections is given by Van Essen and DeYoe [345].

Figure 8.1 shows the sub-cortical (RGC and LGN) and cortical streams for the macaque monkey. The three sub-cortical streams are, in order of prominence, the P, M and K streams. The two most studied, the P and M streams have been described previously. The K (koniocellular) stream is the least understood, despite the fact that K cells are as numerous as M cells. K cells project to the blobs of V1, and receive selective inputs from the superior colliculus. K cells have a small size, are sparse in their

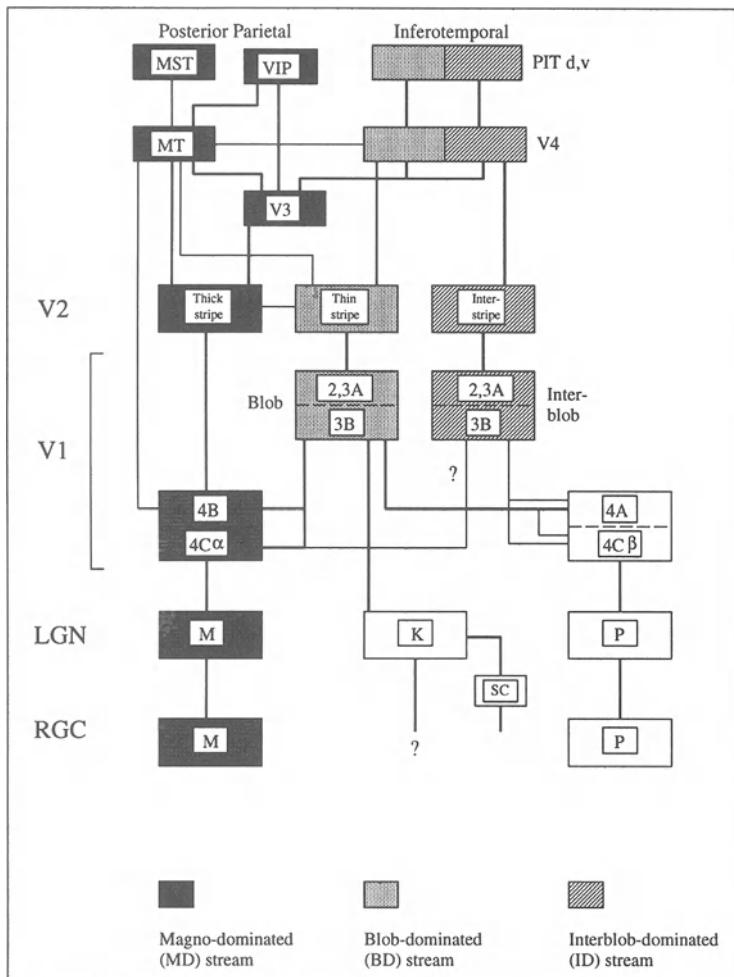


Figure 8.1: Subcortical and Cortical Processing Streams in the Macaque. (Adapted from Van Essen and DeYoe [345].)

cortical terminations, and show a sluggish response to visual inputs. For these reasons, they are thought to play only a relatively minor role in the transmission of ascending sensory information [345].

Within V1 and V2, the P and M streams are split and combined into a number of streams which project on to areas V3, V4 and V5 (MT), and then on to other areas. There are also a number of interconnections between areas V3, V4 and MT. These paths and connections are shown in Figure 8.2.

Most cells in area V1 of the cortex are orientation selective. Each cell

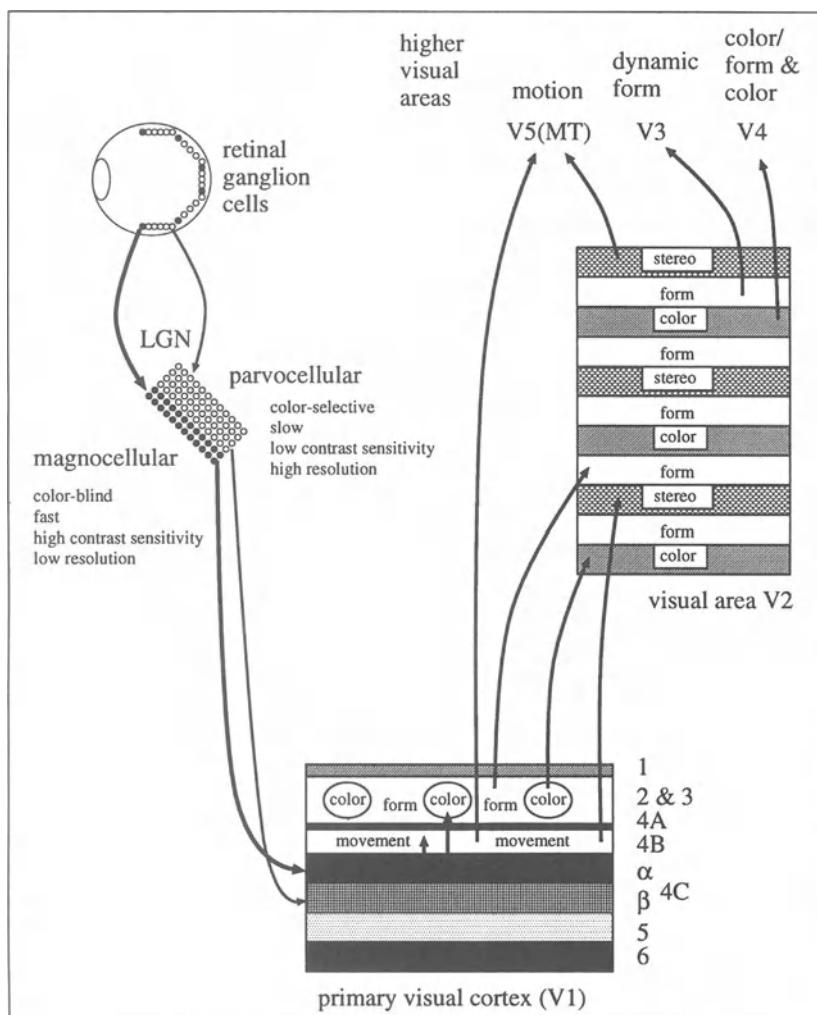


Figure 8.2: Functional Segregation In The Primate Visual System. (Adapted from Livingstone and Hubel [184]. Original copyright 1988 by the AAAS.)

responds to a receptive field at a given position in retinal space. Area V1 thus provides a ‘map’ of the retinal surface in terms of spatial frequencies and angles.

As well as the described forward connections, numerous back connections are made between cortical areas. Van Essen and DeYoe [345] cite a general principle that connections seem to be arranged in reciprocal pairs. For example, V1 projects to V2, and V2 projects back to V1. Most re-

ciprocal pathways are asymmetrical with regard to the cortical layers in which connections originate and terminate, suggesting forward pathways with reciprocal feedback connections. In the visual system, there are strong feedback connections at every level beyond the optic nerve. Lateral connections are also found. Van Essen and DeYoe have postulated 10 hierarchical levels within the visual cortex , with the two sub-cortical levels at the beginning, and projection on to the frontal cortex following visual processing. Connections occur more frequently between areas at nearby levels of the cortical hierarchy than between areas that are separated by many levels.

The functionality of higher cortical areas is still uncertain. Selectivity of various kinds, (as encountered in area V1 with orientation selectivity), is also found at coarser spatial scales in extra-striate areas. In V4, colour-selective and orientation-selective cells are reported to occur in separate clusters [376]. Motion-related selectivities have been found in areas MST, including motion characteristics relating to optic flow such as rotation, expansion and contraction [331], while area MT also exhibits motion selectivity. Area V3 has been shown to respond to spatial frequencies and angles [378].

Zeki and Shipp [377] describe V1 and V2 as segregators, containing separate groupings of orientation, colour and motion selective cells. Signals from V1 and V2 are then sent to three pivotal cortical regions: areas V3 and V3A (the V3 complex), which is involved in dynamic form analysis; area V4 (the V4 complex) involved in colour analysis in association with static form analysis; and areas V5 and V5A (the V5 complex) involved with motion.

At even higher levels in the hierarchy, neurons in inferotemporal cortex respond to complex shapes, including natural stimuli such as faces and hands [330]. For the posterior parietal cortex, the response properties suggest involvement in the analysis of spatial relationships, eye movements, and target selection for visual attention [71].

### 8.1.1 Eye Movements

Eye movements play a very important role in visual perception. The eyes saccade every 200 to 300 msec, with the saccade period being very brief—only about 10% of the total viewing time. Saccades are both ballistic and extremely rapid, with speeds of up to 1,000 degrees per second [41]. The actual saccade may take as little as 10 msec, and even a 40° saccade has a duration of only about 100 msec, much too short a time for there to be any guidance of the movement by error feedback from the retina or from proprioceptors. Thus it seems likely that saccades must be predetermined [142].

Visual sensitivity is reduced just prior to, and during, a saccade. This *saccadic suppression* severely degrades motion and position information

taken in during saccades.<sup>1</sup>

Fixations are the pauses between saccades. For the few hundreds of milliseconds of the fixation period, the gaze is virtually still with respect to the head.

The foveal area of a human eye restricts high-resolution vision to about 2° of visual angle. For comparison, this is about twice the angle subtended by a thumbnail when the hand is held at arms length (1°).

Not much is known about the relationship of eye movements to actual cognitive processes, but Viviani [347] summarises some results of eye movements which can be related to discriminable mental states.

- When subjects observe the two (or three) perceptual solutions of a polysemous figure which have well-known attractors (such as the eyes and lips in Boring's 'Wife and Mother-in-Law' image shown in Figure 8.3), their eye fixations tend to cluster around the appropriate attractors while perceiving a particular solution.
- In viewing difficult, hidden images such as that shown in Figure 8.4, in which definite contour information is lacking, a similar shift of attention occurs. As soon as the target pattern is (partially) recognised,<sup>2</sup> the eye quickly fixates the salient features of the new gestalt. The resultant fixations mean that some hitherto meaningless blobs now acquire a representational meaning, and the new, emergent configuration is confirmed.
- In observing technical images such as chest X-rays or radar displays, skilled professionals immediately concentrate their fixations on certain details which are only meaningful and important in the context of an expert, conceptual description of what is represented [59]. Laymen, on the other hand, tend to scan the whole image in a random, uniform way.<sup>3</sup>

## 8.2 Current Theories of Visual Recognition

People are able to perceive and move around in a dynamic world without any difficulty, but computer vision systems, in the main, require extremely

---

<sup>1</sup>Note also that the spontaneous activity of V1 neurons is suppressed according to the onset time of saccades. The suppression begins about 20-30 msec after the saccade is initiated, and lasts about 200 msec [92].

<sup>2</sup>'Pops out' is the term often used. The scene in Figure 8.4 is a dalmatian dog approaching a tree. On first viewing, it is very difficult to decipher the image, but having deciphered the image once, it is virtually impossible not to pick out the dog and tree almost immediately.

<sup>3</sup>Similar differences in eye movements between experts and novices occurs in the analysis of chess positions [81].



Figure 8.3: 'Wife and Mother-in-Law' Polysemous Image

limited and static environments, usually requiring large amounts of computer resources and background knowledge.

Most theories of shape recognition postulate some form of internal representation (or a set of representations) for each object. The representation contains information about the shape and other properties of the object, including a label—a name for that object. The aim of the system is to be able to correctly retrieve that label during recognition. Representations are stored in long-term memory as separable, (usually symbolic) wholes, one (or a set) for each object or class of object. Some form of *learning* of the representations by inductive methods is currently being explored (see previous chapters).

Current models usually do not postulate a representation which is a direct replica of the retinal stimulation. Rather they introduce some form of representation which attempts to capture the supposed invariant properties of each object in various positions, sizes, rotations, and even under various lighting conditions. During recognition, the retinal image corresponding to the unknown object is converted to the same format, and the representation that provides the best match using some form of similarity measure is taken to be the object recognised.

Each theory may make different assumptions regarding the type of representation used (e.g., feature space, geons, predicates, graph), the number of representations per object (e.g., store one composite 3D representation or multiple 2D representations obtained from different viewing positions), the breakup of objects into classes for mapping into representations, the inclusion of spatial relationships between objects and their component parts



Figure 8.4: Difficult Image

(relational or propositional representations), the amount and type of pre-processing given to the initial retinal image (e.g., various forms of edge detection, filtering, and contrast enhancement), how the matching is to be performed (e.g., sub-graph isomorphism, decision trees, feed-forward neural net, etc.), or a suitable metric for similarity matching.

Most adherents to the computational view of visual cognition would claim that the primary issues in computer vision (as well as other artificial intelligence domains) are representation and search—how to form an appropriate representation for objects, and how to then efficiently search these representations for a match at recognition time. In the next few sections we briefly examine various representations used in the majority of traditional theories.

### 8.2.1 Template Matching

Template matching is the simplest form of representation in which a replica of the retinal stimulation pattern projected by a shape is stored in long-term memory. The recognition process simply compares all stored object templates with the input array, selecting the best match based on, say, the ratio of matching to non-matching points.

There are, however, many problems with this method: partial matches can give false results (e.g., an ‘O’ in a ‘Q’ template); any change in the distance, location or orientation of the input object in relation to the corresponding stored object will produce a different pattern, thus preventing recognition; and any occlusion, shadow or other distortion of the input object may also produce inaccurate matching.

Some systems, for example, attempt to compensate for these problems by storing multiple templates, each recorded at various displacements, rota-

tions and sizes. However, the combinatorics of the transformations usually prove to be unwieldy. The option of rotating, displacing or scaling of the input pattern to a canonical form before matching is also not feasible, as the required transformations cannot be known until the object is recognised.

But by far the major problem with this representation (and in fact most of the representations that we will discuss) is that it is only appropriate for an object recorded in isolation. If multiple objects are present in a scene (and there are very few realistic situations in which this is not the case), then the template models fall down. The method is unable to determine which bits belong to which object—without, obviously, first recognising the objects.

It is also not reasonable to postulate some pre-process which is able to segregate figure and ground by employing depth information to separate objects. This is not only too complex, but also presupposes the identification and existence of separable objects. The differences in intensity supposedly due to depth (or a change in material) may in fact be due to differences in orientation, pigmentation, shadows, surface scratches or specular reflections. It is only by presupposing the existence of particular objects that these sources of intensity change can be separated. *Depth from shading* and other ‘depth from …’ methods, while providing valuable cues, are insufficient in themselves to separate figure and ground.<sup>4</sup> Most systems which purport to separate figure from ground are rather artificial and rely on flat lighting and relatively neutral backgrounds (ground).

### 8.2.2 Feature Models

Instead of storing templates for entire shapes, the feature model method utilises a series of mini-templates or *feature-detectors*. Generally the features included are of a geometric type, such as vertical and horizontal lines, curves and angles. Feature detectors may be located at every position in the input array, or more global feature detectors may be used.

In the case of multiple feature detectors, a calculation is made of the degree of match between the target feature and each section of the input array. The levels of activation for each feature may then be summed across the input array, or the number of occurrences of each feature counted, thus providing a set of numbers, one for each feature. This list of numbers, in the form of a vector of weights for the different features, or a list of the feature occurrences, is used as the stored representation of the object. The intention is to give each shape an invariant representation as each feature is independent of location.

---

<sup>4</sup>It is also reasonable to ask just what is meant by figure-ground. Most real-world scenes, which present no problem to human and animal observers, are made up of numerous components at various depths. Consider the case of observing a bird in a tree—what is the figure and what is the ground? The option of recognising objects that are isolated on a neutral background is simply not realistic.

Recognition again consists of finding the best match between the stored representations and the levels of activation of the feature detectors in the input. The comparison to find the best match may use the product of the two vectors, or simply count the number of matching features minus the number of mismatched features. The object with the highest match is the shape recognised.

There are a number of problems with this model including: the representation is not invariant to scale or rotation; spatial relationships are not considered—that is, the relative locations of features are not recorded, only the presence or absence of the feature (for example, in recognising a face it may be important to determine the position of the eyes relative to the nose and mouth); and the choice of features may be an issue in that natural shapes are not composed of simple lines or curves.

### 8.2.3 Fourier Models

In the Fourier representation model, a two-dimensional input intensity array is subjected to a spatial Fourier analysis in which the original array is decomposed into a set of spatial frequency components of various orientations and frequencies (sinusoidal gratings). Amplitude and phase are both recorded for the spectrum of spatial frequencies and angles.

The original input image is thus represented as the sum of the spatial frequency components, and this Fourier transform retains all of the information in the original image (given no restriction on angles and frequencies, and no computational problems such as *aliasing*).

Each shape is stored in memory in terms of its Fourier transform, and recognition proceeds by matching these stored representations with a similarly transformed input image.

Some form of *pattern* and *texture* recognition may be possible, however, because of the particular peaks and other structures found in the transformed image that correspond to the period of repetition of the pattern. The transform process is also useful in that it separates information about sharp edges and small details from other information pertaining to gross overall shape. Techniques such as edge-detector filters and convolutions may be used to tease apart these different details of the original image. The method also overcomes the problem of trying to match blurred edges, wiggly lines, and other slight distortions by means of edge enhancement and smoothing.

While initially giving promise, the Fourier model is not without significant problems. For example, the so-called invariances only hold for entire scenes and isolated objects. If a number of objects are found in a scene, then any rearrangement of the objects will drastically alter the transformed image. The frequencies due to particular objects cannot be isolated in the transform, with the component frequencies of all objects simply added to

gether. Thus the transform represents aggregated information for the whole scene, with no way to disambiguate the components.

Further, the inability to separate components of the transformed image makes it difficult, if not impossible, to recognise familiar objects in novel scenes by attempting to match transforms of isolated objects in memory.

Although the amplitude spectrum contains shape information, and the phase spectrum contains position information, there is no method of combining this information (short of performing a reverse transform to the spatial domain) in order to locate a particular objects at a particular location.

Psychological research suggests that the visual system does partition the information in the retinal image into a series of channels, each specific to a certain range of spatial frequencies [55]. This perhaps explains why the Fourier transform representation is so accepted, especially by many researchers in psychophysics and visual physiology. However, as pointed out by Pinker [256] “... *filtering* the image according to its spatial frequency components is not the same as *transforming* the image into its spectra.”

The filtering alternative has the advantage that the original image may be split into a number of separate sub-images, each filtered at particular spatial frequencies. Each sub-image then allows an analysis of the original at different scales. The processing is still performed in the spatial domain, but each sub-image array is the result of a bandpass filtering of the original image. This allows a segregation of gross shape from fine detail, but as the analysis is performed in the spatial domain it can still find correspondence between the parts of the representation and the parts of the scene.

### 8.2.4 Structural Models

Structural models came about because early representations were unable to include *relational* information—information about the relative positions and relationships *between* parts of an object. The representation held in memory is a *structural description*, that is, a data structure which maintains a list (or tree, or graph) of predicates. A simple example might be:

```
eyeshape(X), eyeshape(Y), noseshape(Z),
distance(X,Y,2.1), between(X,Y,Z), below(X,Z), ...
```

The arguments of the clauses correspond to parts, while the predicates correspond to properties (features) of the parts and to spatial relationships among them. The representation is often depicted as a graph where nodes correspond to the parts or the properties, and the edges correspond to the spatial relations [211, 249, 363].

The structural representation is purported to be able to factor apart the information in a scene without necessarily losing information in it. This enables the representation to not only be able to supply a list of labels for

the objects in a scene, but also how they are oriented and where they are with respect each other and to the human observer.

By specifying the shape of the object in one set of logical clauses, and its location, orientation, size, and spatial relationships with other objects in another set, various spatial reasoning operations may be performed that selectively access the relevant information that pertains to a particular processing operation. Further, the recognition problem may be successively decomposed into simpler subprocesses. Statistical and logical operations may be included to further enhance the recognition process; for example, a designation of what *must* be found in an image to ensure the recognition of an object.

So far so good, but the appeal of this method is actually an illusion. Structural representations simply denote a theory of representation—it tells us nothing of the actual process of recognition. Structural modelling is only appropriate (if at all) *after* recognition has taken place using some other method.

Some may argue that other processes may be used for the recognition of parts, and then structural representations and analysis may be used to ‘assemble’ the parts into recognisable objects. However, this still leaves open the very real problem of finding the ‘parts’. It also leaves open the question of determining a suitable ‘assembly plan’, whether this is taken to be innate (or created by the programmer in the case of computer vision) or learned by some inductive method. Neither problem has been solved to any degree despite numerous attempts.

Other problems with this representation question its biological acceptability. The approach is based on a type of formalised semantics. But where do the semantic concepts originate? That is, what is the origin of the predicates? Are they innate? Where do the concepts ‘above’, ‘left’, ‘between’, and so on actually come from?

Further, given that the concepts are semantic and hence culturally determined, how does this theory relate to animal vision? Presumably animals do not share these ‘linguistic’ descriptions. And infants? And how might a new concept be incorporated into the scheme? Or how might existing terms be combined to form new semantic descriptions?

The use of structural descriptions seems to be driven purely by computational convenience, and has no biological support whatsoever.

### 8.3 The Computational Theory Of Marr

The work of David Marr [194–196] is perhaps one of the best (and most detailed) examples of the computational approach to the recognition problem. It also probably remains the most influential contemporary model of three-dimensional shape recognition. Although many researchers may not accept

the specifics of the Marr theory, there is nevertheless a tacit acceptance of his overarching claims.

### World View & Representation

The goal of the visual system is to provide to the rest of the brain with a *full representation* of the visual world as observed by the creature at that instant. The world view includes all of the panoramic scene (of 180° or more) that is perceived by the eyes.

### Hierarchical Processing

There is a hierarchy of stages through which the visual signal processing must proceed—from the retina, through the LGN and then on to higher cortical stages. A typical scheme is shown in Figure 8.5. At each stage, the system builds up a representation of each object by extracting increasingly specific features, which are then combined into a fully elaborated representation specific to the object. It is only at this final stage that object recognition actually occurs—that is, visual learning occurs at later rather than earlier stages.

### Dependency Relations

The flow of information in the system hierarchy is one-way. Higher levels in the hierarchy receive input from lower levels, but not, in general, vice versa. Some specific processing stages are considered to be part of the early (low level) visual system (for example, edge finding, stereo correspondence matching, shape from shading, motion), and these do not require any input from later stages such as segmentation or pattern recognition.

### Levels Of Description

Marr made the claim that any “algorithm” of perception could be understood at three different levels:

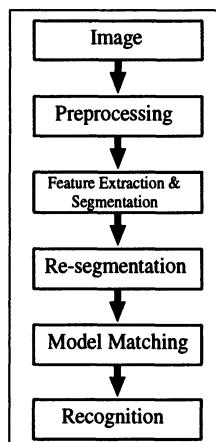


Figure 8.5: Typical Computational Vision System Structure.

**system level:** an abstract statement of the problem,

**algorithm level:** the actual steps used in the process,

**hardware level:** the actual hardware used in the implementation.

Marr suggested that these levels were independent and that the solution to the perceptual problem could be discussed at any of these levels, without paying any attention to the others. However, various experiments show that this cannot be the case, and that “our perceptual experience of the world is powerfully constrained by the actual neural machinery, the hardware that mediates perception.” [277].

## 8.4 Overall Problems With Existing Models As Related To Biological Processes

The first comment should be that there is nothing wrong with computer models that make no claim to biological reality, but propose some engineering solution to the issue of vision. After all, aeroplanes do not need to flap their wings for propulsion. We concern ourselves here with criticism of models that do make some claim as a viable model of biological vision, especially that of human and other primates.

We first deal with some computational issues, then some objections based on epistemology, and then in the next section discuss some physiological and biological points which seem to question the viability of most current models.

### The Full Task

A major criticism of many models of human and other vertebrate vision is that they do not address the whole problem. Many models which utilise structural descriptions, for example, take as their starting point a set of semantic primitives that are *assumed* to be supplied by some other, as yet unspecified process (but see Dillon [86]). To claim that a primitive ‘feature’ such as ‘eye’ or ‘nose’ will simply fall out of a lower order process really begs the question, and places no importance on biological veracity.

A related question concerns the required output of the system—what the recognition process must provide to the rest of cognitive facility. Is a label (of unknown origin), an adequate output from a vision system? A realistic model must be able to progress from biologically acceptable visual inputs through to some appropriate behaviour following recognition.

### Representations & Shape Primitives

There is much discussion, in the computer vision domain at least, about which representation is more appropriate. The discussion generally centres around computational considerations—which provides better search efficiency, which allows more compact descriptions, which can allow relational comparisons and so on—rather than on biological considerations.

Many of the proposed representations utilise shape primitives such as generalised cones, codons, or canonical volumetric shapes such as spheres, parallelepipeds, pyramids, and cones. But these representations are only appropriate to ‘regular’ shaped objects and are very limiting for more realistic general shapes. The concept of dividing objects up into smaller component parts of ‘standard’ shapes (such as cylinders) is inadequate.

A more appropriate approach might be to question the very notions of shape and representation. The concept of ‘shape’ as a primary component of vision is suspect—most objects are much too dynamic and fractal to be covered by the simplistic notion of shape. What shape is a tree, a fence, a coastline, clouds? Yet humans and animals have no problem in recognising these ‘objects’, and discerning them in all their detail *if the need arises*. The notion of ‘object’ is not quite so atomic as we sometimes believe.

### Static Images & Re-Cognition

Most visual systems (with the exception of the Active Vision paradigm) deal only with *static* images. They attempt to extract *as much* information as possible from stationary views of the world—at *each instant* of time. Most systems use only a single image—any additional images for a dynamic interpretation of the world would require a complete re-calculation of everything.

Not only is this computationally intractable, it is also not biologically realistic. The world is a dynamic, changing, real-time environment, with a continuous flow of imagery. Humans and animals do not *re-cognise* each object at each instant of time, over and over again. The process of vision is one of *confirmation* rather than repeated *recognition*.

### Computationally Based

Current computer vision systems are essentially database solutions. The system builds up representations of world objects in the machine (either during manual construction of the program, or during learning), and then uses some search process to match inputs with representations at a subsequent recognition time. Each time the system is given the task of recognising an object in a scene, the input is put into some representation, and a search performed through the entries in the database of stored representations seeking a match.

But there is one fundamental problem with this scheme from a computational point of view—the system will necessarily be slower at recognising when there are a larger number of objects in the database.

However, this is not the case for humans. Recall times for adults are not longer than those of children, and recall times for experts are not longer than those of novices. There is in fact the opposite finding—developmental trends in perceptual functioning show a progressive increase in specificity of discrimination, an optimisation of attention, and a progressive selectivity in information pickup with a correspondingly greater efficiency in ignoring irrelevant information [119].

Computer vision recognition time is usually nowhere near the approximately 150-300 msec recognition time (and approximately 100 synaptic joins) of human recognition [103]. Humans can recognise information about facial expressions as reliably from a 20 msec view as from much longer exposures [315]. Recognition of simple scenes and familiar faces may actually occur at the first glance [27, 28, 97, 149].

In all, then, the (serial von Neuman) computational approach simply does not have the right profile to be taken seriously as a model of human vision.

### Machine Learning

Most current vision systems take no account of previous history, of learning, or of context. Some are now attempting to incorporate machine learning (see previous chapters), but because of the nature of symbolic processing and learning by examples, this presents a number of problems.

Machine learning employs *induction* over examples as its principal mechanism of inference and generalisation. It thus has to contend with all of the open-ended problems associated with induction. How are the terms to be grouped for a more general rule? Should the *most* or *least* general generalisation be used? What *heuristic* generalisation technique should be used—minimum message length, entropy, Occam's razor, and so on? Each of these methods leads to different generalisations, and in principle there are an infinite number of generalisations that are possible.

Some minor use of *deductive* methods is made with explanation-based learning—but this requires a priori 'rules' which may then be put into a more operational form.

The problem is that of symbolisation. Machine learning requires the prior existence of symbols assigned to parts or features, and the process of machine learning is the forming of combinations of these symbols to describe an object.

### Segmentation

Most current models of vision use segmentation as the primary means of dividing the world into separate objects or parts. But segmentation is an inherently difficult task, especially when there are many objects in a scene partially occluding one another.

Segmentation typically tries to statically recognise an object in a scene by finding light intensity or depth discontinuities, so that the scene may be reduced to a series of lines and ultimately connected regions. The result (using natural scenes) is usually a confusing tangle of disconnected regions which is difficult to aggregate into separable real-world objects. Segmentation is invariably the bane of any artificial vision system when applied to anything other than simple, regular-shaped objects in limited numbers.

The problem for any segmentation system is that global information is required to make decisions at the local level concerning what goes with what. Some developers have attempted to supply such *background knowledge*.

*edge* as a built-in component, but without explaining where this knowledge originated.

In the hierarchical structure of conventional vision systems, segmentation is typically attempted prior to recognition. It is thus critical to correct recognition in this architecture—if the segmentation is poor, then the subsequent recognition is doomed.

### **Object-centred vs. Concept-centred**

Most systems are *object*-centred, and consider that the object is the basic unit of recognition and identification. But the world is not composed of ready-made labelled objects waiting to be discovered. The task of the visual system (as well as the other senses) is to somehow separate the world into separable objects or events, so that appropriate behaviour may be performed upon their subsequent recognition.

But what constitutes an *object*? When, as infants, we start to recognise objects, do we have the same criteria to ‘carve up the world’ into objects that we do in later years? Clearly the existence of objects is ill posed. Current vision systems, in pre-selecting and naming the ‘objects’ they want to recognise, are assuming the result they are trying to achieve.

The existence of objects is determined by a need to know. Birds are able to recognise and utilise trees, but other than their differential use of certain components of a tree, the bird would not be aware of the separate ‘object’ status that humans attribute to roots, leaves, boughs, bark and so on. The bird does not need to differentiate these objects.

All that is available to infants are the perceptual inputs as yet not separated into object correspondences. The task of the infants brain is to (self-)organise these inputs into suitable *concepts* that may then be associated with particular external objects and events.

### **What Constitutes a “Part”?**

Many computer models break up objects into component parts in the belief that recognition by parts is an easier problem. Apart from the veracity of this belief (detecting these ‘parts’ may be no less difficult than recognising the whole), there is also an important epistemological issue here—exactly what constitutes a part?

For example, a division of a tree into parts such as ‘trunk’, ‘root’, ‘leaf’, ‘bough’ is not only culturally relative, but is arbitrary both in the linguistic terms used and in the “carving of nature at her joints”. What constitutes a ‘part’ for an expert botanist when recognising a tree is very different from what may constitute a ‘part’ when the same tree is recognised by a non-expert at plant classification.

How is the botanist’s representation different to that of the non-expert, and how do we move from one representation to the next with learning? Does the botanist have a different object representation for each of the different sub-species that he/she recognises? And how does recognition by parts apply to animals who do not (supposedly) make use of such arbitrary

divisions in their visual recognition. Yet animals, such as birds, are quite able to recognise and utilise a tree.

Again it seems the method has little psychological backing and is introduced only for computational ‘efficiency’.

In many artificial vision systems, the use of ‘parts’ presupposes recognition. Part labels are assumed to be provided by some lower-level process. Once given labels for parts in some symbolic form, the process of ‘recognition’ is reduced to a form of symbol matching—typically using computational devices such as decision-trees, graph matching, and so on.

### Labels

The role of most computer vision systems is to produce a *label* to indicate recognition of an object. These labels are provided to the system *a priori* with no concern as to their origin.

But the production and assignment of labels to objects in the world is a dynamic process which occurs every day. Experts in various fields give labels to newly discovered or invented objects, new marketing and slang terms are invented, and new words are applied to different ‘carvings’ of existing objects.

While the world may have been initially divided into different concepts on the basis of simple behavioural needs—we learn to differentiate brightly coloured frogs from dull ones because the brightly coloured ones are poisonous while the dull ones are edible—the use of labels extends our ability to divide the world into ever finer concepts. Whereas prior to labels each individual had to rediscover these linkages between different sensations and appropriate behaviour, the use of labels allows easier conceptualisation and differentiation through explicit training.

For example, two insects may appear the same to an untrained eye, yet a research entomologist, in associating new labels to specific sensory differences between the insects, allows us to separate the two species within the genus.

A model of cognition must include the provision for the creation and allocation of new labels, much less the problem of language.

## 8.5 Biological Considerations

There are many examples of experiments which tend to question the traditional view. Churchland et al. [67], and several preceding papers by Ramachandran and his coworkers [274–279], report on a number of experiments which bring into question the serial hierarchical view in which there is a one-way flow of information from *early processes* (such as edge detection, shape from shading, stereo matching) to *later processes* (pattern recognition). The experiments provide evidence for a much richer, more interactive perceptual system, and provide strong neurobiological and psy-

chological plausibility to the need for an overhaul of the traditional views.

Indeed, the adaptive systems discussed in the previous chapters are some initial attempts to install learning, adaption, feedback and recurrency into machine-based image interpretation systems.

### 8.5.1 Hierarchical Processing & Priming Effects

The traditional flow diagram for a typical computer vision system resembles that shown in Figure 8.5. The process is a top-down hierarchical one, with only forward interactions between sub-processes. Unfortunately this model is not tenable for biological visual systems for a number of reasons.

One such reason is the observation that many recognition tasks are *primed* by familiarity and context. A number of experiments dealing with eye movements are significant. Compatibility effects result in reaction times which are shorter when stimulus and response are spatially homologous than when they are not [132]. Preparatory effects are seen when a prior signal pre-cues a required movement, reducing its latency. For example, a non-specific visual warning before a target onset reduces saccadic latencies [296].

There are a number of priming effects that are observed which involve figural and linguistic stimuli. For example, a letter string can be presented in the peripheral field to facilitate the naming of a visually similar word which appears later in foveal vision [280].

Similar priming was found when simple line drawings were used instead of words [263]. Subjects were shown a line drawing of an object in peripheral vision. During a saccade to the object, the initially presented picture was replaced with another picture that the subject was instructed to name as quickly as possible. Strong facilitation (150 msec) compared to a control condition occurred if the second picture was identical to the first. Facilitation also occurred if the two pictures were conceptually similar, for example two different pictures of a horse (90 msec) or visually similar (e.g., a ball and a tomato).

### 8.5.2 Adaptation to Discordant Stimulation

It is generally assumed that the sub-cortical sensory systems responsible for the initial coding of stimulus attributes are ‘hard wired’ in the human neonate. For instance, the retinotopic projection of ganglion cells to the visual cortex is fixed before birth, and is normally not affected by subsequent environmental influences. However, other sensory mechanisms, such as the orientation detectors of the visual cortex, pass through a critical period in early infancy. During this development phase, environmental influences may indeed modify the orientation detectors, as was shown in the numerous experiments performed on animals which were raised in artificial environ-

ments (see Howard [142] for a review). For example, Blakemore and Cooper [34] report an experiment in which kittens were raised for several months in an environment which consisted entirely of vertical or horizontal stripes. Careful behavioural studies showed that the cats suffered a permanent loss of visual acuity to lines at orientations other than the one to which they had been exposed. Similar experiments with normally-reared adult cats do not produce comparable results, so that following development, the orientation detectors remain largely immune to such influences.<sup>5</sup>

Similarly, the basic machinery of the motor system is laid down in the embryo or in early infancy. Nevertheless, the spatial coordinated behaviour of adult human beings is remarkably flexible and readily adjusted to suit changes in stimulus conditions. This flexibility seems to arise in the processes which relate one sensory system with another, or which coordinate sensory events and motor responses, and not in the basic sensory or motor mechanisms themselves.

A number of studies of the adaption of the visual system to discordant stimulation are discussed by Howard [142]. The classical study is that of Stratton, who, for a number of days, wore spectacles which inverted and reversed the visual scene [326, 327]. Stratton found that his actions gradually became more smoothly coordinated over time, although each one had to be mastered separately. He found himself gradually adjusting to the conflicts produced by the inverting lenses, and people and objects that were in his surroundings began to look real, rather than incongruously reversed. Early on in the period of adjustment to the inversion of the visual image, Stratton reported that there appeared to be a *double representation* of the felt position of a limb being viewed, but that the older representation weakened progressively, especially during activity.

Various studies have shown adaption to displaced vision with the use of attachments such as a prism [142]. Following a period of inaccuracy, the subject is able to adapt and correctly locate objects. When the prism is removed, a further period of adaption to reverse the prism effect is required.

Numerous other examples of imposed transformations of the visual stimulus array are described by Gibson [119], including displacement, inversion, false colouring, minification, magnification, enhanced binocular disparity, and displaced auditory stimulation. In all cases for which the sensory alteration was systematic and continuous, the subject's perception underwent some form of adaption in the direction of a realignment to their prior unmodified world-view. Further, when the modifying instrument was removed, a further period of reverse adaption was required before their perceptions returned to normal.

---

<sup>5</sup>Mize and Murphy [214] suggest a note of caution in generalising these findings. They found that rabbits have a full compliment of innate orientation-selective cells, and that rearing with vertical lines has no effect. However, Blakemore [33] has suggested that plasticity of orientation detectors is required in animals with binocular vision. While humans and cats do have binocular vision, rabbits do not.

### 8.5.3 Invariance

Many computer vision systems assume invariance to translation, rotation and scale. However, there is good evidence that at least rotational and scale invariances are not evident in human visual perception. For example, the recognition of upside-down faces is extremely poor, even for very familiar faces [372]. Thompson [336] showed that upside viewing was extremely poor at determining the expression (smiling, frowning) on a face.

In a test of spatial invariance, Nazir and O'Regan [227] exposed subjects to a completely new and unfamiliar pattern, but ensured that it impinged on the retina in only a single retinal location. Following learning, the same pattern was presented at other retinal locations. The subjects were unable to recognise the objects at the new locations, some in fact denying that they had ever seen the shape before. After a few presentations, however, subjects *were* able to make a correspondence with the initial pattern.

This suggests that some form of global operator for translational invariance is not found, but rather that the perceived translational invariance of objects is due to individual learning episodes at many retinal locations during the long training period of early life, a view shared by Hebb [133].

### 8.5.4 Saccades, Fixation & Spatial Fusion

Many papers have been devoted to the question of how the brain maintains stability of internal image representations following saccadic eye movements—a trans-saccadic fusion of visual information (see, for example, Bridgeman et al. [41]). An implicit (and sometimes explicit) assumption of most current vision systems is that the internal representation of the retinal image is somehow maintained across saccades, with the fovea providing a detailed sub-image to *fill-in* the total perceived image. For example, Potter [265] proposes a form of integrative visual buffer where information from successive fixations is pasted together to provide a coherent alignment of the individual snapshots.

O'Regan [239] points out the ‘imperfect’ nature of the image we obtain of the world.

- The retina of the human (and other vertebrate) visual system is constructed with *obstacles* in the light path (blood vessels, neurons and axons making up the early stages of the visual system, and even the body of the rods and cones themselves, which appear to be “in backwards”). The receptive system seems to be *inverted*.
- A “blind spot” formed by the optic nerve exiting the retina.<sup>6</sup>

---

<sup>6</sup>The angle subtended by the blind spot is relatively large—about 4° of visual angle (dva). It is generally assumed that the brain somehow “fills in” the missing information into the region of the blind spot, but according to O'Regan this hypothesis has not been tested seriously—but see Ramachandran [277].

- A severely non-uniform *spacing* of both the rods and cones in the retina (including *within* the foveal area) (see Section 8.1).
- A non-homogeneity of the colour receptors (cones) across the retina, as well as a thinning of the yellowish macular pigment, both producing a non-uniform colour image [240].
- Optical aberrations within the eye lens [240].
- A power difference of two diopters for red and blue light in the eye lens [240].
- Saccadic eye movements which smear and displace the image.
- Torsion (small cyclorotations around the retinal visual axis [142]).
- Various small correction movements such as micro-saccades, micro-drifts, small vergence movements and micro-tremors [380].
- Further eye, head and body movements which not only move the retina relative to the world, but are also responsible for retinal image slip during fixation.

If anything, this list understates the case. Howard [142] includes an extended discussion of the various movements, deviations, slips and twists observed in the behaviour of the retina. Yet despite all of these difficulties, we still seem to have a ‘perfect’ view of the world, an image that is stable and complete. We are not aware of these ongoing processes and imperfections.

Various solutions are proposed. For example, it is suggested [41] that information for a fusion could be provided by structural ‘cues’ in the visual world, ocular muscle input, proprioceptive input from extra-ocular muscles and motor system commands.

The displacement of the retinal image is assumed to be accounted for by an *adjustment* provided by extra-retinal signals. Each incoming retinal view is placed in the correct ‘position’ on the internal representation by allowing for any displacement caused by saccadic, eye or body movements. This is illustrated in Figure 8.6. The dotted box indicates the extent of the ‘internal representation’, and the image information provided by the three saccades indicated is positioned correctly using the displacement information.<sup>7</sup> All of the eye and body movements are somehow added up

<sup>7</sup>The actual situation is somewhat more severe than that indicated in Figure 8.6. The foveal area (corresponding to the dotted area in the figure) subtends an angle of only 2°, whereas the full visual field (and hence supposedly the internal image) is purported to be as much as 210° [213]. The postulated extra-retinal signals must account for all of the changes in the retinal frames across saccades. It would need to compensate for eye movements, head and body movements, variable focus of the eye, movement in targets and so on.

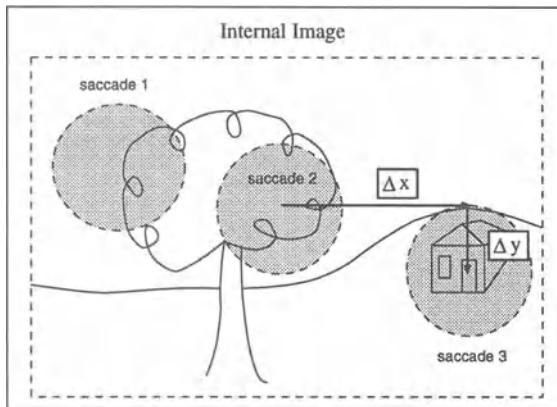


Figure 8.6: Internal Image Representation.

to produce a correcting adjustment (the  $\Delta x$ ,  $\Delta y$  in Figure 8.6) so that the foveal image is placed correctly in the overall internal world image, “guaranteeing a seamless visual percept and the ability to accurately locate objects in our environment” [239]. The process is termed “trans-saccadic fusion”.

These traditional views assume that any internal representation has metric properties like those found within the world. They pre-suppose that a picture-like image is retained in the brain. However, there are a number of other problems with this *internal screen* model including:

- A difficulty incorporating depth information into the representation. How would eye convergence, accommodation and focus be taken into account?
- What mechanism could produce the fusion given the extreme difficulty of obtaining a precise registration because of widely differing resolutions, colours, and extent (due to the non-linearity of the retina)?
- The extra-retinal signal must be perfectly accurate or else errors will build up to cause poor registration.
- What mechanism could account for head and body movements?

Most contemporary vision models make the assumption of spatial integration. They assume that information from successive fixation images is somehow brought together into a metric-preserving internal representation. The representation may be explicit, such as a photograph-like representation in some area in the cortex, or else as a more implicit representation in which higher-order representations are joined in a metric-preserving manner. In many cases of computer vision models, the issue is not even mentioned. These models begin their early processing with a full world scene,

which is then segmented to provide the initial primitives for further processing.

We could always reject the photograph-like representation for reasons of parsimony. The visual system seems to go to a lot of effort to tease apart numerous components of the image in the retina, LGN and early striate cortex (as described in the initial sections of this chapter) for it to be put back together again as a replica of the outside world, ready for subsequent processing.

Spatial integration of higher-order representations has no psychological basis, and would appear to be an immense, if not impossible computational task *even if* the appropriate ‘extra-retinal signal’ were available.

It seems clear that the various vestibulo-oculo reflexes and other extra-ocular neural signals provide general directional sense and orientation, but this is a very different proposition from insisting that they are able to compensate for retina, head and body movements sufficient to ensure a computational spatial registration of images across saccades.

More importantly, there is good experimental evidence that spatial fusion is just not performed [13, 151, 237, 340]. Thus we are lead to reject the notion of *spatial integration* of retinal images across saccades.

### 8.5.5 The World As Outside Memory

An alternate view to the “internal screen” model described above has been proposed by a number of people [120, 121, 133, 138, 189, 190, 237, 239, 240, 349]. In this view, the brain does not record internal representations (icons) of objects in the world, but rather uses the world itself as an “outside memory store”.

O'Regan [239] has conducted a number of critical experiments which cast serious doubts on the “trans-saccadic fusion” model. Figure 8.7 illustrates the setup of one experiment. Each stimuli consisted of two halves, the second half being the same for all three stimuli. The two halves, when superimposed, form the three words at the bottom of Figure 8.7. During each trial, the two halves of each stimulus were presented in sequence, and in the same physical location in space. However, the first was presented before a saccade, with the second presented after the saccade. Over a number of trials with various stimulus durations and delays, subjects were *unable* to ‘fuse’ the two apparently random patterns into a recognisable word (as would be expected if the ‘internal screen’ hypothesis was valid). Other researchers have also confirmed that no fusion appears to take place (for details, see O'Regan [239]).

Several carefully planned and independent studies have confirmed this finding [150, 281]. These results appear to rule out the strong interpretation of a visual buffer, including the perceptual availability of the result.

Experiments in which line drawings are exchanged during saccades also

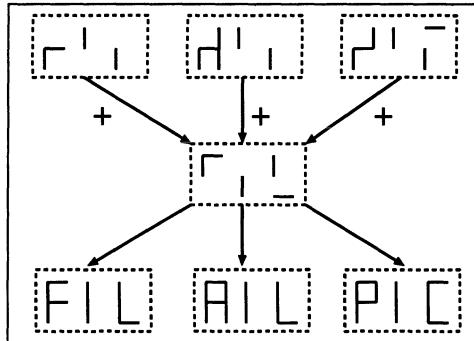


Figure 8.7: Trans-Saccadic Fusion Experiment. (Adapted from Fig. 3. O'Regan and Lévy-Schoen (1983))

indicate “that an integrative visual buffer, if it exists, plays little part in the perception of line drawings” [263]. The conclusion drawn is that “it seems unlikely that an integrative visual buffer is ever used to combine information across saccades.”

A tactile analogy is given by MacKay [189, 190]. When we explore the tactile sensations of an object with our eyes closed, we do not need to obtain a *full* and detailed tactile map of the object to identify it. A single touch or a series of cues may be sufficient. Further, the surface details corresponding to the spaces between our fingers do not necessarily need to be ‘filled in’.

It is the process of active *exploration* of the tactile sensations and the recall of previously learned sensations that enables us to recognise, and thus be able to name, the object. If an initial sensation is insufficient for recognition, further exploratory touches at other locations of ‘expected’ high discriminatory value will be undertaken until sufficient evidence is obtained. We recognise the object through a series of tactile experiences rather than first obtaining a full tactile description. In this case, there is no thought of postulating a tactile mechanism which compensates for the movements of the hand as it ‘fills-in’ the tactile representation of the object—the equivalent of the extra-retinal signal in vision.

And so it is with visual and other perceptual sensations. As stated by O'Regan [239], “... ‘perception’ is *getting to know* or *verifying* the sensations caused by possible actions.”

There is no need for a full metric (internal replica) representation of the object—all that needs to be stored is a series of linked (temporally connected) cues. The object is *in the world* for further sampling if required. Other contextual associations will also tend to support or inhibit the perceptual cues that are currently being experienced. Note that we are not suggesting that *no* representation is stored, but that the ‘representation’ is not a spatially fused iconic form. Rather, it is a temporal sequences of

sensations.

Awareness is an active process. If we are unaware of something, then it is not part of our current conscious thinking. It is only if context or other urgings (such as hunger) cause us to actively seek out some object or phenomena, or some sensation (such as motion) brings an object to our attention, that we become aware of it.

As put by O'Regan [239]

Since “seeing” involves both *interrogation* of the visual field, and also *apprehension* or *integration* or *comprehension* within the current mental framework, one would predict that a person would fail to see something either (a) if he or she does not interrogate or wonder about the appropriate aspect of the visual field or (b) if he or she is unable to integrate the obtained sensations into his or her mental framework. In particular, even if you are directing your eyes directly at something, unless (a) you are (at least unconsciously) wondering about it, and (b) you are able to apprehend it, you will not have the impression of “seeing” it.

In a series of experiments in which items on a screen were changed during a subject's saccades, McConkie [205, 206] was able to show that even obvious and large objects could be removed, changed in colour, or shifted without the subject being aware of the modification. Unless there was some cognitive reason for the subject to attend to an object (such as watching for a change), its presence or absence in the scene was not noted.

A number of careful experiments involving the reading of text on a screen provides additional support. The screen viewed by subjects contained masked text, (for example, all words X'd out), except for those words within a moving window which is tied to the subjects foveal view. This is illustrated in Figure 8.8. As readers move their eyes along the line of text, the window moves with the eyes so that actual text is always at the fovea, while the window is surrounded on both sides by masked text. The changes occur during each saccade. However, the subject is completely unaware of the masking, and maintains that the whole text remains unchanged and available [98, 219, 238, 262, 280].

The perception that we have visual access to all objects in our field of view *at once* is an illusion. We really only see what we are attending to at the time, other than some diffuse sense of other ‘objects’ being in the periphery. If we wonder about what a particular ‘objects’ might be, we then direct our conscious attention towards it by either saccading to it and thus bring it into full vision, or relying upon a previous memory of the object which can be cued by some imprecise sensations. For example, as I am writing this paragraph, I can direct my attention to the right periphery of my vision and obtain a blue sensation. I remember that I placed a blue book on my desk a few moments ago in roughly that position, and a

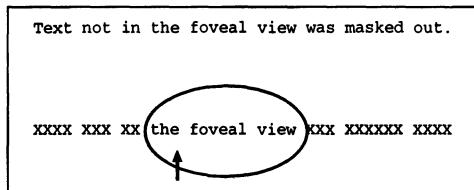


Figure 8.8: Perceptual Span In Reading. The oval indicates the ‘perceptual span’, and the arrow the centre of the foveal view. In a typical (English reading) subject, the perceptual span is about 17-18 characters, with about 2-3 characters to the left of fixation and about 15 characters to the right.

saccade in that direction confirms my recall. Other vague sensations in the periphery are ignored unless my flow of attention is directed towards them.

I can actively explore the environment to fill in my expectations and inquiry, or my attention can be directed to a location because of some prompting sensation, such as visual motion or a sound. Seeing is the action of interrogating the environment.

Visual perception is such an intensely rich sensation of *total* external presence that most people find it difficult to accept that it is not as it appears. The illusion results from the fact that we have *some* sensation over a very wide visual field, detailed information only in a very small foveal area, and an attentional mechanism that can range across the whole area very quickly. It is the flow of attention (that is, the temporal sequences of foveal inputs) that is the key to understanding visual recognition.

Churchland et al. [67] propose four reasons for our strong perception of a whole-scene visual representation: the ability to repeatedly *visit* stimuli in the scene; a short-term *semantic memory* (of a few seconds duration) that maintains a general ‘feeling’ of the overall context without creating and maintaining the point-by-point detail; the *objectification* of our sensory perceptions so that they are represented as being about some object in space; and a *predictive* element within pattern recognition based on context.

The rejection of a metric basis for vision is a difficult one. Our Western cultural heritage is very heavily influenced by metric-preserving representations such as perspective in paintings, maps, and photography. Further, metric-preserving metaphors are strong in Western thought as opposed to some other cultures for whom space is not a linear Cartesian geometry.<sup>8</sup>

### 8.5.6 Semantic Integration

Given that spatial fusion is unlikely, another possibility is that the visual system operates at a more abstract level. O'Regan and Lévy-Schoen [237]

<sup>8</sup>See, for example, a discussion by Whorf [359] on the different treatment of space by the Hopi language. Talmy [329] also discusses how languages structure space.

have put forward the hypothesis of a level of representation more general than the analog visual code available on the retina as a possible explanation of the perceived phenomenal stability and continuity of the visual world. Their suggestion is that our mental representation of a visual scene is essentially semantic rather than ‘photographic-like’.

The representation is based on labels ('tags' is their term). The image of a chair is coded as the label 'chair', with spatial relational terms (such as 'near', 'far', 'in front of', and so on) providing enough relative-position clues to guide eye movements should further spatial information need to be extracted from the visual field. This semantic representation has the obvious advantage of not requiring complex shifting or aligning of successive 'snapshots', but has serious epistemological problems.

This use of semantic labels takes no account of the source of the labels, especially in relation to animal and child visual recognition. If visual objects need to be labelled with semantic terms, what do children use before they learn the label (or name or referent) of a particular object? Do they use a place marker (if so why bother with a label subsequently) or are they unable to perceive the item until they have learned the referent? Do things pop into view when we name them?

Or is the label for each object innate? This implies that some form of object pre-recognition is found in the brain for all objects that are not only recognisable now, but also the names for all new technologies yet to be discovered—clearly another epistemological dead-end.

As we will show later with the ABC model, referents certainly do take a part in conceptualisation, but they play no fundamental part in perceptual conceptualisation or recognition. The acts of recognition and semantic naming are separated. The ABC model does enable a linkage between the *perceptual concept* (the neural attractors formed by the perception of an object) and the associated *referent attractors* (the neural attractor formed by the perceptual learning of the referent—the sound of the referent or the visual perception of the name of the object in written form), but the process proposed for visual recognition does not involve or require the labelling of objects.

### 8.5.7 Global Influences On Perception

There are several experiments which illustrate a global or contextual influence on visual perception, in contrast to the traditional view of most psychologists and computer vision researchers who generally assume that perception occurs in stages. There are very few psychophysical experiments that provide direct evidence for the traditional view, and the experiments discussed in this section indicate that the so-called early vision modules are not autonomous and do interact significantly with each other. They further suggest that higher-order processes *do* affect our perceptions.

### Bistable Motion Perception

Figure 8.9 illustrates an experiment which shows the role of top-down influences (such as attention) within perception. A set of four dots of light are alternately illuminated across the diagonal. The black dots marked A in the figure are first illuminated, then they are turned off and the open circles marked B are illuminated, and so on in a continuous cycle [278]. A number of these four-dot constructs are maintained on a screen, and the oscillating lights in each is seen as a movement of light between pairs of dots.

The striking thing is that subjects do not see a mixture of horizontal and vertical movements, but one or the other (illustrated by the arrows in the two parts of the figure). The perception is one of global apparent motion. The display is essentially bistable just as with the Necker cube. Also, like the Necker cube, subjects can focus their attention to change the direction of oscillation.

This result strongly suggests that global or higher-level considerations do interact with so-called low-level operations such as motion perception. The subjects perception of the cube changes dramatically as the brain switches between alternate *interpretations* of the figure.

### Virtual Occlusion

A variation on the previous experiment is shown in Figures 8.10. In these experiments, a shaded square is found on the right-hand side. In Figure 8.10 (a), all group A dots blink on and off, but only the upper and lower B dots alternate off and on. The perception is that the middle A dot moves behind the “virtual” occluder.

However if A contains only one dot, as in Figure 8.10 (b), the perception is not found. What is seen is a single dot blinking on and off, with a square to the right. The surrounding subjective motion is required to provide the perception.

### Cross-modal Interactions

Using the same single dot and shaded square, with the addition of a tone sounded in each ear simultaneously with the blinking of A and the virtual blinking of B, (as indicated in Figure 8.10 (c)), subjects do indeed

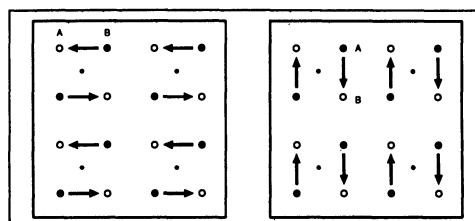


Figure 8.9: Bistable Motion Perception.

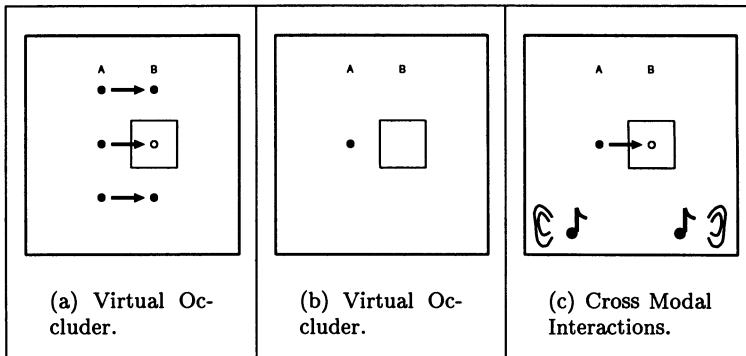


Figure 8.10: Virtual Occlusion.

see the single dot move to the right behind the occluder [67].

The integration of auditory and visual information “pulls” the dot in the direction of the sound movement. A similar (but weaker) effect can be induced if the light blinking is accompanied by a corresponding left-right vibration stimulation to the hands.

This experiment is a very convincing demonstration of the interactive and integrated nature of vision, as opposed to the traditional straightforward ‘image interpretation’ view. While integration of auditory and visual information is expected at some stage in the brain, this demonstration shows that this integration occurs in what was considered to be early visual processing.

### Shape-from-shading

Ramachandran [277] cites a number of experiments in which screen images of shaded ‘spheres’ are perceived as being either concave or convex. The perception in many cases may be reversed by an act of will, and there seems to be a built-in assumption that the light source is above.

If an image contains a number of ‘spheres’, those with shading at the top will appear to be concave, while those with shading at the bottom will appear to be convex. Turning the image upside-down will reverse the perception for individual spheres. Tilting the head in conjunction with the image retains the shape perception, but turning the image by 90° relative to the head so that the shading is from the sides (no matter what the orientation of the head itself) makes it harder to differentiate the spheres into concave and convex shapes.

Thus the assumption of overhead lighting and shape-from-shading is primarily, if not exclusively, dependent upon retinal rather than world coordinates, a somewhat counterintuitive result since it implies that when you tilt your head the visual system ‘assumes’ that the sun is stuck to your head.

Other experiments involve concave masks which are lit by sources at various directions relative to the mask. The perception is always one of a convex face, even when the illumination is from below. This perceptual persistence shows a strong top-down effect in a supposedly low-level task, namely shape-from-shading.

### Stereoscopic Depth Perception

Stereo fusion is usually regarded as an early vision task, yet an experiment described by Ramachandran [275] indicates that stereo vision also makes use of top-down global information. An image is stereoscopically fused on the basis of *subjective contours* rather than actual low-level details.

### Figure-ground

Experiments by Peterson and Gibson [254] indicate that figure-ground separation does not precede shape recognition as postulated in the traditional models.

## 8.5.8 Neuroanatomy & Neurophysiology

The experiments and observations in this section describe more the hardware implications and connections of any cognitive model of human and animal vision.

### Backprojections

The structure of mammalian cortex shows multiple forward and backward projections of axons at several levels. For example, in the monkey cortex, the various forward axon projections<sup>9</sup> are equivalent to, or outnumbered by, back projections [343, 344]. These backprojections in most cases appear not to be merely reciprocating feedforward connections, but are more widely distributed and include distribution to some areas from which they did not receive a forward projection.

Rockland and his associates [290, 291] describe the connections between areas of visual cortex of the monkey as shown in Figure 8.11.

### Illusory Contours

A number of results show that neurons in the visual cortex of monkeys respond to illusory contours.<sup>10</sup> For example, von der Heydt et al. [348] report that neurons in visual area V2 of the macaque respond to illusory contours, and Grosof et al. [129] have shown that some orientation selective

<sup>9</sup>Forward in the sense that the axons project from regions which are closer (in synaptic distance) to the sensory periphery to regions more synaptically distant; for example, projections from V2 to V4

<sup>10</sup>The question must be asked as to whether these contours are actually illusory. In many cases, spatial frequency filters *will* record them as being nearly equivalent to actual contours. They are only *illusory* given higher-level (learned) knowledge of the actual scene and objects involved.

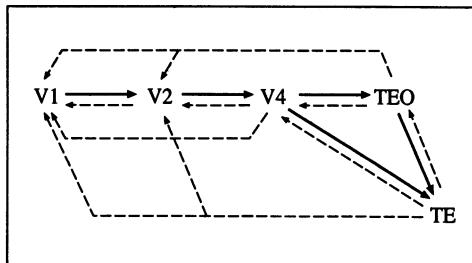


Figure 8.11: Feedforward (solid lines) And Backprojection Connections (dashed lines) In The Monkey. (From Rockland et al. [291]).

cells in V1 respond to a class of illusory contours. The detection of illusory contours depends on some interpolation across a span of the visual field, suggesting higher level operations are backprojected to lower levels.

### Cross-Modal Interactions

Most current theories of vision only deal with the dominant exteroceptive function of vision where we perceive the properties of the visual world, and fail to take account of the other perceptual modalities. That this interaction may be important is demonstrated in the phenomenon ofvection, the illusory sensation of self-motion induced by movements in peripheral vision [39]. While very little is known of the linkage between the exteroceptive and proprioceptive functions of vision, its importance should not be underestimated.

There are a number of other experimental results which show a strong linkage between various modalities. For example, the responses of cells in V4 to a visual stimulus can be modified by somatosensory stimuli [199], and a similar task-dependent modification for cells in somatosensory cortex (area S1) has been shown [116].

## 8.6 Evolutionary Considerations

The experimental results discussed in previous sections suggest that the process of vision is not a simple isolated and hierarchical stream from sensations to perception, but that other sensory modalities and motor systems play a significant role in what is literally seen.

As Churchland et al. [67] point out, one must give some prominence to the biological ‘purpose’ of vision in our considerations of its mechanisms. Improved motor control is surely the evolutionary rationale of vision—to enhance the creature’s chances of survival and reproduction through being able to visually monitor the world around it, allowing it to better seek out appropriate nourishment, to better avoid predators and other dangers, and to enhance its chances of finding a mate.

What is important to a creature at any given moment will depend upon the context, the immediate environment and the creatures current state and needs. Rather than taking in a full iconic view of the world as proposed in current vision models, it would seem that only immediately relevant information is explicitly required by the creature. The creature will tend to foveate to items of interest, and although unattended objects are represented in some minimal fashion (sufficient to guide attentional shifts and eye movements), they are not literally seen in the sense of “visually experienced”.

Interactive vision is exploratory and predictive—visual learning allows an animal to predict what will happen in the future. Further, recognition can be faster and more accurate if the animal can make exploratory movements, particularly of its perceptual apparatus, such as whiskers, ears, and eyes. New dangers and opportunities must be learned and incorporated into existing behavioural patterns.

The traditional view of vision assumes that the connection to the motor system is made only after the scene is fully elaborated. The idea is that the decision centres make a decision about what to do on the basis of the best and most complete representation of the external world. However, many situations may require action long before a complete analysis of the visual terrain can be made—motor action may be required on the basis of preliminary and minimal analysis.

Churchland et al. [67] remind us that “natural selection and reinforcement learning share a certain scientific appeal; to wit, neither presupposes an intelligent humunculus, an omniscient designer, or a miraculous force—both are naturalistic, as opposed to super-naturalistic. They also share reductionist agendas. Thus, as a macrolevel phenomenon, reinforcement learning behaviour is potentially explainable in terms of micromechanisms at the neural level.”

Using active reinforcement learning, the brain is able to build a network more suitable to the current environment, including predictive mechanisms to direct attention to what is worth looking at given one’s interests, and a close linkage between perception and appropriate behaviour. Sophisticated visual perception evolved, not as an end in itself, but in the service of better motor control.

A number of researchers have taken a more behavioural approach to vision. Their view is that the visual system is “more readily understood in the context of the visual behaviours that the system is engaged in, and that these behaviours may not require elaborate categorical representations of the 3D world.” [19]

The active vision paradigm [14, 15, 19, 32, 122] is based on the ability of the perceiving system to purposively change—for example, to change the fixation point, or the focal length, or even the location of the system within the world. Rather than attempting to squeeze every bit of information out

of every image, the active system adapts its behaviour in order to obtain information that is important at the moment. The paradigm is concerned with a continuous flow of images rather than a succession of static images.

Vision is a real-time process, not the passive process postulated by traditional vision systems. Typical short fixation times in the human visual system are about 0.25 seconds, and cortical neurons typically fire at rates of 10 spikes per second, giving 2.5 spikes per fixation [19]. Input to the visual system, then, is a series of short duration ‘views’.

## 8.7 Scanpaths

It is well documented that fixations tend to cluster around places with high gradients of change in the luminance distribution [192]. Moreover, the sequence of saccadic scan patterns for humans is both regular and idiosyncratic [231, 232]. As stated by Noton and Stark [231]:

... every person has a characteristic way of looking at an object that is familiar to him. For each object he has a preferred path that his eyes tend to follow when he inspects or recognises the object.

The eyes seem to follow fairly regular pathways, visiting the features of an object in a somewhat cyclic manner, rather than crisscrossing the object at random [231]. For example, in one experiment performed by Yarbus [371] subjects were asked to view a photograph of a bust of Queen Nefertiti. Recordings of their saccades showed that fixation on a feature, say her eye, was usually followed by a fixation on the *same next feature*, such as her mouth. The saccade path showed regularities, moving from one ‘feature’ to the next in a consistent sequence.

Work done by Noton and Stark [231] found similar results, and showed that “each scan path was characteristic of a given subject viewing a given picture. A subject had a different scan path for every picture he viewed, and for a given picture each subject had a different scan path.”

Noton and Stark proposed a mechanism involving a ‘feature ring’: <sup>11</sup>

... as a subject views an object for the first time and becomes familiar with it he scans it with his eyes and develops a scan-path for it. During this time he lays down the memory traces of the feature ring, which records both the sensory activity and the motor activity. When he subsequently encounters the same object again, he recognises it by matching it with the feature ring, which is its internal representation in his memory. Matching consists in verifying the successive features and carrying out the intervening eye movements, as directed by the feature ring.

---

<sup>11</sup>Their feature ring was essentially a circular linked list.

Our proposal is essentially the same, except that the ‘feature ring’ is replaced by temporal sequences which are stored in self-organising maps of the proposed network. Visual scanning via saccades is then seen to be similar to the more *symbolic* processing of, say, characters or words. Each fixation point will provide a vector to be learned (via the visual filters), just as each sound (phoneme) provides a vector to the auditory system. These vectors are then passed to the temporal learning system in a consistent sequence, in much the same way as with language.

If the theory is correct, then we should expect that the saccade trace laid down during learning of a new object should be reproduced when that same object is being recognised,<sup>12</sup> and this expectation was confirmed by Noton and Stark [231]. They found that in about 65% of cases, subjects reproduced the same learned scanpath when subsequently recognising an object.<sup>13</sup>

The mere existence of such regular scanpaths suggests a process of learning. The finding that different subjects had different scanpaths for a given picture, suggests that the scanpaths are not the result of peripheral feature detectors that control eye movements independent of the recognition process. Any such detectors might be expected to operate similarly for all subjects. Conversely, the observation that a given subject had different scanpaths for different pictures suggests that the scanpaths do not result from some fixed habit of eye movements.

As well, the fixed scanpaths suggest that the eye-movement motor components involved in perception are not independent movements to move a pattern over the retina (in order to fill in the details left out by a limited fovea), but are rather an integral part of the memory sequences on which recognition is based.

Figure 8.12 (a) shows a typical scanpath found by Noton and Stark [231] in which subjects viewed an adaption of a line drawing by Paul Klee. Figure 8.12 (b) shows an idealised scanpath for this particular subject for this particular picture.

Groner, Walder and Groner [128] explored the idea of ‘local scanpaths’ in the sense of Stark et al. (that is, in reflecting consistent patterns of successive fixations), and ‘global scanpaths’ (reflecting the distribution of fixations over a larger time scale irrespective of their immediate succession). In global scanpaths, fixations don’t follow each other, but rather reflect a tendency to concentrate somewhere in the course of the exploration process, representing more of a ‘search’ process to satisfy an expectation of the subject. They found the possibility of discriminating individual styles in face scanning by considering only the relative frequency of fixation triplets. Walker-Smith, Gale and Findlay [351], in studying face recognition, also

---

<sup>12</sup>Assuming a similar context inputs and a similar purpose for the recognition.

<sup>13</sup>The 65% required a strictly serial ordering (their feature ring). In our proposal of learned temporal sequences we are not restricted to the linear sequences of a ring.

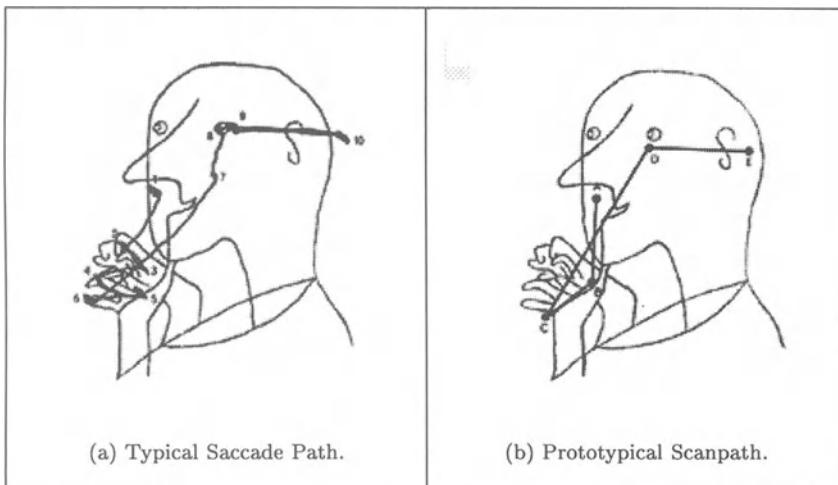


Figure 8.12: Saccade Scanpaths.

observed short sequences (two or three saccades) that were common to both recognition and examination phases.

Groner and Menz [127] found that while most subjects utilised both local and global strategies, there were some subjects who used just one or the other strategy, and even some subjects who used neither strategy.

In their original articles, Noton and Stark claimed that in 65% of cases the same sequence occurred in both memorisation and recognition of a pattern. However, they gave no quantitative criterion for recognising a scanpath. In another study examining the role of scanpaths in the recognition of random shapes, Locher and Nodine [185] applied a more precise criteria. They found the presence of scanpaths in over half of all eye-movements regardless of the shape complexity.

To provide further quantitative measurements of scanpaths, Hacisalihzade, Stark and Allen [130] have modelled sequences of visual fixations as Markov processes, with the sequences abstracted as character strings.<sup>14</sup>

Stark and Choi [324] also employ a string editing technique in which regions of (2D) space are quantised and allocated a character label. All fixations falling within each area are recorded as having the label of that area. Statistical techniques may then be used to compare strings of learned and recognition sequences. The technique is very useful in defining similarity and dissimilarity between eye movement patterns. The results are impressive and confirm the presence of regular, learned scanpath sequences in eye movements, and a high degree of similarity in eye movements between the

<sup>14</sup>Note that this is essentially the approach taken using the ABC model, with self-organising and recurrent nets used rather than Markov processes, and with vectors obtained from local filters instead of string characters.

initial viewing period and subsequent recognition.

One potential objection to scanpath theory is the well-known phenomenon of instantaneous recognition. Simple visual scenes and familiar faces can usually be recognised in one glance. Further, information about facial expressions can be identified from a 20 msec view as reliably as from much longer exposures [347].

## 8.8 The ABC Temporal Paradigm

There have been a number of previous studies on the learning of temporal sequences using neural networks, many in connection with speech recognition and language. Various approaches have been used to incorporate the temporal component, both explicitly and implicitly. One approach is to incorporate time explicitly by a transformation into a spatial dimension [162, 163, 174, 350].

Recurrent neural connections were introduced by Jordan [157–159] who was concerned with the parallelism of speech production, and the relationship between these parallel properties and the overarching sequential nature of speech. Elman [95, 96] conducted a series of important experiments in temporal learning. The network used by Elman consisted of a recurrent feed-forward network, with the hidden layer units copied back to the input layer on a one-to-one basis. The input layer thus consisted of two components; the external inputs containing the elements of the sequence, and the recursive copies of the hidden units as a contextual input.

### 8.8.1 ABC Temporal Learning Model

The temporal learning model we propose brings together two important concepts in neural networks; recursion and self-organising maps. Although some have tentatively combined these two elements before (e.g., Scholtes [304]), their combination has not been given sufficient attention in our opinion, and the research has tended to concentrate on simple feed-forward recursive networks (SRN) using the back-propagation update rule.

To learn a temporal sequence, we first consider the arrangement as shown in Figure 8.13.<sup>15</sup> Here two self-organising maps (we use the Kohonen mapping algorithm in simulations) are combined together, with the second having a recurrent connection back to itself. This arrangement is similar in principle to that used by Elman, but uses self-organised maps and Hebbian learning rather than a feed-forward net with backpropagation. This arrangement is first described in some detail.

The first component (a) is a standard self-organising mapping (SOM) from an input vector onto the SOM surface. The mapping is indicated by

---

<sup>15</sup>This network is referred to as SOMA (Self-Organised Motor Action).

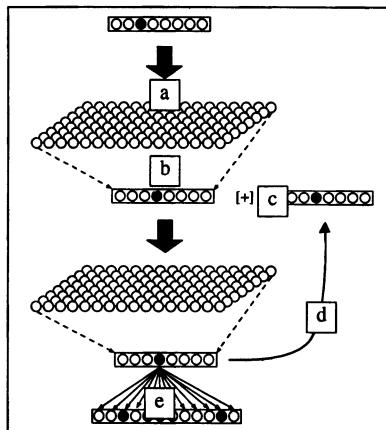


Figure 8.13: Temporal Learning Components.

the solid arrow. Rather than being considered as separate entities, the input vectors are to be regarded as a continuous series of vectors  $v_t, t = 0, \dots, n$ . The aim of the learning algorithm is to find some temporal relationship between the  $v_t$  vectors such that  $v_t = v_t(v_{t-1}, v_{t-2}, v_{t-3}, \dots, v_{t-p})$  where  $p \leq n$ .

The output from the initial SOM is ‘vectorised’ as indicated in Figure 8.13 (b). The process of ‘vectorisation’ is merely a method of obtaining a vector from an array by some process of linearising the matrix. The rows (or columns) of the array are simply stacked end-on-end to produce a vector. The order of the vectorisation process is not relevant as the algorithm is order independent.

The vector that is mapped to the second SOM surface is made by forming a vector sum (concatenation) of two other vectors—the ‘vectorised’ output from the first map, and the ‘vectorised’ output from the second map on the previous epoch (c). That is, if we assume time-locking of steps in the process, and the input to the first SOM is given by  $t_q$ , then the relevant output from the first SOM surface will have been formed at  $t_{q-1}$ , and the recurrent output vector from the second map will have been formed at  $t_{q-2}$ . The new total vector forms the input to the next stage—mapping to the second SOM surface.

The ‘vectorised’ output of the second SOM surface is recursively linked back to its inputs via the recurrent link shown in Figure 8.13 (d).

The final component of this initial temporal configuration is a weighted, fully-connected layer that is updated via Hebbian learning (e). This component learns each output vector that is to be associated with the corresponding input vectors.

The network as shown in Figure 8.13 is useful diagrammatically in that it makes it easier to understand the overall processes. In fact, the vectors as

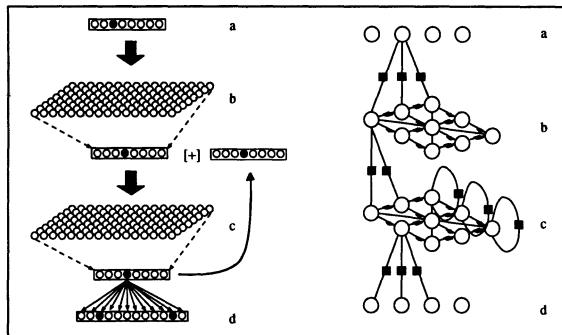


Figure 8.14: Actual Neuron Layout and Connections.

shown in Figure 8.13 are not really necessary. For example, in the biological version of the system, the outputs of the first SOM connect directly to those of the second, without the intermediate vector of neurons as shown in Figure 8.13. The actual connections are shown in Figure 8.14 alongside the corresponding diagrammatic form.

The ABC system is based on ‘matrix’ or ‘array’ transformations—essentially transforming one SOM surface into the next. However, it is sometimes easier to conceptualise the model if, at times, we think of these arrays as vectors. This enables us to better fit the ideas expressed about the ABC model into the current literature on artificial neural networks (ANN).

The weights are modified via Hebbian learning to reproduce the required output vectors. The Hebbian update of the motor weights is illustrated in Figure 8.15. This figure shows the actual output vector connected directly to the SOM surface and not via an intermediate vector as in Figure 8.13 (e). The connections from the training vector to the output vector are not weighted.

The algorithm uses an extension to standard Hebbian learning in that, as well as the weights being increased for temporally synchronised pre- and post-synaptic activity, the weights are also *decreased* when the pre- and post-synaptic neurons are not temporally correlated [12].

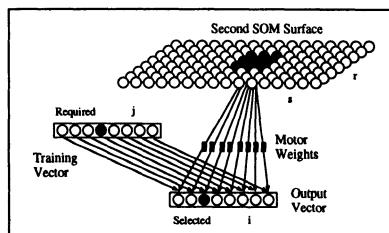


Figure 8.15: Hebbian Update of Motor Control Weights.

Further details on the temporal learning model are to be found in [42].

### 8.8.2 Temporal Learning Experiments

We have conducted a number of simulation experiments in order to duplicate some of the temporal learning results in the literature, and to determine the learning abilities of the network. These results are not covered in any detail here (see [42]). However, we briefly mention some of the results obtained in this section. In most of these experiments the system is trained to predict the next value in the sequence.

#### Elman's Consonant/Vowel Sequences

In this case, the input is in the form of a random sequence of characters (**b**, **d**, **g**), with each of these consonants being followed by a particular vowel. Every **b** is followed by exactly one **a** character; every **d** is followed by two **i** characters, and each **g** is followed by three **u** characters. An input string in this scheme might be, for example, **diibaguuubadiidiiguuu** ... [95].

The consonants, being randomly distributed, are obviously not able to be predicted, but whereas Elman's system could only give a statistical indication of the learning of the sequence, we were able to obtain 100% accuracy in the vowels at the end of the training epochs.

#### Bidirectional Link

The task here is to learn the sequences generated by the finite-state transition network (FSTN) shown in Figure 8.16. The importance of this network is that it can generate a variable number of embedded **C** characters. Various studies attempting to learn this (or similar) sequences with SRNs have shown that the SRN is unable to generalise the loop between states **2** and **4**, but rather learns specific path information [69, 313].

Sharkey and Sharkey [313] found that an SRN could not learn the FSTN of Figure 8.16 at all. They concluded that this was a problem that could not be avoided with SRNs when trying to learn a bidirectional link. If the system does not generalise but records specific paths only, then no matter how many embedded **C**s the network is trained on, the final **C** must predict a **B** only. Thus a test using more embedded **C**s than the training set will fail.

Sharkey and Sharkey could only get a SRN to learn the FSA by explicit restriction and training of the hidden units. In this way, they showed that the network architecture was capable of learning the required generalised transitions, but the backprop algorithm could not do so.

The SOMA network was able to learn the FSTN as shown in Figure 8.17. This description was obtained by examining the allowed transitions from the output of the network and the winning nodes of each SOM surface. The one difference between this network and the original is that the network was unable to determine if an even or odd number of **C**s was

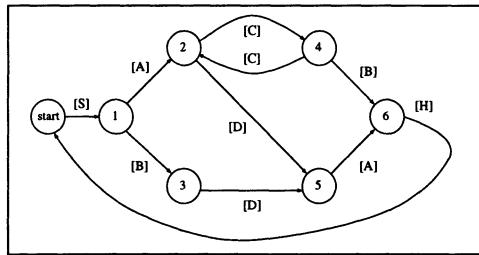


Figure 8.16: Bidirectional Link.

significant. Examination of the winning nodes did show however that the network had been able to generalise as opposed to the SRN network.

In order to learn the original FSTN it is necessary that the two CC transitions are separated, and that the second match up with and generalise with the SA winning node. One method of ensuring that this could occur has been explored. If each node has some form of *refractory period*, that is, a period following firing during which it is unable to fire again, then the successive Cs would be forced to take alternate winning nodes. This was in fact tried and did succeed in splitting the two Cs into separate nodes.

### Counting & Memory

Given that the SOMA network is capable of retaining information about past inputs, and using this information to learn temporal sequences, it is appropriate to ask how far back this ‘memory’ extends. In order to test this, the SOMA network was trained on the continuous sequence

7 a a a a a a 7 a a a a a a ...

Without the refractory period (RP) extension, the network was only able to learn up to 6 character. However with the RP extension installed, the network was able to learn the above sequence to 100% accuracy. It would correctly predict a 7 after 6 a characters, then another 6 a characters

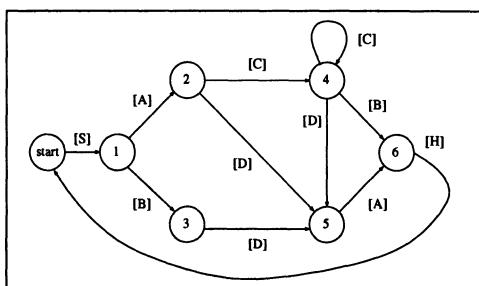


Figure 8.17: Actual FSA Learned By System—Extended Learning.

in sequence, and so on. Despite considerable effort, we were not able to get the system to ‘count’ to eight.

These and other temporal experiments [42] show that the ABC temporal model is a very powerful temporal learning mechanism. The generalisation achieved on the SOM surfaces means that the temporal learning is able to generalise much more than for SRNs to the effect that much more ‘symbolic’-like behaviour is found.

### 8.8.3 Temporal Experiments and Vision

The model of early vision proposed here is similar to that proposed in the previous section for one-dimensional learning. For vision, however, the domain is two-dimensional—the system is required to learn not only the input sequences, but also a 2D scan pattern to follow in order to ‘parse’ and hence ‘recognise’ an object.

The early vision system builds retinotopically indexed maps of environmental features such as spatial frequencies, motion and colour. Temporal sequences of these feature vectors are learned and provide the means of recognising an object.

The proposed vision system, then, is a continuation of the temporal models described previously. The eyes saccade over the objects in an image of the world, touring around its coordinates in some manner. At each fixation point, various ‘filters’ extract information which corresponds to the neural activity associated with the ‘view’ at that instant in time. The retinal information is transformed via receptive fields, and separated into various sub-modalities (spatial frequencies, colour and motion), until several distinct vector representations are available at areas V1 and V2. The learning of the sequences of vectors occurs separately in areas V3, V4, and V5 (and then subsequent areas). The temporal sequence of these restricted ‘part-image’ vectors is learned by the system and associated with the appropriate world object.

In learning (and reproducing) temporal sequences of vectors, the proposed system is consistent with, and provides a mechanism for the scanpath theory of Stark and Noton [231]. Thus, rather than the term scanpath we will use scan-sequence.

While a major portion of the vector corresponds to the components of the scene at the fovea, this is not exclusively the case. The larger receptive fields of the periphery are also included.

Recognition is not separate from the learning process. As the eye saccades over an object, the incoming sequence of vectors will be ‘recognised’ and some behaviour/action initiated. The behaviour could involve the production (speech) of a verbal ‘label’ that has been previously associated with the object—a behavioural demonstration of recognition.

Consider an experiment which attempts to recognise some simple reg-

ular geometrical shapes—an equilateral triangle, a square and a circle. To keep the exercise small, we will use only three colours for each object—red, green and blue, with a white background, and only one standard size for each shape.

We first need to consider how the external world will be converted into an appropriate vector for learning by the model. Consider a gross simplification of the Wilson Modified Line-Element Model [362] that is used in the full ABC simulation model and discussed in some detail in [42].

If a mythical creature inhabited a world of these geometrical figures, and some shapes were good to eat while others had to be avoided, then it is possible that the creature might have evolved a series of spatial frequency filters similar to those shown in Figure 8.18.

At the centre of the creatures retina, the light sensitive cones are interconnected so that it can detect various lines and colours in linear ‘filters’ which subtend certain angles to the horizon as indicated in Figure 8.18. The creature does not have a  $30^\circ$  filter, or a  $150^\circ$  filter as they are not required in *this* world. Whenever the creature looks at an object, the light impinging on these filters sends a signal to the creature’s visual system to enable it to recognise the shape. Each filter provides some data to be included in an input vector that will form the input to a SOMA recognition system. Note that these fictitious spatial filters are double sided; that is, each side of each filter supplies one vector element.

The order in which the vector elements are allocated is indicated by the numbers on the combined filter in Figure 8.18. That is, the left half of the horizontal filter supplies a value for the first element of the vector, the right horizontal half a value for the second element, and so on.

Suppose that the creature evolved two filters of the same general type. The first provides a measure of the existence or not of a line boundary

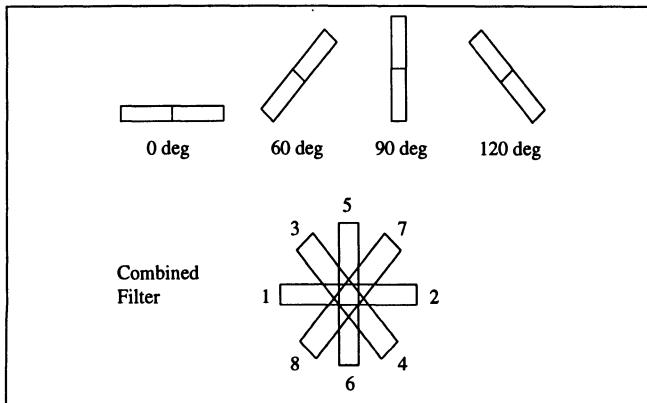


Figure 8.18: Simple Spatial Filters.

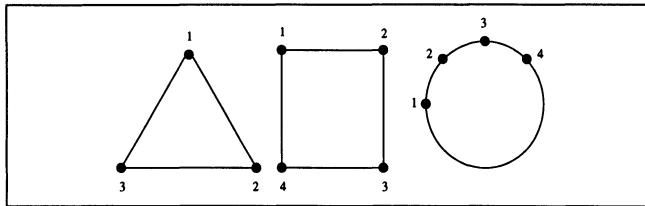


Figure 8.19: Saccade/Fixation Points.

within its domain. If a line boundary falls on a component of a filter then the corresponding vector element is 1, otherwise it is 0. These line spatial frequency filters thus provide an eight ‘bit’ vector.

The second filter is sensitive to colour. For this colour spatial filter, if an area is mostly filled with a particular colour, then the corresponding elements are given the appropriate colour code according to the following table. This filter thus produces a sixteen ‘bit’ vector.

white	00
red	01
green	10
blue	11

We restrict ourselves to considering only a limited set of fixation points for each object, as shown in Figure 8.19. For these fixation points, the values given to the respective vector elements is shown in Table 8.1 for the line spatial frequency filter, and Table 8.2 for the colour spatial filter.

We restrict ourselves further by only saccading in a clockwise direction. This is not a restriction of the system—the system could learn the fixation

Figure	Saccade Point	Vector
triangle	1	0 0 0 1 0 0 0 1
	2	1 0 1 0 0 0 0 0
	3	0 1 0 0 0 0 1 0
square	1	0 1 0 0 0 1 0 0
	2	1 0 0 0 0 1 0 0
	3	1 0 0 0 1 0 0 0
	4	0 1 0 0 1 0 0 0
circle	1	0 0 0 0 1 0 0 0
	2	0 0 0 0 0 0 1 1
	3	1 1 0 0 0 0 0 0
	4	0 0 1 1 0 0 0 0

Table 8.1: Vector Values for Line Filter

Figure	Saccade Point	Vector
red triangle	1	0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1
	2	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
	3	0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
green triangle	1	0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0
	2	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
	3	0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
blue triangle	1	0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1
	2	1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
	3	0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0
red square	1	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0
	2	0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1
	3	0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0
	4	0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0
green square	1	0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
	2	1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
	3	1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
	4	0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0
blue square	1	0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0
	2	1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1
	3	1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0
	4	0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0
red circle	1	0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 0
	2	0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1
	3	0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1
	4	0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1
green circle	1	0 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0
	2	0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0
	3	1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
	4	1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0
blue circle	1	0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0
	2	0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1
	3	1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1
	4	1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1

Table 8.2: Vector Values for Colour Filter

points in any order—but it does reduce the time and complexity of the exercise. We thus train the system on clockwise relational saccades (1-2-3-4-1-...) for each object.<sup>16</sup>

The output from the system is a set of concepts which will indicate the shape and colour of the recognised object. For example, the data supplied to the system consists of a concept and the temporal sequence that, when recognised, would indicate that this concept has been seen. The concept *triangle* includes triangles of all colours, whereas the concept *red* includes all red shapes. The data supplied to the temporal learning system to define the various concepts used in the experiment are:<sup>17</sup>

<sup>16</sup>One could question if the use of particular fixation points is appropriate, and how does this relate to saccades and fixation in the human visual system. This question is discussed in Section 8.7 in relation to the learning of scanpaths.

<sup>17</sup>We have not yet described where the concepts, such as *red* or *triangle*, come from.

```

triangle Tr@1 ReTr@1 Tr@2 ReTr@2 Tr@3 ReTr@3
triangle Tr@1 GrTr@1 Tr@2 GrTr@2 Tr@3 GrTr@3
triangle Tr@1 BlTr@1 Tr@2 BlTr@2 Tr@3 BlTr@3
...
red Tr@1 ReTr@1 Tr@2 ReTr@2 Tr@3 ReTr@3
red Sq@1 ReSq@1 Sq@2 ReSq@2 Sq@3 ReSq@3 Sq@4 ReSq@4
red Ci@1 ReCi@1 Ci@2 ReCi@2 Ci@3 ReCi@3 Ci@4 ReCi@4
...

```

The string  $\text{Tr}@1$  is a shorthand way of indicating the appropriate vector for the line filter at saccade point 1 of a triangle,  $\text{ReTr}@1$  is shorthand for the actual colour filter vector that would be generated were the creature to fixate on point 1 of a red triangle, and so on.

The input vectors for the temporal sequences are supplied in pairs, the first for the line filter vector, the second for the colour filter vector. Thus the first line of the input data shown above is a trace of the three saccade points for a red triangle, the second a trace of a green triangle, and so on.

To ensure that the recognition can begin at any saccade point for each object, the input data values are wrapped-around in a continuous stream; for example, for the triangle, the sequence 1-2-3-1-2-3... is used in training. Training thus associates the continuous ‘visual’ streams produced by the line and colour vectors with the appropriate concepts.

The system structure for this example is that shown in Figure 8.20. The two input vectors produced by the line and colour filters are concatenated for mapping to the first SOM surface, and the output vector is trained to select one or more concepts as indicated. The concepts are shapes *circle*, *square* and *triangle*, as well as colours *red*, *green* and *blue*. If a *blue square* is recognised, both the *blue* and *square* concept nodes should be selected.

Once the system is trained, we are in a position to see if it is able to recognise objects within this micro-world. Consider the two objects shown in Figure 8.21 (a)—a red triangle and a blue square.

The input used to test the system is that which would have been produced by the creatures early visual system in saccading around the objects say from points 1, 2 and 3 on the square to points 4, 5 and 6 on the triangle, all in a clockwise direction to be consistent with our training data. The temporal sequence pairs

```
Sq@1 BlSq@1 Sq@2 BlSq@2 Sq@3 BlSq@3 Tr@2 ReTr@2 Tr@3 ReTr@3 Tr@1 ReTr@1
```

are supplied as input and the output concept vector examined to see which concept has been selected.

---

In this temporal learning experiment, the concept data (including labels) is supplied, and thus the ABC model appears to be using supervised learning. However, within the full ABC cognitive model all elements are learned, including the concept labels. The ‘labels’ are attractors formed by similar learning in the *auditory* modality, which are subsequently associated with the corresponding visual concepts.

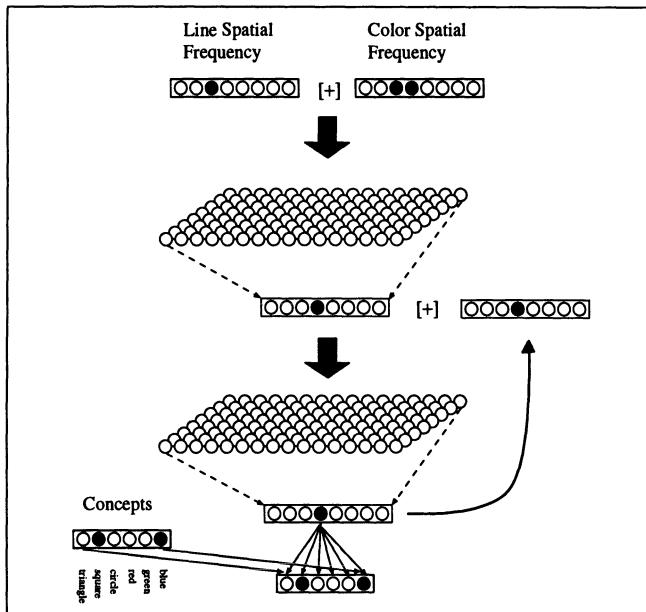


Figure 8.20: Vision Network.

As an example of a more difficult recognition case, consider the ‘scene’ as shown in Figure 8.21 (b). Here a green triangle and a red square partially occlude a blue circle. In this case the vectors produced by the filters at points 2 and 5 will not be as per the training examples but will be ‘mixed’. Figure 8.22 shows the placement of the filters when the creature has fixated on points 2 and 5 respectively. The input filter vectors for these points are:

Point 2	Line SF Colour SF	0 0 0 1 0 0 1 1 w b w g w g b g 00 11 00 10 00 10 11 10
Point 5	Line SF Colour SF	0 1 1 0 0 1 0 0 b r b r w r w b 11 01 11 01 00 01 00 11

When the trained system is tested on the temporal sequence produced by fixating on points 1 to 9 as indicated in Figure 8.21 (b) the output concept vector scores were observed as shown in Table 8.3.

It is clear from this rather simple example that the system *has* been able to recognise the appropriate objects. The output vectors at each recognition stage could be linked to some form of action depending on the recognition; if green triangles are good to eat, a (learned) approach action would be initiated, but if blue circles were dangerous then an avoid action could be undertaken when the blue circle is detected. Of course, this example is

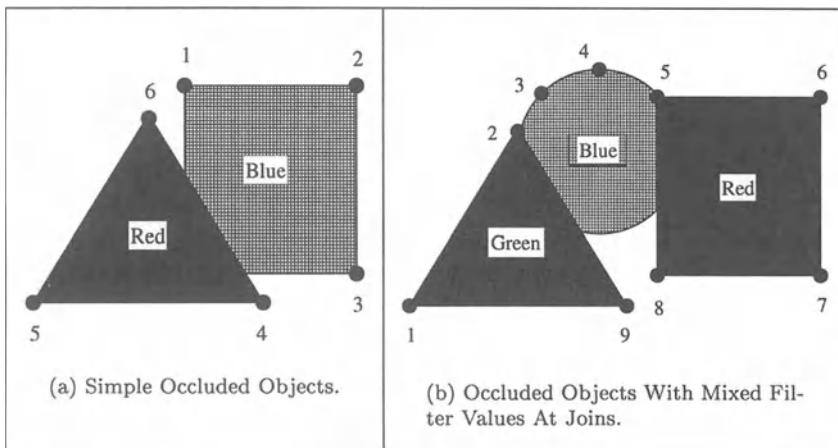


Figure 8.21: Occluded Objects.

rather simplistic, but the general principles are the same as for the full model.

Figure 8.23 illustrates a possible network that would both learn the temporal sequence of ‘features’ and the sequence of motor control vectors to provide anticipatory *next* saccade points. Let us assume that the eye is currently fixated at a point  $P_1(t)$  on a new object that has just been presented to the subject. During the initial learning phase, visual inputs from  $P_1(t)$  enter the system (at time  $t$ ) and progress through the SOM surfaces to be available for association with some eye motor vector. The passage through the system takes a time  $\Delta t$ . During this period, the eye make a further saccade to a point  $P_2(t + \Delta t)$ , and it is the eye muscle vector for this second fixation point that is learned and associated with the initial inputs at time  $t$ .

During a subsequent recognition phase, visual inputs from fixation point  $P_1$  can then generate the appropriate eye muscle vector required to saccade

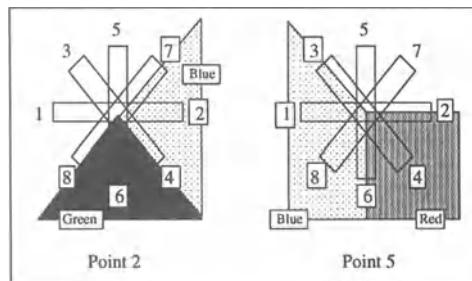


Figure 8.22: Occlusion Filters.

Point No.	Triangle Score	Square Score	Circle Score	Red Score	Green Score	Blue Score	Concept
1	0.62	0.00	0.00	0.00	0.59	0.00	green triangle
2	0.00	0.00	0.77	0.00	0.00	0.83	blue circle
3	0.00	0.00	0.89	0.00	0.00	0.96	blue circle
4	0.00	0.00	0.92	0.00	0.00	0.99	blue circle
5	0.00	0.00	0.92	0.00	0.00	0.99	blue circle
6	0.00	0.95	0.00	0.98	0.00	0.00	red square
7	0.00	0.95	0.00	0.98	0.00	0.00	red square
8	0.00	0.95	0.00	0.98	0.00	0.00	red square
9	0.95	0.00	0.00	0.78	0.86	0.00	green/red triangle
1	0.99	0.00	0.00	0.00	0.95	0.00	green triangle
2	0.99	0.00	0.00	0.00	0.95	0.00	green triangle

Table 8.3: Recognised Concept Scores.

to fixation point  $P_2$ .

## 8.9 ABC Vision Model

Before giving a brief description of the full ABC model of cognition, it is appropriate to make a disclaimer—it is not claimed that the model presented in this chapter completely describes the structure and working of

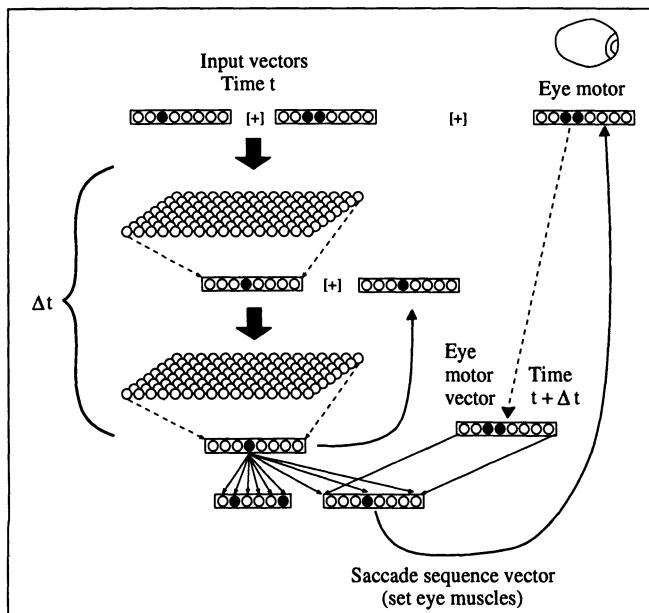


Figure 8.23: Vision Network With Learned Saccades.

the brain. The brain of humans and the higher vertebrates has evolved over many millions of years. Various components have been laid down at different periods of evolution, and it may even be valid to view the brain as several interacting “sub-brains” rather than as a single unit. The brain of vertebrates consists of three basic divisions; the cerebral hemispheres, the brain stem and the cerebellum. We consider only the most recent addition phylogenetically—the cerebral cortex. The cerebral cortex in most pronounced in higher vertebrates, and from the study of aphasias and other deficits has been shown to be the major area for cognitive function [161].

The sheer number of neurons and connections in the brain is bewildering, and the significance of the projections between the various cortical and extracortical areas is not well understood. In this chapter, an attempt is made to examine general principles that may be used in forming an overall reverse-engineering “design” for the brain, and by so doing, to try to form an understanding of how humans and animals are able to conceptualise, to learn, and in the case of humans in particular, to use language—given the neurobiological data that we currently have available.

A number of important features of the brain of higher animals need to be incorporated into any model:

- the ubiquity of topographical maps in the cortex and other brain areas,
- the prevalence and systematic character of feedback loops signifying a dynamic, time-dependent system,
- the cross-linking of the various modality sensory projections in the association areas.

These, and other known properties of the brain, are incorporated into the ABC model. There is strong neuroanatomical and neurophysiological support for the model—see [42] for details.

### World Filters

The external world provides various forms of energy that may be sampled by an organism. Specialised neural structures (sensory receptors) transform these natural stimuli that impinge upon our bodies into neural signals. In addition, the brain receives inputs from other internal sources such as from the movement and tensions of skeletal muscles, and from the viscera.<sup>18</sup>

These different sensory systems may be subdivided into three categories: the *exteroceptive* systems which are sensitive to external stimuli, the *proprioceptive* systems which provide information about the relative

---

<sup>18</sup>Internal organs such as the heart, stomach, lungs, tear ducts, glands, blood vessels and so on.

positions of body segments and the position of the body in space, and the *interoceptive* systems which provide internal bodily measures such as blood pressure and blood glucose levels. We are usually not conscious of the interoceptive signals, whereas we appear to be generally aware of the exteroceptive and proprioceptive stimuli.

In the following discussion we concern ourselves mainly with the exteroceptive systems, although there is no reason to doubt that the same principles would apply to the other systems as well. The exteroceptive systems of concern are the visual system, the auditory system and the somatosensory system. Each of these modalities has several sub-modalities as outlined in Table 8.4.

Each sub-modality ‘filter’, via its sensory receptors, provides a number of neural responses *at each instant of time*. We examined this point briefly in the discussion of temporal learning in the visual system (Section 8.8.3). It is not appropriate to examine these sensory systems in detail at this stage. More is discussed about the actual implementation of the full system in [42]. All we need to consider at this stage is that the sensory systems, both external and internal, initially provide a ‘vector’—that is, a reading  $v_s^m(t)$ , of the sensory receptors at time  $t$ , where  $m$  is some modality, and  $s$  one of its sub-modalities.

### **Self-Organising Maps**

The sub-modality vectors are then mapped to self-organising layers of the primary and secondary sensory areas. The primary and secondary sensory areas in primates are well known to contain topologically organised mappings of the incoming sense data. In the case of the visual system, these are areas V1 and V2, with corresponding areas A1 and A2 for primary and secondary auditory areas, and S1 and S2 the somatosensory primary and secondary areas. The process of topological mapping to various sensory areas is consistent with a mapping to the initial SOM.

### **Recurrent Loops**

As discussed previously, the cortex of primates is richly connected by back-projections to previous layers, consistent with the recurrent loop structure of the ABC model.

The next stage of the model is to combine the multiple sources of information that are available within each modality.

### **Intra-Modal & Inter-Modal Associative Layers**

Association layers surround the primary sensory regions [117, 243]. These are of two types; *intra-modal* association areas which link various regions within each modality, and *inter-modal* association areas which link downstream areas of separate modalities (for example, visual with auditory, visual with somatosensory, and so on).

The ABC model used in simulations assumes that an appropriate filter vector has been obtained. In many of the temporal simulations, for exam-

Modality	Sub-modalities
Vision	Spatial Frequencies & Angles Colour Motion
Auditory	Frequency (Tone) Bandwidth (Spectral profile) Symmetry (Prosody) Intensity
Somatosensory	Touch-pressure Position sense—static limb position & limb movement (kinesthesia) Thermal sensations—separate for both hot and cold Pain sensation

Table 8.4: Exteroceptive Modalities and Sub-modalities.

ple, the vector value assigned to the inputs was assigned at random. In the full simulation of the model [42], each fixation provides appropriate vectors for the visual modality component from:

- a spatial-frequency and angle filter (based on the Wilson Modified Line-element theory [362]),
- a motion filter (based on a Reichardt motion detector),
- a simple RGB colour filter.

Each other modality provides appropriate vectors based on the recording of the external world made by the particular filters. At each time instant (each fixation), every vector is mapped to its own particular SOM. That is, the spatial-frequency and angles vector is mapped to a SOM, the motion vector mapped to another SOM, and the colour vector mapped to a third SOM.

Each of these maps will self-organise to the actual distribution of mapped vectors, thus providing a representation of the external world, and with an appropriate allocation of the surface area being given to the frequent incoming vectors within the creature's experience.

The next stage is to provide associative links between all of the maps within one modality. This is achieved in the ABC model in a manner shown in Figure 8.24. A second layer of neurons, termed the *association layer*, accepts direct inputs from the corresponding SOM layer for each sub-modality, as well as inputs from other SOM surfaces within the modality. The link between each SOM layer and its corresponding association layer is direct (i.e., unweighted), and each neuron in the SOM layer projects only to the corresponding neuron in the association layer. The linkage is 1 : 1, and we call this a *direct weight*.

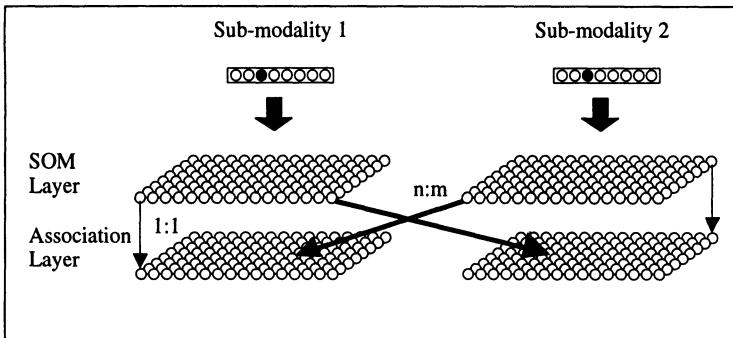


Figure 8.24: ABC Associative Layer.

In contrast, the association link from other SOM surfaces to a particular association layer may be reduced to some fraction of the original. Each neuron in a SOM layer has a weighted link to *every* other neuron in other association layers—the linkage is thus  $n : m$ . We call these *association weights*.

The association weights are learned by Hebbian learning. If both the pre-synaptic neuron (on a SOM layer) and the post-synaptic neuron (on an association layer, excited by the direct weight and possibly other association weights) are firing, then the weight is increased. The weight is decreased if there is no correlation between the pre- and post-synaptic neurons.

The linked connection between sub-modalities allows for associations to be formed between vector inputs—they can mutually support each other. An advantage of this linkage scheme is that it is non-metric. It does not suffer the problem of feature-space representations in attempting to directly link metric features.

The input vector to the next stage is the output of the association layer, rather than the vectors from the external world filters. This is shown in Figure 8.25.

We should also note at this point the equivalence of the diagrammatic representations shown in Figure 8.26. In our discussion of temporal learning we used the diagrammatic form shown in Figure 8.26 *A*. This form was used to emphasise that the motor weights between vectors 2 and 3 were to be updated via Hebbian learning in order to learn to reproduce vector 1. The connections between vectors 2 and 3 are weighted, whereas the connections between vectors 1 and 3 are direct—a weight of unity. Each node in vector 2 is connected to *all* nodes of vector 3, whereas the connection between nodes of vectors 1 and 3 is only to the single positionally-matched node—that is, only one connection per node.

Rearranging the vectors as in Figure 8.26 *B*, and noting that the distinction between vectors and arrays is both somewhat artificial and was

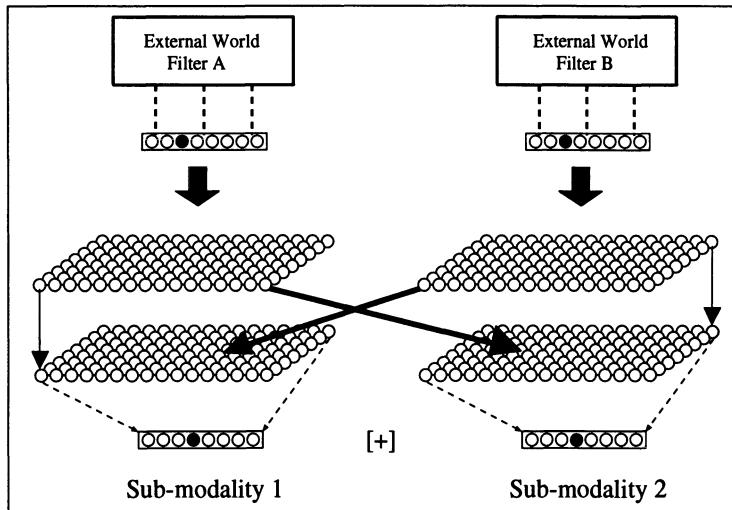


Figure 8.25: Associative Layer And External World Filters.

only introduced for convenience of understanding, we can see that the diagrammatic form of Figure 8.26 *A* is indeed equivalent to that of Figure 8.26 *C*.

We are now in a position to see how we might combine these components to get an overall structure for the cortex. We do not claim that this is the final picture, but rather that this view:

- is biologically and neuroanatomically consistent and reasonable,
- has properties more closely associated with the actual brain than either the computational cognitivist approach, or the simplistic FFNN,
- is internally consistent and self-similar, and is consistent with the observed neuroanatomical structures of the cortex,
- allows for the many attractive properties associated with the connectionist approach, such as generalisation, soft-degradation, learning and association, and so on,
- provides for massive parallelism in the processing between layers, yet is serial in its use of temporal sequences,
- is consistent with the view taken in this chapter that the processes of the brain are to learn associated and temporally connected sequences, rather than ‘facts’ or ‘representations’, and that the learned behaviours resulting from the associated temporal sequences are the means of cognition, rather than computational processes on representations.

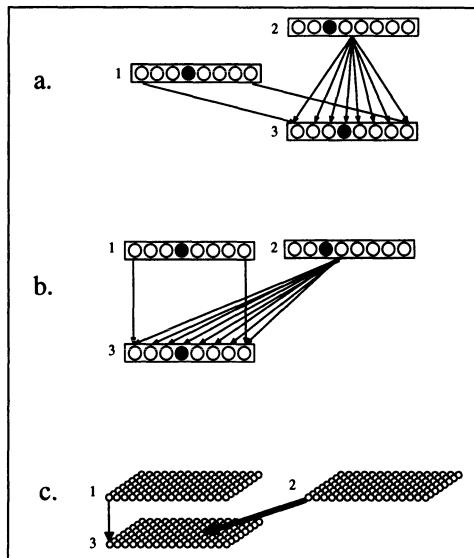


Figure 8.26: Equivalence Of Diagrammatic Forms.

Figure 8.27 shows one possible model for the visual system. Other variations on this model are discussed in [42]. While initially appearing to be complicated, the figure can be explained easily.

- external world filters e.g., spatial frequency and orientation, colour, motion—only two filters are shown here but there may be more,
- vector produced by world filter,
- SOM mapping of sub-modality vector,
- association layers between sub-modalities,
- vectorisation of association layer output,
- concatenisation of sub-modality output vectors for mapping to vision modality SOM,
- vision modality SOM,
- vectorisation of vision modality SOM,
- vision transitions recursive loop SOM,
- recursive temporal loop,
- vision ‘motor’ vector,
- ‘copy’ of input vectors to record temporal differences,
- time displaced ‘copy’ of input vectors,
- association layer to join other modalities,
- association vector to join other modalities,
- weighted associations from other modalities—auditory, somatosensory, proprioceptive,
- sequence generation loop—“imaging”

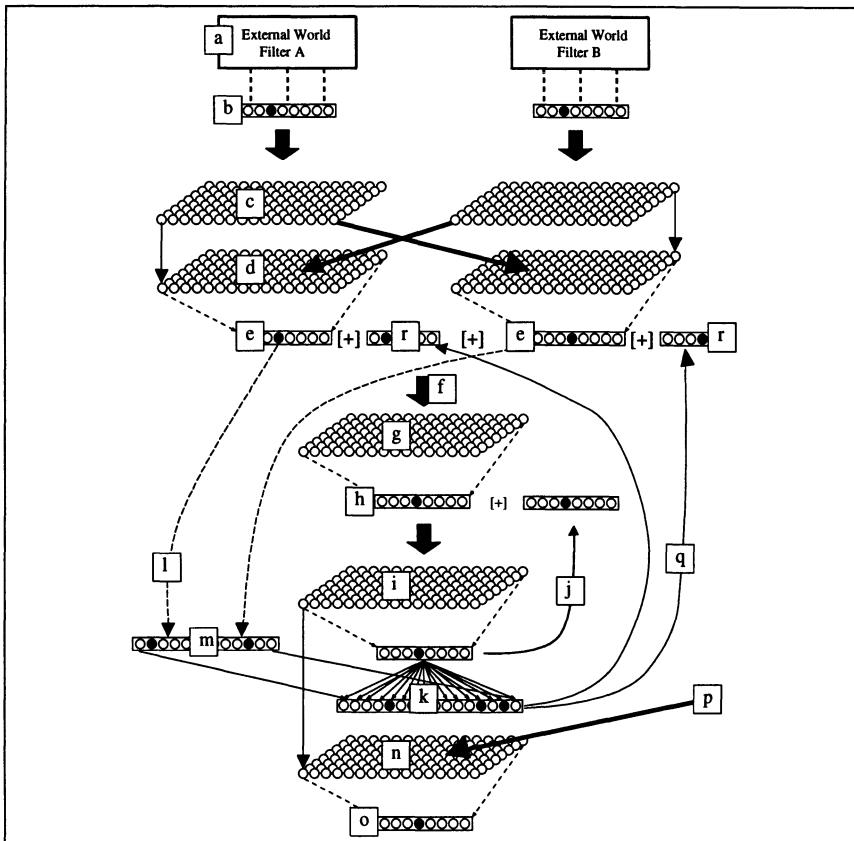


Figure 8.27: ABC Vision Model.

In developing the temporal learning demonstration of the vision model in a previous section, it was not made clear just where the ‘concept’ labels *red*, *triangle* and so on originated. This inability to describe the origin of labels, or to indicate a means whereby new labels may be generated, is a serious problem for other cognitive models. The ABC model does have a mechanism for the origin of labels—they are simply attractors (or combination of attractors and temporal sequences) formed in the auditory modality (in the case of spoken labels), or other visual attractors (or combination of attractors and temporal sequences) in the case of written labels.

If we compare Figure 8.27 with Figure 8.20, this can be readily seen as shown in Figure 8.28. The sound attractors of the spoken label (sign) are associated with the visual attractors of the object (or event or concept). Conceptual ‘cues’ are obtained from association layer links to other modalities, especially the auditory modality which provides the sound ‘naming’ cues.

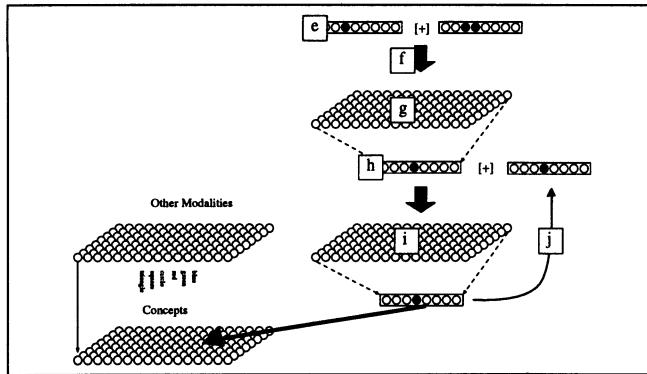


Figure 8.28: Vision System Compared With Vision Sequence-Learning Experiment.

A full description of the ABC model may be found in [42], along with details of a computer simulation.

## 8.10 ABC Model Avoids Problems Of Traditional Models

A major characteristic of the ABC model is that it positions cognition as a dynamical system rather than computational, thus eliminating the multitude of issues associated with the symbol processing model and representations.

ABC is a model of the full task of vision, from early receptors to perceptual conceptualisation, labelling and behaviour. It is a consistent, self-similar architecture which is supported by the neuroanatomical evidence. The model is consistent with the scanpath observations of Noton and Stark, and regards visual recognition as a dynamic process of temporal learning and scan-sequence confirmation rather than the ‘re-cognition’ proposed by the traditional models. As well, the strong generalisation effect of the SOM surfaces allows for some latitude (‘fuzziness’) in finding exact matches for scan-sequences, and local context allows for alternate decisions.

The model is low-level in that it only requires local Hebbian learning, while at the same time allowing for high-level feedback and context, thus providing a mechanism for priming and other top-down influences such as illusory contours. A massively parallel means of ‘information’ flow is described, with serial processing through temporal sequences and feedback.

The arbitrary formalisms of inductive and deductive logic are not required in the underlying model, but may be explained in terms of learned behaviour. ABC is able to separate perceptual learning—learning to recog-

nise and take appropriate behaviour with sensory inputs only, without the use of labels or language—and labelled (or supervised) learning. In this model, ‘parts’ are simply learned and labelled bifurcations of previously recognised objects, not fundamental components.

The non-uniformity of rods and cones does not present a problem as ABC does not attempt to ‘reconstruct’ an outside metric iconic view. All that is required is a series of vectors (at least one at each fixation point) which go to make up a temporal sequence. The ‘evidence’ of the rods and cones in the receptive fields is used to construct each vector element value. It does not matter that the point sources of data (rods and cones) are irregular in their distribution, so long as they are relatively *stable* (i.e., have fixed locations) over extended periods in the life of the animal. The rods and cones are combined to give various vectors such as spatial frequency and orientation (of various sizes and angles), colour and motion. In a sense, the choice of external world filter is arbitrary and at the whim of evolutionary pressures. Provided that each vector provides information to associate with, and discriminate other inputs, it can be employed on the same underlying ABC structural model.

Spatial fusion is not required—rather the model uses temporal integration to achieve recognition. The fill-in of the blind spot is also not an issue, as the inputs that are perceived to come from the blind spot are simply an averaging of the surrounding SOM values. Issues such as shading cues, being learned, will indeed retain their learned orientation—‘vertical’ relative to the head—associated with their usual body position.

The model does not require arbitrary representations or shape primitives, and does not require any form of segmentation. It requires only a retinal frame of reference.

The use of self-organising maps accounts for the re-learning and re-linkage of sensations across modalities, explaining the phenomenon described in the discordant stimulation experiments. For example, placing a prism in front of the eyes for an extended period will mean that the appropriate SOM surfaces in the vision modality will need to re-map the new sensations, and these new (displaced) visual inputs will need to be re-associated with other sensory inputs such as proprioception.

The system is indeed concept-based rather than object-based, allowing the labelling of things which could in no way be recognised as separable objects in the traditional schemes.

### 8.10.1 Recognition

Recognition in this model is of two types; instantaneous and temporal. Instantaneous recognition is achieved in essentially one fixation, and results when sufficient information is available in a single vector input to allow appropriate output behaviour to proceed. For example, one would expect

commonly viewed objects (or concepts such as facial expressions) to be readily recognised because of the learned associations between their input vectors and the linked recognition behaviour.

Temporal recognition, on the other hand, requires a number of fixation vectors to form a temporal sequence for recognition—the scan-sequence. It is the sequence rather than the individual vectors that leads to recognition.

Once an input vector impinges on a SOM surface near to a previously learned vector, then the generalisation on the SOM surfaces will allow the trace to lock-on to a scan-sequence and follow it for confirmation. Each subsequent confirmation loop will in turn give a predictive recurrent vector which will better locate the scan-sequence, and so on until the associated *concept* attractor is reached, leading to the appropriate behaviour previously associated with that attractor—such as perhaps naming the object.

## 8.11 Further Discussion Of The Model

In this section we discuss a number of issues which are consistent with, or explained by, the ABC model.

### 8.11.1 Relational Learning

Minsky and Papert [212] described a number of figures which are impossible to discriminate for any system that does not involve relational factors. Examples of these types of shapes are shown in Figure 8.29. If, for example, only local feature filters such as corners and line segment ‘ends’ were available (as per Figure 8.30), then all figures include four corners and two ends and are not separable. ,

Many have taken this to provide a powerful argument against the belief that an adequate model of human perception can be constructed which takes independent samples from local feature detectors. However, the temporal sequencing of samples from local feature detectors *is* able to differentiate figures such as these.

At each fixation point, a pair of values is stored—the local feature, either a corner or a line end, and an indicator of the eye position. The eye position indicators are given as numbers 1 to 4 on Figure 8.30. Each of the four shapes in Figure 8.29 then has a different temporal sequence of vectors if scanning of the objects is consistent. The sequence vectors for each of the four shapes are indicated in the text of Figure 8.30. A consistent scan of the images, then, will enable them to be discriminated.

The use of eye positions is justified as neurons sensitive to eye position have been found in the LNG [173], V1 [338, 356], and V3 [118], and also in the prefrontal cortex [117].

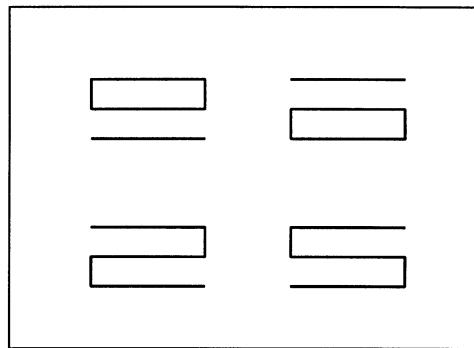


Figure 8.29: Minsky-Papert Figures.

### 8.11.2 World Filters

The initial component of the ABC model is a series of ‘filters’ which provide the outermost connection to the external world. For the human visual system, the likely filters are spatial frequency, colour, motion and possibly spatial frequency and colour combined. These at least are the filters chosen for the model implementation.

In the model, these filters are taken to be fixed. However, there is good evidence that at least some of the filters of higher animals undergo some form of self-organisation during development.

For example, Blakemore and Cooper [34] reared kittens in an environment consisting entirely of horizontal or vertical stripes. The kittens were allowed normal binocular vision, but their visual environment contained no corners or edges, and they were even prevented from seeing their own body by a wide black collar. Neurophysiological experiments performed after development was completed indicated cells of visual cortex (area 17) that

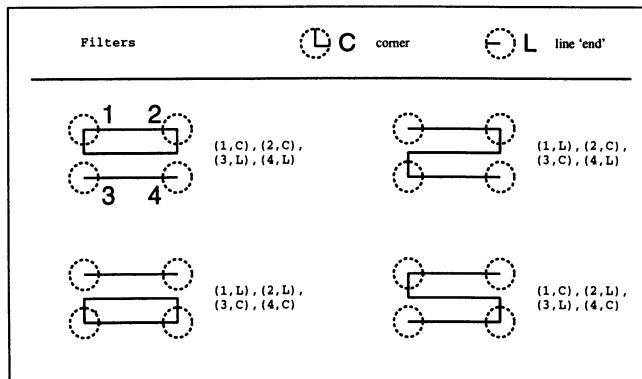


Figure 8.30: Minsky-Papert Figures Recognised Using Temporal Learning.

were mostly binocular, and in most other respects like those of a normal animal. However, the distribution of preferred orientation of the cells was quite abnormal, with no cell having an optimal orientation within  $20^\circ$  of the perpendicular to the environmental direction, and with very few cells responding to orientations within  $45^\circ$  of it. They did not attribute this to a passive degradation of certain cortical neurons due to under-activity, but instead concluded that the visual cortex had *adjusted* itself during maturation to the nature of their visual experiences.

This finding is consistent with a self-organising map learning the actual spatial frequencies experienced during the development period of the cat. We postulate a SOM surface as a component of the external world filter for spatial frequencies. Given that the newborn kitten has early receptive fields for all spatial frequencies, only those that it actually experiences during its development period will get mapped to the SOM surface of the external world filter. If the learning rate for this surface decreases to zero over the development period, then the distribution of spatial frequencies experienced during development will become ‘hard-wired’.

### 8.11.3 Receptive Fields in Periphery

The full vector taken in at each fixation point is strongly biased towards the fovea, but the larger receptive fields within the periphery are still represented. This is illustrated in Figure 8.31. This figure is incomplete as there are many more overlapping receptive fields at numerous distances from the fovea which is represented as the small black dot at the centre.

Kuffler [170] found that receptive fields vary in size across the retina. In the foveal region, the receptive fields are very small with a field centre of a few minutes of arc, whereas at the periphery the fields are much larger with a field centre of  $10$  to  $20^\circ$ .

Objects are usually learned at (or close to) the fovea, although it is possible to recognise objects in the periphery using the appropriate peripheral receptive field vector components. However, the vector components associated with the periphery are used to draw attention to objects not currently being fixated. This is especially the case for instances of motion and flicker in the periphery.

Multiple vectors are learned and associated with the same label [227], thus allowing recognition of the object from various viewpoints.

As an aside, particular components of the input foveal vector associated with the periphery may be used as the basis for learning relational terms such as ‘above’, ‘left’, ‘below’, and so on. For example, a high value found in the vector element associated with a peripheral receptive field above the fovea (indicating that something is *above* the current fixation point) may be used to initiate the contextual concept of ‘above’ (or ‘below’).

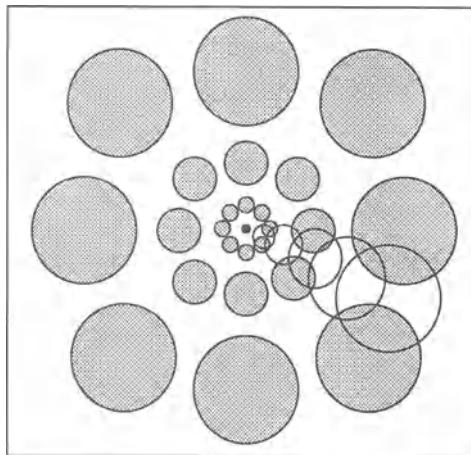


Figure 8.31: Receptive Fields.

#### 8.11.4 Treisman Figures

The ABC model is able to explain the constant versus linear search time experiments of Treisman and her coworkers [337]. In these experiments, subjects were asked to find particular target symbols in a scene with a number of distractors. For example, Figure 8.32 (a) illustrates a scene in which a target ‘white square’ is surrounded by black square and white circle distractors. The reaction time for finding the target object in these “conjunctive search” tasks is linear in the number of distractors. The task is conjunctive in the sense that the subject is required to search for a combination of features found in the distractors—here white and square.

In contrast, targets which have a feature different from the distractors tended to ‘pop-out’, giving a reaction time that is independent of the number of distractors. Treisman [337] termed this task a ‘feature search’.

Treisman and Gelade [337] showed that for a wide range of stimulus attributes, attention seems to be shifted serially across fixations when a subject is attempting to conjoin independent attributes of a stimulus.

As we discussed in Section 8.10.1, the ABC model can explain *instantaneous* recognition provided the vector obtained from the fixation has unique values for one or more of the vector elements. The feature search condition of Treisman is such a case—unique vector elements are present, and these correspond to a particular direction so that a subsequent saccade can be made directly to the target. The unique vector elements result from a particular target feature being found in one (or several adjacent) receptive fields, even in the periphery. Target features for this ‘pop-out’ are restricted to the external world filters—spatial frequencies and angles, colour and motion. That is, for a ‘feature search’, only the location in the periphery where

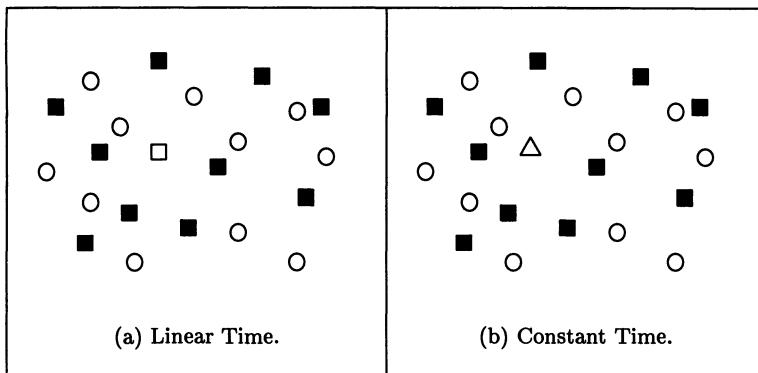


Figure 8.32: Treisman Visual Search.

the independent feature is found will a filter be exciting the corresponding neurons, and so attention will be directed to it in constant time.

The *linear* scan times are required in cases where unique vector elements are not found within any receptive field. To find the target in such cases, the subject is forced to saccade (possibly at random), but with not necessarily one fixation per distractor object, to bring target closer to the fovea. Because of prior learning, recognition is best at the fovea, and so the target must be brought to the fovea so that the learned associations and recognition behaviour can be undertaken.

### 8.11.5 Integration of Local Features

The traditional explanation for the eventual resolution of ‘difficult’ images such as Figure 8.4 (the dalmatian dog) is that the “percept must result from the synthesis of contour information gathered from over a large portion of the picture. Looking at only local regions of the picture, as cortical cells do, would never lead to the recognition of the scene and the central object within that scene. Rather, we see the scene and the object because of some more widespread, or global, comparison of local contour information” [307].

The ABC model suggests a different explanation. The so-called *global* contours are simply not there, and recognition is achieved instead through finding a sequence of smaller foveal images (in which some local *illusory contours* may be generated by spatial frequency filters) such that the vectors generated by the local images *approximates* those corresponding to the actual object (the dalmatian dog), and the sequence corresponds to a learned scan-sequence for that object. The generalisation processes on the SOM surfaces will enable the ‘noisy’ vectors to *latch on* to the sequence used for recognition of the object, and once this partial recognition of a ‘possible’ object is achieved, then the subsequent scan-sequence may be followed for confirmation.

When the recognition of the object is ‘confirmed’, recursive contextual information (top-down knowledge) makes it easier to find additional cues and to latch on to a scan-sequence.

### 8.11.6 Polysemous Figures

Polysemous images such as the famous ‘Wife and Mother-in-law’ image shown in Figure 8.3 present a problem for traditional vision theories. While the low-level stimulus information reaching the eye (such as ‘contours’) remains constant, the interpretation of the image may fluctuate between a picture of a coy wife and one of the ugly old mother-in-law.

However, an explanation is forthcoming if we consider scan-sequences. The alternating interpretations may be achieved through the alternation of several different scan-sequences, one for each interpretation of the image. If the scan-sequence associated with the wife is currently being employed by the subject, then the wife interpretation will be the accompanying percept, whereas following the scan-sequence associated with the mother-in-law will produce a perception of that figure.

### 8.11.7 Reinforcement & Learning

The ABC model employs a reinforcement learning strategy. Some have maintained that conventional reinforcement learning faces a serious difficulty. The assumption that the brain creates and maintains a picture-perfect visual scene at each moment creates the problem of just how the brain determines which of the many objects and features in its current world view are the appropriate ones to be reinforced. How does a naive brain determine which “stimulus” in the richly detailed stimulus array is to be credited following a reward—which synapses should be strengthened?

Churchland et al. [67] suggest that some ‘attention’ mechanism is the solution. They suggest that perhaps the brain has been hard-wired by evolution to bias attention towards properties which are relevant to the survival of a particular species. Their point is that an attended feature of the visual environment (either directly or via some iconic or working memory) is more likely to be causally associated as being relevant for reinforcement than when a rich-replica visual world is used as the representation.

Our view is similar, with fixation visual sequences taking the place of the attention mechanism. The primarily foveal views of each fixation form the component to be reinforced, and it is generally only when an object or event is at, or near, the fovea that it is ‘identified’. The larger receptive fields of the retinal periphery are less involved with the recognition process (although may still participate) and serve more to bring items of interest to the ‘attention’ of the creature and cause it to foveate to the area of interest. While the vector formed at each fixation has a preponderance of

information from the well-sampled foveal areas, it still includes information from the periphery that may be used for recognition.

The neonate will initially shift attention in a more or less random manner (ignoring any innate reflexive behaviour), and the regularities of the external world, along with any exploration by the organism, will result in linkages forming between the various sensory modality inputs and behaviour through associative reinforcement. The organism will tend to reproduce options which were successful in the past, further strengthening any associations [357, 358]. The approach is that of standard operant conditioning in which a positively reinforced response is more likely to be reproduced when similar conditions are subsequently encountered [135, 161, 191, 283].

Reinforcement learning has had a number of strong critics in the past, mainly as a result of the seeming plethora of information that a creature has available to it, and the above-mentioned need for some means of separating out the relevant details for associative reinforcement. Some have even gone as far as suggesting that reinforcement learning cannot be the mechanism appropriate to the sophisticated learning typical of higher cognitive organisms (e.g., Chomsky [63, 64] and Fodor [114]).

The ABC model does not suffer from this inability to associate appropriate sensory inputs for reinforcement. In the ABC vision model, for example, the foveal vector (including any peripheral receptive field information that may be relevant) is exactly the type of attentional detail needed to form a relevant causal link. Only the specific detail of the current (and perhaps some preceding) temporal sequences obtained from attentional fixations is associated with the reward or punishment of the reinforcement signal.

Churchland et al. [67] suggest that a further skepticism of reinforcement learning as a cognitive learning model results from neural net modelling. Neural nets appear to scale poorly as the number of dimensions of the input space increases to supposedly approximate the rich requirements of a realistic visual scene, resulting in inappropriately long training times and thus rendering reinforcement learning as impractical for complex task domains. However, this criticism of the supposed slowness of reinforced learning in neural nets misses a number of crucial points, and shows a lack of understanding of biological neural structures.

Biological neural structures (and the ABC model) are massively parallel *hardware* systems. The update time is constant no matter how many input lines there are as only local (hardware) interactions are involved. In fact, the more lines the better as it enables better discrimination. A *serial simulation* of a neural net on a von Neuman computer, however, will scale poorly as the number of input lines increases, but this is not relevant. This is a limitation of the simulation vehicle (the serial computer), not of the neural model. A suitable hardware simulation, as discussed in [42], would not suffer from this deficit.

Another issue ignored is that learning with artificial neural networks

depends upon a number of parameter settings, such as the learning rate. In biological neural nets this is also the case. Faster learning is achieved with a *high-gain* resulting from a high *value-based* (or ‘emotional’) component. Learning not to put a hand in a fire a second time is an example—the initial pain will result in a sufficiently high gain that the appropriate learned association is achieved very rapidly.

The final point ignored is that for humans and animals, learning is performed over the whole development period and even into adulthood. For humans especially, this is an extended period.

## 8.12 Conclusion

The ABC model of vision, described in this chapter, is very different in many respects from the good-old-fashioned hierarchical recognition proposal of traditional theories. Recognition within the model (including predictive, what-next recognition) depends on richly recurrent neural networks, and involves associations between the various modalities. This rich recurrence, especially with continuing multi-cortical area input, challenges the conventional conception of a chiefly unidirectional low-to-high processing hierarchy.

There is no real distinction between learning and recognition in the model, both taking place concurrently. The model is dynamic and continuous, and provides very good agreement with experimental findings.

The linkage of self-organising maps, association layers, and recurrency employed in the ABC model shows much promise in providing a real alternative to the established computational paradigm and to finding a better understanding of the processes of cognition.

# References

- [1] Yaser S. Abu-Mostafa and Demetri Psaltis. Recognitive aspects of moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):698–706, November 1984.
- [2] Yaser S. Abu-Mostafa and Demetri Psaltis. Image normalisation by complex moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):46–55, January 1985.
- [3] H. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computing*, C-23:90–93, 1974.
- [4] James Allen. *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, 1987.
- [5] Y. Aloimonos. Introduction: Active Vision revisited. *Active Perception*, pages 1–18, 1993.
- [6] Y. Aloimonos. REPLY: What I have learned. *CVGIP: Image Understanding*, 60(1):74–85, July 1994.
- [7] Jürgen Altmann and Herbert J. P. Reitböck. A fast correlation method for scale- and translation-invariant pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):46–57, January 1984.
- [8] S. Amari and K. Maginu. Statistical neurodynamics of associative memory. *Neural Networks*, 1(1):63–73, 1988.
- [9] Adnan Amin, Claude Sammut, and K. C. Sum. Learning to recognize hand-printed Chinese characters using inductive logic programming. Technical report, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia., 1996.
- [10] Stig K. Andersen, Kristian G. Olesen, Finn V. Jensen, and Frank Jensen. HUGIN – A shell for building Bayesian belief universes for expert systems. In *IJCAI89*, pages 1080–1085, Detroit, August 1989.

- [11] E. Andre, G. Herzog, and T. Rist. On the simultaneous interpretation of real world image sequences and their natural language descriptions: the system SOCCER. In Y. Kodratoff, editor, *ECAI 88. Proceedings of the 8th European Conference on Artificial Intelligence*, pages 449–54, UK, 1988.
- [12] Michael A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. The MIT Press: A Bradford Book: Cambridge, MA, 1995.
- [13] R. N. Baber. The impending demise of the icon: A critique of the concept of iconic storage in visual information processing. *Behavioral and Brain Sciences*, 6:1–54, 1983.
- [14] R. Bajcsy. Passive perception vs. active perception. In *Proc. IEEE Workshop on Computer Vision*, pages 333–356, 1986.
- [15] R. Bajcsy. Active perception. *Proc. of the IEEE*, 76(8):996–1005, 1988.
- [16] J. F. Baldwin. Support logic programming. *International Journal of Intelligent Systems*, 1:73–104, 1986.
- [17] D. Ballard and C. Brown. Principles of animate vision. *CVGIP:Image Understanding*, 56(1):3–21, July 1992.
- [18] Dana Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1982.
- [19] Dana H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.
- [20] Etienne Barnard and Elizabeth C. Botha. Back-propagation uses prior information efficiently. *IEEE Transactions on Neural Networks*, 4(5):794–802, September 1993.
- [21] H. G. Barrow and R. M. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters*, 4(4):83–84, January 1976.
- [22] S. O. Belkasim, M. Shridhar, and M. Ahmadi. Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognition*, 24(12):1117–1138, 1991.
- [23] B. Bell and L. F. Pau. Context knowledge and search control issues in object-oriented prolog-based image understanding. *Pattern Recognition Letters*, 13(4):279–290, April 1992.

- [24] Robert Bergevin and Martin D. Levine. Generic object recognition: building and matching coarse descriptions from line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):19–36, January, 1993.
- [25] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum, New York, 1981.
- [26] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.
- [27] I. Biederman, A. L. Glass, and E. W. Stacy. Searching for objects in real world scenes. *J. Exp. Psychol.*, 97:22–27, 1973.
- [28] I. Biederman, J. C. Rabinowitz, A. L. Glass, and E. W. Stacy. On the information extracted from a glance at a scene. *J. Exp. Psychol.*, 103:597–600, 1974.
- [29] Irving Biederman. Human image understanding: recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32:29–73, 1985.
- [30] W. F. Bischof and T. Caelli. Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation. *Pattern Recognition*, 27(5):689–697, 1994.
- [31] Walter F. Bischof and Terry F. Caelli. Learning how to find patterns or objects in complex scenes. In C. Braccini, L. De Floriani, and G. Vernazza, editors, *Proceedings of the International Conference on Image Analysis and Processing (ICIAP95)*, pages 287–292. Springer-Verlag, 1995.
- [32] A. Blake and A. Yuille, editors. *Active Vision*. Cambridge, MA: MIT Press, 1992.
- [33] C. Blakemore. Development of the mammalian visual system. *Brit. Med. Bull.*, 30:152–157, 1970.
- [34] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, 1970.
- [35] R. C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. *Int. J. of Robotics Research*, 1986.
- [36] G. Borgefors. Distance transformations in digital images. *CVGIP*, 34:344–371, 1986.
- [37] Chinmoy B. Bose and Shyh-Shiaw Kuo. Connected and degraded text recognition using hidden Markov model. *Pattern Recognition*, 27(10):1345–1363, 1994.

- [38] Kevin W. Bowyer, Lawrence O. Hall, Pat Langley, Bir Bhanu, and Bruce A. Draper. Report of the AAAI fall symposium on machine learning and computer vision, 1993.
- [39] T. Brandt, J. Dichgans, and E. Koenig. Differential effects of central versus peripheral vision on egocentric and exocentric motion perception. *Exp. Brain Res.*, 16:476–491, 1973.
- [40] I. Bratko and S. Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, 1995.
- [41] Bruce Bridgeman, A. H. C. Van der Heijden, and Boris M. Velichkovsky. A theory of visual stability across saccadic eye movements. *Behavioral and Brain Sciences*, 17:247–292, 1994.
- [42] G. Briscoe. *Adaptive Behavioural Cognition*. PhD thesis, School of Computing, Curtin University of Technology, Perth, Western Australia, 1997.
- [43] G. Briscoe and T. Caelli. *A Compendium of Machine Learning*. Ablex, New York, 1996.
- [44] Roberto Brunelli and Stefano Messelodi. Robust estimation of correlation with applications to computer vision. *Pattern Recognition*, 28(6):833–841, 1995.
- [45] Leonard T. Bruton and Norman R. Bartley. The enhancement and tracking of moving objects in digital images using adaptive three-dimensional recursive filters. *IEEE Transactions on circuits and systems*, 33(6):604–612, June 1986.
- [46] A. Bulsari. Some analytical solutions to the general approximation problem for feedforward neural networks. *Neural Networks*, 6(7):991–996, 1993.
- [47] H. Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):574–82, 1982.
- [48] H. Bunke and B. Messmer. Recent advances in graph matching. In T. Caelli, H. Bunke, and P. Lam, editors, *Spatial Computing*. World Scientific, Singapore, 1996.
- [49] J. Brian Burns, Richard S. Weiss, and Edward M. Riseman. View variation of point-set and line-segment features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):51–68, January, 1993.

- [50] T. Caelli and A. Dreier. Variations on the evidence-based object recognition theme. *Pattern Recognition*, 27(2):185–204, 1994.
- [51] T. M. Caelli, G. A. N. Preston, and E. R. Howell. Implications of spatial summation models for processes of contour perception: A geometric perspective. *Vision Research*, 18:723–734, 1978.
- [52] Terry Caelli and Peter Dodwell. Orientation-position coding and invariance characteristics of pattern discrimination. *Perception and Psychophysics*, 36(2):159–168, 1984.
- [53] Terry M. Caelli and Zhi-Qiang Liu. On the minimum number of templates required for shift, rotation and size invariant pattern recognition. *Pattern Recognition*, 21(3):205–216, 1988.
- [54] Terry M. Caelli, David McG. Squire, and Tom P. J. Wild. Model-based neural networks. *Neural Networks*, 6:613–625, 1993.
- [55] F. W. Campbell and J. G. Robson. Application of Fourier analysis to the visibility of gratings. *J. Physiol.*, 197:551–566, 1968.
- [56] J. F. Canny. A computational approach to edge detection. *IEEE: Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [57] John F. Canny. Finding edges and lines in images. Technical Report AI-TR-720, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA, 1983.
- [58] D. Carevic and T. Caelli. Region-based coding of color images using Karhunen-Loeve transform. *Graphical Models and Image Processing*, 59(1):27–38, January 1997. 8849.
- [59] D. P. Carmody, C. F. Nodine, and H. L. Kundel. Global and segmented search for lung nodules of different edge gradients. *Invest. Radiol.*, 15:224–233, 1980.
- [60] Robin L. P. Chang and Theodosios Pavlidis. Fuzzy decision tree algorithms. *IEEE Transactions Systems, Man and Cybernetics*, SMC-7(1):28–35, January 1977.
- [61] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AUTOCLASS: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64. Morgan Kaufmann, San Mateo, CA, 1988.
- [62] Y-S. Chen and W-H. Hsu. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recognition*, 7:99–106, 1988.

- [63] N. Chomsky, editor. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA, 1965.
- [64] N. Chomsky, editor. *Rules and Representations*. Columbia University Press, New York, 1980.
- [65] Gloria Chow and Xiaobo Li. Towards a system for automatic facial feature detection. *Pattern Recognition*, 1993.
- [66] Chen-Chau Chu and J. K. Aggarwal. The integration of image segmentation maps using region and edge information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1241–1252, December 1993.
- [67] Patricia S. Churchland, V. S. Ramachandran, and Terrence J. Sejnowski. A critique of pure vision. In Christof Koch and Joel L. Davis, editors, *Large-Scale Neuronal Theories of the Brain*, pages 23–60. A Bradford Book: The MIT Press, 1994.
- [68] W. J. Clancey. Situated cognition: How representations are created and given meaning. In *American Educational Research Association (AERA) 1991 Symposium*, Chicago, 1991.
- [69] Axel Cleeremans. *Mechanisms of Implicit Learning: Connectionist Models of Sequence Processing*. The MIT Press, A Bradford Book; Cambridge, MA, 1993.
- [70] V. Clement and M. Thonnat. Supervision of perception tasks for autonomous systems: The OCAPI approach. *IEEE*, pages 210–217, May 1992.
- [71] C. Colby. The neuroanatomy and neurophysiology of attention. *J. Child Neurol. Suppl.*, 6:S90–S118, 1991.
- [72] James B. Cole, Hiroshi Murase, and Seiichiro Naito. A Lie group theoretic approach to the invariance problem in feature extraction and object recognition. *Pattern Recognition Letters*, 12:519–523, September 1991.
- [73] Tom Conlon. *Programming in PARLOG*. Addison-Wesley, Menlo Park, Ca., 1989.
- [74] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray. Using vector quantization for image processing. *Proceedings of IEEE*, 81(9), September 1993.
- [75] S. Culhane and J. Tsotsos. A prototype for data-driven visual attention. In *Proceedings 11th IAPR International Conference on Pattern Recognition*, pages 36–40, The Hague, The Netherlands, 1992. IAPR.

- [76] Sandy Dance and Terry Caelli. On the symbolic interpretation of traffic scenes. In *ACCV93 Proceedings of the Asian Conference on Computer Vision*, pages 798–801, Osaka Japan, November 1993.
- [77] Sandy Dance and Terry Caelli. A symbolic object-oriented picture interpretation network: SOO-PIN. In Horst Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition, Proceedings of the International Workshop*, Machine perception and Artificial intelligence, pages 530–541, Bern, Switzerland, 1993. World Scientific Publishing Co.
- [78] Sandy Dance, Terry Caelli, and Zhi-Qiang Liu. *Picture Interpretation: a Symbolic Approach*. Series in Machine Perception and Artificial Intelligence. World Scientific Publishing Co., Singapore., 1995.
- [79] Larry Davis and Azriel Rosenfeld. Cooperating processes for low-level vision: A survey. *Artificial Intelligence*, 17:245–263, 1981.
- [80] Andrew Davison. From Parlog to Polka in two easy steps. In *PLILP'91: 3rd Int. Symp. on Programming Language Implementation and LP*, number 528 in Springer LNCS, pages 171–182, Passau, Germany, August 1991. Springer.
- [81] A. D. De Groot. *Thought and Choice in Chess*. Mouton, The Hague, 1978.
- [82] Arthur P. Dempster. A generalization of Bayesian inference. *Journal of the Royal Statistical Society*, 30:205–247, 1968.
- [83] Sven J. Dickinson, Alex P. Pentland, and Azriel Rosenfeld. 3D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):174–198, February, 1992.
- [84] Sven J. Dickinson, Alex P. Pentland, and Azriel Rosenfeld. From volumes to views: an approach to 3D object recognition. *Computer Vision, Graphics and Image Processing: Image Understanding*, 55(2):130–154, Mar, 1992.
- [85] M. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39(2):253–280, April 1992.
- [86] Craig Dillon. *A Theory of Scene Understanding and Object Recognition*. PhD thesis, School of Computing, Curtin University of Technology, Perth, WA, 1996.

- [87] Yannis A. Dimitriadis and Juan Lopez Coronado. Towards an ART-based mathematical editor, that uses on-line handwritten symbol recognition. *Pattern Recognition*, 28(6):807–822, 1995.
- [88] D.L. Dowe, J. Oliver, T.I. Dix, L. Allison, and C.S. Wallace. A decision graph explanation of protein secondary structure prediction. In *26th Hawaii International Conference on Systems Sciences*, 1993.
- [89] B. Draper, T. Collins, J. Brolio, A. Hanson, and E. Riseman. The Schema system. *International Journal of Computer Vision*, 2:209–250, 1989.
- [90] D. Dubois and H. Prade. A discussion of uncertainty handling in support logic programming. *International Journal of Intelligent Systems*, 5(1):15–42, March 1990.
- [91] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley, New York, 1973.
- [92] F. H. Duffy and J. L. Burchfield. Eye movement-related inhibition of primate visual neurons. *Brain Res.*, 89:121–132, 1975.
- [93] S. Edelman and D. Weinshall. A self-organising multiple-view representation of 3D objects. *Biological Cybernetics*, 64:209–219, 1991.
- [94] Shimon Edelman. On learning to recognise 3D objects from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):833–837, August 1993.
- [95] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
- [96] Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99, 1993.
- [97] C. W. Eriksen and B. A. Eriksen. Visual perception processing rates and backward and forward masking. *J. Exp. Psychol.*, 89:306–313, 1971.
- [98] S. F. Erlich and K. Rayner. Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, 20:641–655, 1981.
- [99] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 38–51. Morgan Kaufmann, San Mateo, CA, 1990.

- [100] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, September 1988.
- [101] Ting-Jun Fan, Gerard Medioni, and Ramakant Nevatia. Recognizing 3-D objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11), November 1989.
- [102] U. Fayyad and K. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.
- [103] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Psychology*, 6:205–254, 1982.
- [104] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1. Wiley and Sons, New York, 3rd edition, 1968.
- [105] M. Ferraro and T. Caelli. The relationship between integral transform invariances and Lie group theory. *Journal of the Optical Society of America (A)*, 5:738–742, 1988.
- [106] Mario Ferraro and Terry M. Caelli. Lie transform groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8(1):33–44, 1994.
- [107] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computing*, 22:67–92, 1973.
- [108] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.
- [109] P. Flynn and A. Jain. *Handbook of Pattern Recognition and Image Processing*, volume 2. New York: Academic Press, 1993.
- [110] P. J. Flynn and A. K. Jain. CAD-based computer vision: From CAD models to relational graphs. *T-PAMI*, 13(2):114–132, February 1991.
- [111] Patrick Flynn and Anil Jain. BONSAI: 3D object recognition using constrained search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1066–1075, October 1991.
- [112] Patrick J. Flynn. Saliencies and symmetries: Toward 3D object recognition from large model databases. In *Proceedings CVPR 1992*, pages 322–327, Champaign, Il., 1992. IEEE.

- [113] Patrick J. Flynn and Anil K. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *Computer Vision Graphics and Image Processing: Image Understanding*, 55(2):119–129, 3 1992.
- [114] J. A. Fodor, editor. *Representations*. MIT Press, Cambridge, MA, 1981.
- [115] David Forsyth, Joseph L. Mundy, and Andrew Zisserman. Transformational invariance - A primer. *Image and Vision Computing*, 10(1):39–45, 1992.
- [116] J. M. Fuster. Inferotemporal units in selective visual attention and short-term memory. *J. Neurophysiol.*, 64:681–697, 1990.
- [117] Joaquin M. Fuster. The prefrontal cortex and temporal integration. In Alan Peters and Edward G. Jones, editors, *Cerebral Cortex*, volume 4, Association and Auditory Cortices, chapter 4, pages 151–177. Plenum Press: NY, 1985.
- [118] C. Galletti and P. P. Battaglini. Gaze-dependent visual neurons in area V3A of monkey prestriate cortex. *J. Neurosci.*, 9:1112–1125, 1989.
- [119] Eleanor J. Gibson. *Principles of Perceptual Learning and Development*. Appleton-Century-Crofts, New York, 1969.
- [120] J. J. Gibson. *The perception of the visual world*. Boston: Houghton Mifflin, 1950.
- [121] James Gibson. *The Senses Considered as Perceptual Systems*. Houghton-Mifflin, Boston, Massachusetts, 1966. Also published by George Allen & Unwin Ltd, 1968.
- [122] G.-J. Giefing, H. Janßen, and H. Mallot. Saccadic object recognition with an active vision system. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition*, volume I, pages 664–667. IEEE Computer Society Press, 1992.
- [123] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.
- [124] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The Latex Companion*. Addison-Wesley, 1994.
- [125] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

- [126] W. Eric. L. Grimson. *Object Recognition By Computer: The Role of Geometric Constraints*. The MIT Press, 1990.
- [127] Rudolf Groner and Christine Menz. The effect of stimulus characteristics, task requirements and individual differences on scanning patterns. In R. Groner, G. W. McConkie, and C. Menz, editors, *Eye Movement and Human Information Processing*, pages 239–250. Elsevier Science Publishers B.V. (North-Holland), 1985.
- [128] Rudolf Groner, Franziska Walder, and Marina Groner. Looking at faces: Local and global aspects of scanpaths. In A. G. Gale and F. Johnson, editors, *Theoretical and Applied Aspects of Eye Movement Research*, pages 523–533. Elsevier Science Publishers B.V. (North-Holland), 1984.
- [129] D. H. Grossof, R. M. Shapley, and M. J. Hawken. Macaque striate responses to anomalous contours? *Invest. Ophthalmol. Vis. Sci.*, 33:1257, 1992.
- [130] Selim S. Hacisalihzade, Lawrence W. Stark, and John S. Allen. Visual perception and sequences of eye movement fixations: A stochastic modeling approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):474–481, 1992.
- [131] John F. Haddon and James F. Boyce. Image segmentation by unifying region and boundary information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):929–948, October 1990.
- [132] P. E. Hallet and B. D. Adams. The predictability of saccadic latency in a novel voluntary oculomotor task. *Vision Research*, 20:329–339, 1980.
- [133] D. O. Hebb. *The organization of behavior*. New York: Wiley, 1949.
- [134] Steven J. Henkind and Malcolm C. Harrison. An analysis of four uncertainty calculi. *IEEE Transactions on Systems, Man and Cybernetics*, 18(5):700–714, 1988.
- [135] Wendon W. Henton and Iver H. Iverson. *Classical conditioning and operant conditioning :A response pattern analysis*. New York : Springer-Verlag, 1978.
- [136] Carl Hewitt. How to use what you know. In *IJCAI75*, pages 189–198, Tbilisi, Georgia, September 1975.
- [137] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8:323–363, 1977.

- [138] J. Hochberg. Form perception: Experience and explanations. In P. C. Dodwell and T. M. Caelli, editors, *Figural synthesis*, pages 1–30. Hillsdale, NJ: Erlbaum, 1984.
- [139] W. C. Hoffman. The Lie algebra of visual perception. *Journal of Mathematical Psychology*, 3:65–98, 1966.
- [140] W. C. Hoffman. The Lie transformation group approach to visual neuropsychology. In E. Leewenberg and H. Buffart, editors, *Formal theories of visual perception*, pages 27–66. Wiley, New York, 1978.
- [141] J. E. Hopcraft and R. M. Karp. A  $n^{\frac{5}{2}}$  algorithm for maximum matching in bipartite graphs. *J. SIAM Comp*, 2:225–231, 1973.
- [142] Ian P. Howard, editor. *Human Visual Orientation*. John Wiley & Sons, 1982.
- [143] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions Information Theory*, IT-8:179–187, February 1962.
- [144] Timothy T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber. Automatic symbolic traffic scene analysis using belief networks. In *Proceedings of the American Association of Artificial Intelligence Conference (AAAI-94)*, pages 966–972, Seattle, August 1994.
- [145] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.
- [146] R. A. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):267–287, May 1983.
- [147] K. Ikeuchi and M. Hebert. Task oriented vision. In *1990 DARPA Image Understanding Workshop Proceedings*, pages 497–507, September 1990.
- [148] K. Ikeuchi and T. Kanade. Automatic generation of object recognition programs. *Proceedings of the IEEE*, 76(8):1016–1035, 1988.
- [149] H. Intraub. Presentation rate and the representation of briefly glimpsed pictures in memory. *J. Exp. Psychol. Hum. Learn. Mem.*, 6:1–12, 1980.
- [150] D. E. Irwin, S. Yantis, and J. Jonides. Evidence against visual integration across eye movements. *Percept. Psychophys.*, 34:49–57, 1983.

- [151] David E. Irwin. Perceiving an integrated visual world. In *Attention and Performance XIV*, pages 121–142. The MIT Press: Cambridge, MA, 1993.
- [152] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.
- [153] Anil Jain and Richard Hoffman. Evidence-based recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):783–802, November 1988.
- [154] Anil K. Jain. A fast Karhunen-Loeve transform for finite discrete images. In *Proceedings of the National Electronics Conference*, pages 323–328, Chicago, IL, October 1974.
- [155] Anil K. Jain. *Fundamentals of digital image processing*. Prentice-Hall information and system sciences series. Prentice-Hall International, London, 1989.
- [156] R. Jain and T. Binford. DIALOGUE: Ignorance, myopia and naivete in computer vision systems. *CVGIP:Image Understanding*, 53(1):112–117, January 1991.
- [157] M. I. Jordan. Serial order: A parallel, distributed processing approach. In J. L. Elman and D. E. Rumelhart, editors, *Advances in Connectionist Theory: Speech*. Hillsdale, NJ: Erlbaum, 1989.
- [158] Michael I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 112–127, 1986.
- [159] Michael I. Jordan and David A. Rosenbaum. Action. In Michael I. Posner, editor, *Foundations of Cognitive Science*, chapter 18, pages 727–767. The MIT Press: A Bradford Book: Cambridge, MA, 1989.
- [160] Avinash Kak and Whoiyul Kim. 3-D object recognition using bipartite matching embedded in discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224–251, March 1991.
- [161] Eric R. Kandel and James H. Schwartz. *Principles of Neural Science*. Elsevier/North-Holland, 1983.
- [162] Jari Kangas. Time-delayed self-organizing maps. *Proc. IJCNN'89 Int. Joint Conf. on Neural Networks*, II:331–336, 1989.
- [163] Jari Kangas. Time-dependent self-organizing maps for speech recognition. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors,

- Artificial Neural Networks - Volume 2 (Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91), Espoo, Finland, (June 1991), pages 1591–1594. North-Holland, 1991.*
- [164] I. Kant. *Critique of Pure Reason*. Willey Book Co., 1943.
  - [165] D. Kapur and J. L. Mundy. Wu's method and its application to perspective viewing. *Artificial Intelligence*, 37:15–36, 1988.
  - [166] Alireza Khotanzad and Jiin-Her Lu. Classification of invariant image representations using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):1028–1038, June 1990.
  - [167] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem, Its Structural Complexity*. Birkhäuser, 1993.
  - [168] B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice Hall, 1992.
  - [169] Raghu Krishnapuram and James M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, May 1993.
  - [170] S. W. Kuffler. Neurons in the retina: Organization, inhibition and excitation problems. *Cold Spring Harbor Symp. Quant. Biol.*, 17:281–292, 1952.
  - [171] Kurta. *XGT Serial Digitizing Tablet Owner's Guide; Kurta Corporation*. Kurta Corporation, Phoenix, Arizona, USA, 1994.
  - [172] George Lakoff. *Women, Fire and Dangerous Things*. University of Chicago Press, 1987.
  - [173] Ratneshwar Lal and Michael J. Friedlander. Gating of retinal transmission by afferent eye position and movement signals. *Science*, 243:93–96, 1989.
  - [174] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:33–43, 1990. Reproduced in *Readings in Machine Learning* [314], pages 150–170.
  - [175] Steffen L. Lauritzen. *Graphical Models*. Oxford statistical science series 17, Oxford science publications. Oxford : Clarendon Press ; New York : Oxford University Press, 1996.
  - [176] Chung-Mong Lee, Ting-Chuen Pong, Albert Esterline, and James Slagle. KOR: A knowledge-based object recognition system. In L. Shapiro and A. Rosenfeld, editors, *Computer Vision and Image Processing*. Academic Press, Inc., 1992.

- [177] Chung-Mong Lee, Ting-Chuen Pong, James Slagle, and Albert Esterline. An experimental study of an object recognition system that learns. *Pattern Recognition*, 27(1):65–89, 1994.
- [178] Seong-Whan Lee. Translation-, rotation- and scale- invariant recognition of hand-drawn symbols in schematic diagrams. *International journal of pattern recognition and artificial intelligence*, 4(1):1–25, 1990.
- [179] Reiner Lenz. Group invariant pattern recognition. *Pattern Recognition*, 23(1/2):199–217, 1990.
- [180] Ming Li and Paul M. B. Vitanyi. Inductive reasoning and Kolmogorov complexity. *Journal of Computer and System Sciences*, 44:343–384, 1992.
- [181] S. Z. Li. Matching: Invariant to translations, rotations and scale changes. *Pattern Recognition*, 25(6):583–594, 1992.
- [182] Xiaobo Li and Nicholas Roeder. Experiments in detecting face contours. In *Vision Interface Conference*, May 1994.
- [183] Feng Lin and Robert D. Brandt. Towards absolute invariants of images under translation, rotation and dilation. *Pattern Recognition Letters*, 14:369–379, May 1993.
- [184] Margaret Livingstone and David Hubel. Segregation of form, color, movement, and depth: Anatomy, physiology, and perception. *Science*, 240:740–749, 1988. Reproduced in *Biology and Computation: A Physicist’s Choice, Reprint Volume*, edited by H. Gutfreund and G. Toulouse. Advanced Series in Neuroscience – Vol. 3; World Scientific, 1994.
- [185] Paul J. Locher and Calvin F. Nodine. The role of scanpaths in the recognition of random shapes. *Perception & Psychophysics*, 15(2):308–314, 1974.
- [186] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.
- [187] John D. Lowrance, Thomas M. Strat, Leonard P. Wesley, Thomas D. Garvey, Enrique H. Ruspini, and David E. Wilkins. The theory, implementation and practice of evidential reasoning. SRI Project 5701, SRI International, Menlo Park, CA 94025, June 1991. Final report.
- [188] Si Wei Lu, Ying Ren, and C. Y. Suen. Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition*, 24(7):617–32, 1991.

- [189] D. M. MacKay. Ways of looking at perception. In W. Wathen-Dunn, editor, *Models for the perception of speech and visual form*, pages 25–43. Cambridge, MA: MIT Press, 1967.
- [190] D. M. MacKay. Visual stability and voluntary eye movements. In R. Jung, editor, *Handbook of sensory physiology*, volume VII/3A, pages 307–331. Berlin: Springer, 1973.
- [191] N. J. Mackintosh. *The Psychology of Animal Learning*. Academic Press, New York, 1974.
- [192] N. H. Mackworth and A. J. Morandi. The gaze selects informative details within pictures. *Percept. Psychophys.*, 2:547–552, 1967.
- [193] Sidhartha Maitra. Moment invariants. *Proceedings of the IEEE*, 67(4):697–699, April 1979.
- [194] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. Lond.*, 200:269–294, 1978.
- [195] David Marr. *Vision*. San Francisco: Freeman, 1982.
- [196] David C. Marr. Representing visual information. In Werner E. Reichardt and Tomaso Poggio, editors, *Theoretical Approaches in Neurobiology*, pages 151–166. The MIT Press, Cambridge, MA, 1978.
- [197] G. Masini and R. Mohr. MIRABELLE, a system for structural analysis of drawings. *Pattern Recognition*, 16(4):363–72, 1983.
- [198] T. Matsuyama and V. Hwang. *Sigma: A Knowledge-Based Aerial Image Understanding System*. Plenum Press, 1990.
- [199] J. H. Maunsell, G. Sclar, T. A. Nealey, and D. D. DePriest. Extraretinal representations in area V4 in the macaque monkey. *Vis. Neurosci.*, 7:561–573, 1991.
- [200] Brendan McCane and Terry Caelli. Multi-scale adaptive segmentation using edge and region based attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995. Under Revision.
- [201] Brendan McCane and Terry Caelli. Multi-scale adaptive segmentation using edge and region based attributes. Technical Report 95/10, James Cook University of North Queensland, 1995. <http://www.cs.jcu.edu.au/ftp/web/research/techreports/ftp95.html#95-10>.
- [202] Brendan McCane, Terry Caelli, and Olivier de Vel. Learning and recognising 3D objects from 2D views. Technical Report 95/12, James Cook University of North Queensland, 1995.

- [203] Brendan J. McCane. *Learning to recognise 3D objects from 2D intensity images*. PhD thesis, James Cook University of North Queensland, Qld, Australia, 4811, 1996.
- [204] Brendan J. McCane and Olivier de Vel. A stereo matching algorithm using curve segments and cluster analysis. Technical Report 94/5, Dept. Computer Science, James Cook University, 1994. <http://www.cs.jcu.edu.au/ftp/web/research/techreports/ftp94.html#94-5>.
- [205] G. McConkie. On the role and control of eye movements in reading. In P. A. Kolers, M. E. Wrolstad, and H. Bouma, editors, *Processing of visual language*, pages 37–48. New York: Plenum, 1979.
- [206] G. McConkie. Where vision and cognition meet, 1990. Paper presented at the *H.F.S.P. Workshop on Object and Scene Perception*, Leuven, Belgium.
- [207] B. T. Messmer and H. Bunke. Subgraph isomorphism detection in polynomial time on preprocessed model graphs. In S. Li, D. Mital, E. Teoh, and H. Wan, editors, *Recent Developments in Computer Vision. Second Asian Conference on Computer Vision, ACCV '95, Singapore, Invited Session Papers*, pages 373–82, Berlin, Germany, December 1995. Springer-Verlag.
- [208] B. T. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In Rangachar Kasturi and Karl Tombre, editors, *Graphics Recognition - methods and applications*, volume 1072 of *Lecture Notes in Computer Science*, pages 123–34. Springer Verlag, May 1996.
- [209] R. Michalski and R. E. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:396–409, 1983. not held.
- [210] T. P. Minka and R. W. Picard. Interactive learning using a ‘society of models’. Technical Report 349, MIT Media Laboratory Perceptual Computing Section, MIT Media Laboratory, Cambridge, MA, 1996.
- [211] Marvin Minsky. A framework for representing knowledge. In Patrick H. Winston, editor, *The psychology of computer vision*. McGraw-Hill, New York, 1975.
- [212] Marvin Minsky and Seymour Papert. *Perceptrons. An introduction to computational geometry*. The MIT Press, Cambridge, London, 1969.

- [213] Toshiaki Miura. Behavior oriented vision: Functional field of view and processing resources. In J. K. O'Regan and A. Lévy-Schoen, editors, *Eye Movements: From Physiology to Cognition*, pages 563–572. North-Holland, 1987.
- [214] R. R. Mize and E. H. Murphy. Selective visual experience fails to modify receptive field properties of rabbit striate cortex neurons. *Science*, 180:320–321, 1970.
- [215] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-D structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1121–1139, 1989.
- [216] Rakesh Mohan and Ramakant Nevatia. Perceptual organisation for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):616–635, June, 1992.
- [217] Farzin Mokhtarian and Alan K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):789–805, August 1992.
- [218] Shunji Mori, Ching Y. Suen, and Kazuhiko Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE; Special Issue on Optical Character Recognition*, 80(7):1029–1058, July 1992.
- [219] Robin K. Morris, Keith Rayner, and Alexander Pollatsek. Eye movement guidance in reading: The role of parafoveal letter and space information. *Journal of Experimental Psychology: Human Perception and Performance*, 16(2):268–281, 1990.
- [220] S. Muggleton and Wray Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning; Ann Arbor, MI, USA*, pages 339–352, San Mateo, CA, USA, 1988. Morgan Kaufmann.
- [221] Stephen Muggleton. *Inductive Logic Programming*. Academic Press, 1992.
- [222] Stephen Muggleton. Stochastic logic programs. In *Inductive Logic Programming (ILP) '95 Workshop*, 1995.
- [223] Joe L. Mundy, R. Curwen, J. Liu, C. Rothwell, Andrew Zisserman, and D. Forsyth. MORSE: An architecture for 3D object recognition based on invariants. In *Second Asian Conference on Computer Vision*, volume II, pages 16–21. School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, December 1995.

- [224] Joe L. Mundy, C. Huang, J Liu, W. Hoffman, D. A. Forsyth, C. A. Rothwell, Andrew Zisserman, S. Utcke, and O. Bournes. MORSE: A 3D object recognition system based on geometric invariants. In *23rd Image Understanding Workshop*, pages 1393–1402, 1994.
- [225] Patrick M. Murphy and Michael J. Pazzani. Exploring the decision forest: An empirical investigation of Occam’s razor in decision tree induction. *Journal of Artificial Intelligence Research*, 1:257–75, 1994.
- [226] D. W. Murray, D. A. Castelow, and B. F. Buxton. From image sequences to recognized moving polyhedral objects. *International Journal of Computer Vision*, 3:181–208, 1989.
- [227] T. A. Nazir and J. K. O’Regan. Some results on translation invariance in the human visual system. *Spatial Vision*, 5:81–100, 1990.
- [228] Bernd Neumann. Natural language description of time-varying scenes. In David L. Waltz, editor, *Semantic Structures: advances in natural language processing*, pages 167–207. Lawrence Erlbaum, Hillsdale, N.J, 1989.
- [229] A. Newell and H. Simon. Computer science as empirical enquiry: Symbols and search. *Communications of the Association for Computing Machinery*, 19:113–126, 1976.
- [230] Nils J. Nilsson. *Problem-Solving methods in artificial intelligence*. McGraw-Hill Book Company, 1971.
- [231] David Noton and Lawrence Stark. Eye movements and visual perception. *Scientific American*, 224(6):34–43, 1971.
- [232] David Noton and Lawrence Stark. Scanpaths in eye movements during pattern perception. *Science*, 171(3968):308–311, 1971. Jan 22.
- [233] Mattias Ohlsson. Extensions and explorations of the elastic arms algorithm. Technical Report LU TP 92-28, Department of Theoretical Physics, University of Lund, Sölvegatan 14A, S-22362 Lund, Sweden, December 1992.
- [234] J. Oliver. Decision graphs - an extension of decision trees. Shortened version to appear in AI and Statistics 1993, December 1992.
- [235] J. Oliver and C.S. Wallace. Inferring decision graphs. Technical Report 91/170, Monash University, November 1992.
- [236] Mark Ollila and Terry Caelli. Networking of geometric concepts for algebraic metrology. In *ACCV95—Proceedings of the Asian Conference on Computer Vision*, Singapore, December 1995.

- [237] J. K. O'Regan and A. Lévy-Schoen. Integrating visual information from successive fixations: Does trans-saccadic fusion exist? *Vision Res.*, 23(8):765–768, 1983.
- [238] J. Kevin O'Regan. Eye movements and reading. In Eileen Kowler, editor, *Eye Movements and Their Role in Visual and Cognitive Processes*, pages 395–453. Elsevier, 1990.
- [239] J. Kevin O'Regan. Solving the “real” mysteries of visual perception: The world as an outside memory. *Canadian Journal of Psychology*, 46(3):461–488, 1992.
- [240] J. Kevin O'Regan. The world as an outside iconic memory—no strong internal metric means no problem of visual stability. *Behavioral and Brain Sciences*, 17:258–2259, 1994. Open Peer Commentary on Target Article: *A theory of visual stability across saccadic eye movements*, by Bruce Bridgeman, A.H.C. Van der Heijden and Boris M. Velichkovsky.
- [241] Nikhil R. Pal and Sankar K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.
- [242] M. Palhang and A. Sowmya. Learning object models from real images. In *Proceedings of The First International Conference on Visual Information Systems*, pages 335–44. Victoria Univ. Technol; Melbourne, Vic., Australia, 1996.
- [243] Deepak N. Pandya and Edward H. Yeterian. Architecture and connections of cortical association areas. In Alan Peters and Edward G. Jones, editors, *Cerebral Cortex*, volume 4, Association and Auditory Cortices, chapter 1, pages 3–61. Plenum Press: NY, 1985.
- [244] Y. Park. A comparison of neural net classifiers and linear tree classifiers: Their similarities and differences. *Pattern Recognition*, 27(11):1493–1503, 1994.
- [245] Theo Pavlidis and Yuh-Tay Liow. Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, March 1990.
- [246] Theo Pavlidis and Shunji Mori. Scanning the issue: Optical character recognition. *Proceedings of the IEEE; Special Issue on Optical Character Recognition*, 80(7):1027–8, July 1992.
- [247] M. J. Pazzani, C. A. Brunk, and G. Silverstein. A knowledge-intensive approach to learning relational concepts. In *Proceedings of the eighth international workshop on machine learning; Evanston, IL, USA*, pages 432–436, San Mateo, CA, 1991. Morgan Kaufmann.

- [248] Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine learning*, 9(1):57–94, 1992.
- [249] Adrian Pearce. *Relational Evidence Theory and Spatial Interpretation Procedures*. PhD thesis, School of Computing, 1996.
- [250] Adrian Pearce and Terry Caelli. On the efficiency of spatial learning. In *Proceedings of the second Asian conference on computer vision 1995 (ACCV95); Singapore*, pages 79–82, Parkville, Australia, December 1995.
- [251] Adrian R. Pearce, Terry Caelli, and Walter F. Bischof. Rulegraphs for graph matching in pattern recognition. *Pattern Recognition*, 27(9):1231–47, 1994.
- [252] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [253] Stavros J. Perantonis and Paulo J. G. Lisboa. Translation, rotation, and scale invariant pattern recognition by higher-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3(2):241–251, March 1992.
- [254] M. A. Peterson and B. Gibson. The initial identification of figure-ground relationship: Contributions from shape recognition processes. *Bell. Psychonomic Soc.*, 29:199–202, 1991.
- [255] R. W. Picard, T. P. Minka, and M. Szummer. Modeling user subjectivity in image libraries. Technical Report 382, MIT Media Laboratory Perceptual Computing Section, MIT Media Laboratory, Cambridge, MA, September 1996.
- [256] Steven Pinker. Visual cognition: An introduction. In Steven Pinker, editor, *Visual Cognition*, pages 1–63. A Bradford Book: The MIT Press, 1986. Reprinted from “Cognition: International Journal of Cognitive Psychology”, volume 18.
- [257] David A. Pintsov. Invariant pattern recognition, symmetry, and radon transforms. *Journal of the Optical Society of America (A)*, 6(10):1544–1554, October 1989.
- [258] Rejean Plamondon. Handwriting processing and recognition. *Pattern Recognition*, 26(3):379, March 1993. Editorial.
- [259] T. Poggio. Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497, 1990.
- [260] T. Poggio and S. Edelman. A network that learns to recognise 3D objects. *Nature*, 343:263–266, 18th January 1990.

- [261] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1981.
- [262] Alexander Pollatsek and Keith Rayner. Reading. In Michael I. Posner, editor, *Foundations of Cognitive Science*, chapter 10, pages 401–436. The MIT Press: A Bradford Book: Cambridge, MA, 1989.
- [263] Alexander Pollatsek, Keith Rayner, and William E. Collins. Integrating pictorial information across eye movements. *Journal of Experimental Psychology: General*, 113(3):426–442, 1984.
- [264] J. Porrill, S. B. Pollard, T. P. Pridmore, J. B. Bowen, J. E. W. Mayhew, and J. P. Frisby. TINA: a 3D vision system for pick and place. *Image and Vision Computing*, 6(2):91–99, May 1988.
- [265] Mary C. Potter. Representational buffers: The eye-mind hypothesis in picture perception, reading, and visual search. In Keith Rayner, editor, *Eye Movement in Reading: Perceptual and Language Processes*, pages 413–437. Academic Press, Inc., 1983.
- [266] David R. Pratt, Michael Zyda, and Kristen Kelleher. Virtual reality: In the mind of the beholder. *IEEE Computer*, pages 17–19, July 1995.
- [267] William K. Pratt. *Digital Image Processing*. John Wiley and Sons, 2nd edition, 1991.
- [268] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and monographs in computer science. Springer-Verlag, 1985.
- [269] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Press Syndicate of the University of Cambridge, Cambridge, U.K., 2nd edition, 1992.
- [270] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. Reproduced in *Readings in Machine Learning* [314], pages 57–69.
- [271] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
- [272] J. R. Quinlan and R. M. Cameron-Jones. FOIL: a midterm report. In P. B Brazdil, editor, *Machine Learning: ECML-93. European Conference on Machine Learning Proceedings; Vienna, Austria*, pages p. xii+469, 3–20, Berlin, Germany, April 1993. Springer-Verlag.

- [273] J. Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufman, 1993.
- [274] V. S. Ramachandran. Apparent motion of subjective surfaces. *Perception*, 14:127–134, 1985.
- [275] V. S. Ramachandran. Capture of stereopsis and apparent motion by illusory contours. *Percept. Psychophys.*, 39:361–373, 1986.
- [276] V. S. Ramachandran. Perception of depth from shading. *Scientific American*, 269:76–83, 1988.
- [277] V. S. Ramachandran. Perception: A biological perspective. In Gail A. Carpenter and Stephen Grossberg, editors, *Neural Networks for Vision and Image Processing*, pages 45–91. A Bradford Book: The MIT Press, 1992.
- [278] V. S. Ramachandran and S. M. Anstis. Perceptual organization in moving patterns. *Nature*, 304:529–531, 1983.
- [279] V. S. Ramachandran and S. M. Anstis. Perception of apparent motion. *Scientific American*, 254:102–109, 1986.
- [280] K. Rayner, G. W. McConkie, and S. Ehrlich. Eye movements and integrating information across fixations. *J. Exp. Psychol. Hum. Percept. Performance*, 4:529–544, 1978.
- [281] K. Rayner and A. Pollatsek. Is visual information integrated across saccades? *Percept. Psychophys.*, 34:39–48, 1983.
- [282] R. Reiter and A. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41:125–155, 1990.
- [283] R. A. Rescorla and A. R. Wagner. A theory of pavlovian conditioning: The effectiveness of reinforcement and non-reinforcement. In *Classical Conditioning II: Current Research and Theory*, pages 64–69. Appleton-Century-Crofts, New York, 1972.
- [284] Bradley Rhodes. Pronomes in behavior nets. Private correspondence, available at <http://rhodes.www.media.mit.edu/people/rhodes/pronomes.html>, 1995.
- [285] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw-Hill, New York, 2nd edition, 1991.
- [286] R. Rimey and C. Brown. Control of selective perception using Bayes' nets and decision theory. *International Journal of Computer Vision*, 12(2/3):173–207, 1994.

- [287] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Ann. Statist.*, 11:416–431, 1982.
- [288] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.
- [289] Jairo Rocha and Theo Pavlidis. A shape analysis model with applications to a character recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):393–404, April 1994.
- [290] K. S. Rockland. Laminar distribution of neurons projecting from area V1 to V2 in macaque and squirrel monkeys. *Cerebral Cortex*, 2:38–47, 1992.
- [291] K. S. Rockland, K. S. Saleem, and K. Tanaka. Widespread feedback connections from areas V4 and TEO. *Soc. Neurosci. Abst.*, 18:390, 1992.
- [292] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, June 1976.
- [293] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, Orlando, FL., 1982.
- [294] P. L. Rosin and G. A. W. West. Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–53, 1995.
- [295] Paul L. Rosin. Representing curves at their natural scales. *Pattern Recognition*, 25(11):1315–1325, 1992.
- [296] S. M. Ross and L. E. Ross. Saccade latency and warning signals: Effects of auditory and visual onset and offset. *Percept. Psychophys.*, 29:429–437, 1980.
- [297] Jacob Rubinstein, Joseph Segman, and Yehoshua Zeevi. Recognition of distorted patterns by invariance kernels. *Pattern Recognition*, 24(10):959–967, 1991.
- [298] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [299] S. L. Salas, Einar Hille, and John T. Anderson. *Calculus: One and several variables, with analytic geometry*. John Wiley & Sons, New York, fifth edition, 1986.

- [300] Warren S. Sarle. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, April 1994.
- [301] Roger Schank and R. P. Abelson. *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale, N.J, 1977.
- [302] J. R. J. Schirra, G. Bosch, C. K. Sung, and G. Zimmermann. From image sequences to natural language: a first step toward automatic perception and description of motions. *Applied Artificial Intelligence*, 1(4):287–305, 87.
- [303] J. Schlimmer and D. Fisher. A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 496–501, 1986.
- [304] J. C. Scholtes. Recurrent Kohonen self-organization in natural language processing. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks - Volume 2 (Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91), Espoo, Finland, (June 1991)*, pages 1751–1754. North-Holland, 1991.
- [305] Robert Sedgewick. *Algorithms in C*. Addison-Wesley Pub. Co., Reading Mass. USA, 1990.
- [306] Joseph Segman, Jacob Rubinstein, and Yehoshua Y. Zeevi. The canonical coordinates method for pattern deformation: Theoretical and computational considerations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(12):1171–1183, December 1992.
- [307] Robert Sekuler and Randolph Blake. *Perception (Second Edition)*. McGraw-Hill Publishing Company, 1990.
- [308] Kuntal Sengupta and Kim L. Boyer. Organizing large structural modelbases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):321–332, April 1995.
- [309] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [310] E. Shapiro and A. Takeuchi. Object oriented programming in concurrent prolog. In E. Shapiro, editor, *Concurrent Prolog*, volume 2, chapter 29, pages 251–273. MIT Press, 1987.
- [311] Linda G. Shapiro and Robert M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504–519, September 1981.

- [312] Linda G. Shapiro and Robert M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):90 – 94, January 1985.
- [313] Noel E. Sharkey and Amanda J. C. Sharkey. Separating learning and representation. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 17–32. Springer, 1996.
- [314] J. W. Shavlik and T. G. Dietterich, editors. *Readings in Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1990.
- [315] W. E. Simpson and S. J. Crandall. The perception of smiles. *Psychonomic Sci.*, 29:197–200, 1972.
- [316] Peter Slezak. Situated cognition: Empirical issue, paradigm shift or conceptual confusion. In Ashwin Ram and Kurt Eiselt, editors, *Proceedings of the Sixteenth Conference of the Cognitive Science Society*, pages 806–811, Atlanta, August 1994. Lawrence Erlbaum, Hillsdale, New Jersey.
- [317] Padhraic Smyth, David Heckerman, and Michael I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9:227–269, 1997.
- [318] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Chapman and Hall, 1993.
- [319] Eduardo D. Sontag. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, 3:981–990, 1992.
- [320] Lilly Spirkovska and Max B. Reid. Higher-order neural networks applied to 2D and 3D object recognition. *Machine Learning*, 15(2):169–199, 1994.
- [321] David McG. Squire. *Model-based Neural Networks for Invariant Pattern Recognition*. PhD thesis, School of Computing, Curtin University of Technology, Perth, Western Australia, October 1996.
- [322] David McG. Squire and Terry M. Caelli. Shift, rotation and scale invariant signatures for two-dimensional contours, in a neural network architecture. In Stephen W. Ellacott, John C. Mason, and Iain J. Anderson, editors, *Mathematics of Neural Networks: Models Algorithms and Applications*, Statistics and OR, pages 343–349, Boston, July 1995. Kluwer Academic Publishers.
- [323] V. Srinivasan, P. Bhatia, and S. H. Ong. Edge detection using A neural network. *Pattern Recognition*, 27(12):1653–1662, 1994.

- [324] Lawrence W. Stark and Yun S. Choi. Experimental metaphysics: The scanpath as an epistemological mechanism. In *Visual Attention and Cognition*, pages 3–69. Elsevier, 1996.
- [325] Bradley S. Stewart, Ching-Fang Liaw, and Chelsea C. White. A bibliography of heuristic search research through 1992. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(2):268–293, February 1994.
- [326] G. M. Stratton. Upright vision and the retinal image. *Psychol. Rev.*, 4:182–187, 1897.
- [327] G. M. Stratton. Vision without inversion of the retinal image. *Psychol. Rev.*, 4:341–360, 463–481, 1897.
- [328] S. Swain and M. Stricker. Promising directions in active vision. *International Journal of Computer Vision*, 11(2):109–126, 1993.
- [329] Leonard Talmy. How language structures space. In Pick Jr, Herbert L. and Linda P. Acredolo, editor, *Conference on Spatial Orientation—Theory, Research, and Application*, pages 225–282. Plenum Press, NY, 1983.
- [330] K. Tanaka, H.-A. Saito, Y. Fukada, and M. Moriya. Coding visual images of objects in the inferotemporal cortex of the macaque monkey. *J. Neurophysiol.*, 66:170–189, 1991.
- [331] K. Tanaka and H.A. Saito. Analysis of motion of the visual field by direction, expansion/contraction, and rotation cells clustered in the dorsal part of the medial superior temporal area of the macaque monkey. *J. Neurophysiol.*, 62:626–641, 1989.
- [332] R. E. Tarjan and A. E. Trojanowski. Finding a maximum independent set. *SIAM Journal of Computing*, 6(3):537–546, 1977.
- [333] Robert E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the Association for Computing Machinery*, 22(2):215–225, April 1975.
- [334] Robert E. Tarjan and Jan Van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the Association for Computing Machinery*, 31(2):245–281, April 1984.
- [335] Michael Reed Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, August 1980.
- [336] P. Thompson. Margaret Thatcher: A new illusion. *Perception*, 9:483–484, 1980.

- [337] A. M. Treisman and G. Gelade. A feature integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.
- [338] Yves Trotter, Simona Celebrini, Brigitte Stricanne, Simon Thorpe, and Michel Imbert. Modulation of neural stereoscopic processing in primate area V1 by viewing distance. *Science*, 257:1279–1281, 1992.
- [339] John K. Tsotsos. The complexity of perceptual search tasks. In N. S. Sridharan, editor, *Proc. International Joint Conference on Artificial Intelligence*, pages 1571–1577, Detroit, August 1989. Morgan Kaufman.
- [340] M. T. Turvey. Contrasting orientations to the theory of visual information processing. *Psychological Review*, 84:67–88, 1977.
- [341] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, January 1976.
- [342] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2):161–186, 1989.
- [343] D. Van Essen and J. H. R. Maunsell. Hierarchical organization and functional streams in the visual cortex. *Trends Neurosci.*, 6:370–375, 1983.
- [344] David C. Van Essen and Charles H. Anderson. Information processing strategies and pathways in the primate retina and visual cortex. In S. F. Zornetzer, J. L. Davis, and C. Lau, editors, *Introduction to Neural and Electronic Networks*, pages 43–72. Academic Press, Orlando, FL, 1989.
- [345] David C. Van Essen and Edgar A. DeYoe. Concurrent processing in the primate visual cortex. In Michael S. Gazzaniga, editor, *The Cognitive Neurosciences*, pages 383–400. A Bradford Book: The MIT Press: Cambridge, MA, 1995.
- [346] Tim van Gelder and Robert Port. It's about time: An overview of the dynamical approach to cognition. In Robert Port and Tim van Gelder, editors, *Mind as motion: explorations in the dynamics of cognition*, pages 1–41. MIT Press, Cambridge MA, 1995.
- [347] Paolo Viviani. Eye movements in visual search: cognitive, perceptual and motor control aspects. In Eileen Kowler, editor, *Eye Movements and Their Role in Visual and Cognitive Processes*, pages 353–393. Elsevier, 1990.
- [348] R. von der Heydt, E. Peterhans, and G. Baumgartner. Illusory contours and cortical neuron responses. *Science*, 224:1260–1262, 1984.

- [349] H. von Helmholtz. *Physiological optics*, volume 3 (J.P.C. Southall, Trans.). Rochester, NY: Optical Society of America., 1925. (Original work published 1909).
- [350] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.
- [351] G. J. Walker-Smith, A. G. Gale, and J. M. Findlay. Eye movement strategies involved in face perception. *Perception*, 6:313–326, 1977.
- [352] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *The Journal of the Royal Statistical Society, Series B. Methodology*, 49(3):223–265, 1987.
- [353] C. S. Wallace and J. D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.
- [354] S. Watanabe. Pattern recognition as a quest for minimum entropy. *PR*, 13(5):381–387, 1981.
- [355] Harry Wechsler. *Computational Vision*. Academic Press Inc., 1250 Sixth Avenue, San Diego, CA 92101, 1990.
- [356] T. G. Weyland and J. G. Malpeli. Responses of neurons in primary visual cortex are influenced by eye position. *Soc. Neurosci. Abstr.*, 15:1016, 1989.
- [357] S. D. Whitehead and D. Ballard. Active perception and reinforcement learning. *Neural Comp.*, 2:409–419, 1990.
- [358] S. D. Whitehead and D. Ballard. Connectionist designs on planning. In *Neural Information Processing Systems 3*, pages 357–370. Morgan Kaufmann, San Mateo, CA, 1991.
- [359] Benjamin Lee Whorf. Languages and logic. In John B. Carroll, editor, *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*, pages 233–245. The MIT Press, Cambridge, MA, 1971. Reprinted from *Technol. Rev.*, 43:250–252, 266, 268, 272, 1941.
- [360] Janet Wiles and Jeff Elman. Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Society*, Cambridge, MA, 1995. MIT Press.
- [361] Janet Wiles and Mark Ollila. Intersecting regions: The key to combinatorial structure in hidden unit space. *Advances in Neural Information Processing Systems*, 5:27–33, 1993.

- [362] Hugh R. Wilson and Douglas J. Gelb. Modified line-element theory for spatial-frequency and width discrimination. *J. Opt. Soc. Am. A*, 1(1):124–131, 1984.
- [363] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The psychology of computer vision*. New York: McGraw Hill, 1975.
- [364] Patrick Henry Winston. *Artificial Intelligence*. Addison Wesley, 1984.
- [365] L. Wixson and D. Ballard. Using intermediate objects to improve the efficiency of visual search. *International Journal of Computer Vision*, 12(2/3):209–230, 1994.
- [366] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3):287–303, 1992.
- [367] Michael Wooldridge and Nicholas R. Jennings. *Intelligent Agents: Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in AI*. Springer-Verlag, January 1995.
- [368] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.
- [369] Stefan Wrobel. Concept formation during interactive theory revision. *Machine Learning*, 14:169–91, 1994.
- [370] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(8):841–847, August 1991.
- [371] A. L. Yarbus. *Eye Movements and Vision*. New York: Plenum Press, 1967.
- [372] Robert K. Yin. Looking at upside-down faces. *Journal of Experimental Psychology*, 81(1):141–145, 1969.
- [373] P. Yip and K. R. Rao. A fast computational algorithm for the discrete sine transform. *IEEE Transactions on Communications*, COM-28(2):304–307, 1980.
- [374] P. Yip and K. R. Rao. On the computation and the effectiveness of discrete sine transform. *Computer and Electrical Engineering*, 7:45–55, 1980.
- [375] Akinori Yonezawa and Carl Hewitt. Modelling distributed systems. In *IJCAI77*, pages 370–376, Massachusetts, August 1977. Kaufman.

- [376] S. Zeki. The distribution of wavelength and orientation selective cells in different areas of monkey visual cortex. *Proc. R. Soc. Lond. [Biol.]*, 217:449–470, 1983.
- [377] S. Zeki and S. Shipp. The functional logic of cortical connections. *Nature*, 335:311–317, 1988.
- [378] Semir Zeki. *A Vision of the Brain*. Oxford: Blackwell Scientific Publications), 1993.
- [379] Christopher Zetzsche and Terry Caelli. Invariant pattern recognition using multiple filter image representations. *Computer Vision, Graphics and Image Processing*, 45:251–262, 1989.
- [380] B. L. Zuber, L. Stark, and G. Cook. Microsaccades and the velocity amplitude relationship for saccadic eye movement. *Science*, 15:1459–1460, 1965.

# Author Index

- Abelson, R. P. , 227  
Abu-Mostafa, Y. S. , 31  
Adams, B. D. , 328  
Aggarwal, J. K. , 26  
Ahmadi, M. , 31  
Ahmed, H. , 36  
Allen, J. , 231  
Allen, J. S. , 345  
Allison, L. , 215, 219  
Aloimonos, Y. , 192, 193  
Altmann, J. , 257  
Amari, S. , 7  
Amin, A. , 73, 101  
Andersen, S. K. , 226  
Anderson, C. H. , 311, 340  
Andre, E. , 237  
Anstis, S. M. , 327, 338  
Arbib, M. A. , 348  
  
Baber, R. N. , 333  
Bajcsy, R. , 342  
Baldwin, J. F. , 225, 229, 234  
Ballard, D. , 25, 193, 195, 197,  
    257, 258, 342, 343, 374  
Ballard, D. H. , 325  
Barnard, E. , 305  
Barrow, H. G. , 85  
Bartley, N. R. , 251  
Battaglini, P. P. , 368  
Baumgartner, G. , 340  
Belkasim, S. O. , 31  
Bell, B. , 226, 228  
Beni, G. , 41  
Bergevin, R. , 19  
  
Bezdek, J. C. , 41  
Bhanu, B. , 23  
Bhatia, P. , 282  
Biederman, I. , 9, 19, 20, 193, 325  
Binford, T. , 193  
Bischof, W. F. , 9, 10, 18, 40–43,  
    45, 62, 72, 73, 75, 78, 85,  
    117  
Blake, A. , 342  
Blake, R. , 310, 372  
Blakemore, C. , 329, 369  
Bolles, R. C. , 72, 197  
Borgefors, G. , 207  
Bosch, G. , 237  
Bose, C. B. , 117  
Botha, E. C. , 305  
Bournes, O. , 21  
Bowen, J. B. , 22  
Bowyer, K. W. , 23  
Boyce, J. F. , 26  
Boyle, R. , 207  
Brandt, R. D. , 257  
Brandt, T. , 341  
Bratko, I. , 73  
Bridgeman, B. , 314, 330, 331  
Briscoe, G. , 5, 349, 351, 352, 359–  
    361, 364, 366, 374  
Brolio, J. , 2, 120, 227  
Brown, C. , 193, 195, 197  
Brown, C. M. , 25, 257, 258  
Brunelli, R. , 51  
Brunk, C. A. , 73, 97, 116  
Bruton, L. T. , 251  
Bulsari, A. , 277

- Bunke, H. , 52, 73, 101, 117, 148  
 Buntine, W. , 98  
 Burchfield, J. L. , 315  
 Burns, J. B. , 31, 37  
 Burstall, R. M. , 85  
 Buxton, B. F. , 197
- Caelli, T. M. , 2, 4, 5, 8–10, 14,  
     18, 22, 26, 28, 40–43, 45,  
     62, 70, 72, 73, 75, 78, 85,  
     117, 148, 229, 239, 255,  
     257, 258, 260, 266, 273,  
     281, 282
- Cameron-Jones, R. M. , 72, 73, 98  
 Campbell, F. W. , 320  
 Canny, J. F. , 4, 27  
 Carevic, D. , 4  
 Carmody, D. , 315  
 Castelow, D. A. , 197  
 Celebrini, S. , 368  
 Chang, R. L. P. , 45  
 Cheeseman, P. , 6  
 Chen, Y-S. , 296  
 Choi, Y. S. , 345  
 Chomsky, N. , 374  
 Chow, G. , 258  
 Chu, C. , 26  
 Churchland, P. S. , 327, 336, 339,  
     341, 342, 373, 374  
 Clancey, W. J. , 238  
 Cleeremans, A. , 349  
 Clement, V. , 192  
 Colby, C. , 314  
 Cole, J. B. , 260  
 Collins, T. , 2, 120, 227  
 Collins, W. E. , 328, 334  
 Conlon, T. , 229  
 Cook, G. , 331  
 Cooper, G. F. , 329, 369  
 Coronado, J. L. , 101  
 Cosman, P. C. , 7  
 Crandall, S. J. , 325  
 Culhane, S. , 197  
 Curwen, R. , 21
- Dance, S. , 2, 229, 239  
 Davis, L. , 161  
 Davison, A. , 230  
 De Groot, A. D. , 315  
 De Vel, O. , 28, 43  
 Dempster, A. P. , 234  
 DePriest, D. D. , 341  
 DeYoe, E. A. , 311–313  
 Dichgangs, J. , 341  
 Dickinson, S. J. , 19  
 Dillencourt, M. , 155  
 Dillon, C. , 323  
 Dimitriadis, Y. A. , 101  
 Dix, T. I. , 215, 219  
 Dodwell, P. , 266  
 Dowe, D. L. , 215, 219  
 Draper, B. , 2, 23, 120, 227  
 Dreier, A. , 8, 22, 148  
 Dubes, R. C. , 6  
 Dubois, D. , 236  
 Duda, R. O. , 7  
 Duffy, F. H. , 315
- Edelman, S. , 20, 21  
 Ehrlich, S. , 328, 335  
 Elman, J. , 290, 346, 349  
 Elschlager, R. A. , 72  
 Eriksen, B. A. , 325  
 Eriksen, C. W. , 325  
 Erlich, S. F. , 335  
 Esterline, A. , 193  
 Esterline, A. , 193
- Fahlman, S. E. , 7, 282  
 Fan, T. , 72  
 Fayyad, U. , 7  
 Feldman, J. A. , 325  
 Feller, W. , 235  
 Ferraro, M. , 257, 260, 273  
 Findlay, J. M. , 344  
 Fischler, M. A. , 72  
 Fisher, D. , 7, 143  
 Flannery, B. P. , 28, 292  
 Flynn, P. J. , 71, 73, 197  
 Flynn, P. J. , 132

- Fodor, J. A. , 374  
Forsyth, D. A. , 21, 259  
Freeman, D. , 6  
Freeman, P. R. , 65  
Friedlander, M. J. , 368  
Frisby, J. P. , 22  
Fukada, Y. , 314  
Fuster, J. M. , 341, 360, 368  
  
Gale, A. G. , 344  
Galletti, C. , 368  
Garvey, T. , 234  
Gelade, G. , 371  
Gelb, D. J. , 352, 361  
Gibson, B. , 340  
Gibson, E. J. , 324, 329  
Gibson, J. , 256, 333  
Giefing, G. J. , 342  
Glass, A. L. , 325  
Gonzalez, R. C. , 32, 36, 51  
Goossens, M. , 69, 106  
Graham, R. , 33  
Gray, R. M. , 7  
Grimson, W. E. L. , 71, 73  
Groner, M. , 344  
Groner, R. , 344, 345  
Grosof, D. H. , 340  
  
Hacisalihzade, S. S. , 345  
Haddon, J. F. , 26  
Hall, L. O. , 23  
Hallet, P. E. , 328  
Hanazawa, T. , 346  
Hanson, A. , 2, 120, 227  
Haralick, R. M. , 70  
Harrison, M. C. , 234  
Hart, P. E. , 7  
Hawken, M. J. , 340  
Hebb, D. O. , 330, 333  
Hebert, M. , 192  
Henkind, S. J. , 234  
Henton, W. W. , 374  
Herzog, G. , 237  
Hewitt, C. , 227, 229  
Hinton, G. , 346  
  
Hinton, G. E. , 281, 346  
Hlavac, V. , 207  
Hochberg, J. , 333  
Hoffman, R. , 8, 10, 22  
Hoffman, W. C. , 21, 260, 266  
Hopcraft, J. E. , 148  
Horaud, P. , 72, 197  
Howard, I. P. , 314, 329, 331  
Howell, E. R. , 258  
Hsu, W-H. , 296  
Hu, M. K. , 31  
Huang, C. , 21  
Huang, T. T. , 226  
Hubel, D. , 266, 313  
Hummel, R. A. , 160  
Hwang, V. , 2, 120, 193, 226, 228  
  
Ikeuchi, K. , 71, 192  
Imbert, M. , 368  
Intraub, H. , 325  
Irani, K. , 7  
Irwin, D. E. , 333  
Iverson, I. H. , 374  
  
Jain, A. , 132  
Jain, A. K. , 6, 8, 10, 22, 35, 71,  
73, 197, 257  
Jain, R. , 193  
Jansen, H. , 342  
Jennings, N. R. , 228, 238  
Jensen, Finn. V. , 226  
Jensen, Frank. , 226  
Jonides, J. , 333  
Jordan, M. I. , 346  
  
Kak, A. C. , 148, 149, 257  
Kanade, T. , 71  
Kandel, E. R. , 359, 374  
Kangas, J. , 346  
Kant, I. , 194  
Kapur, D. , 14  
Karp, R. M. , 148  
Kelleher, K. , 102  
Keller, J. M. , 42  
Kelly, J. , 6

- Khotanzad, A. , 257  
 Kibler, D. , 73, 97, 116  
 Kim, W. , 148, 149  
 Knight, K. , 95, 102, 227  
 Kobler, J. , 211  
 Koenig, E. , 341  
 Koller, D. , 226  
 Kosko, B. , 237  
 Krishnapuram, R. , 42  
 Kuffler, S. W. , 370  
 Kundel, H. L. , 315  
 Kuo, S-S. , 117  
 Kurta , 102  
 Lakoff, G. , 228  
 Lal, R. , 368  
 Lang, K. J. , 346  
 Langley, P. , 23  
 Lebiere, C. , 7  
 Lee, C-M. , 193  
 Lee, S-W. , 101  
 Lenz, R. , 257  
 Levine, M. D. , 19  
 Levy-Schoen, A. , 333, 336  
 Li, M. , 78  
 Li, S. Z. , 258  
 Li, X. , 258  
 Liaw, C-F. , 73  
 Lin, F. , 257  
 Liow, Y-T. , 26  
 Lisboa, P. J. G. , 276, 285  
 Liu, J. , 21  
 Liu, Z-Q. , 2, 229, 239, 257  
 Livingstone, M. , 313  
 Locher, P. J. , 345  
 Lowe, D. G. , 28, 72, 102  
 Lowrance, J. D. , 234  
 Lu, J-H. , 257  
 Lu, S-W. , 101  
 MacKay, D. M. , 333, 334  
 Mackintosh, N. J. , 374  
 Mackworth, A. , 2, 103, 118  
 Mackworth, N. H. , 343  
 Maginu, K. , 7  
 Maitra, S. , 31, 32  
 Malik, J. , 226  
 Mallot, H. , 342  
 Malpeli, J. G. , 368  
 Marr, D. , 321  
 Masini, G. , 101  
 Matsuyama, T. , 2, 120, 193, 226,  
     228  
 Maunsell, J. H. , 340, 341  
 Mayhew, J. E. W. , 22  
 McCane, B. J. , 26, 28, 43, 53, 54,  
     73  
 McConkie, G. , 328, 335  
 Medioni, G. , 72  
 Menz, C. , 345  
 Messelodi, S. , 51  
 Messmer, B. T. , 52, 73, 101, 117,  
     148  
 Michalski, R. , 72  
 Minka, T. P. , 2  
 Minsky, M. , 227, 229, 287, 320,  
     368  
 Mittelbach, F. , 69, 106  
 Miura, T. , 331  
 Mize, R. R. , 329  
 Mohan, R. , 45, 72  
 Mohr, R. , 101  
 Mokhtarian, F. , 103, 118  
 Morandi, A. J. , 343  
 Mori, S. , 68, 72  
 Moriya, M. , 314  
 Morris, R. K. , 335  
 Muggleton, S. , 72, 73, 97, 98, 116  
 Mundy, J. L. , 14, 21, 259  
 Murase, H. , 260  
 Murphy, E. H. , 329  
 Murphy, P. M. , 73  
 Murray, D. W. , 197  
 Naito, S. , 260  
 Natarajan, T. , 36  
 Nazir, T. A. , 330, 370  
 Nealey, T. A. , 341  
 Neumann, B. , 226, 228, 237

- Nevatia, R. , 45, 72  
Newell, A. , 237  
Nilsson, N. J. , 95, 102  
Nishihara, H. K. , 321  
Nodine, C. F. , 315, 345  
Noton, D. , 343, 344, 351  
  
O'Regan, J. K. , 330-336, 370  
Oehler, K. L. , 7  
Ogasawara, G. , 226  
Ohlsson, M. , 257  
Olesen, K. G. , 226  
Oliver, J. , 215, 219  
Ollila, M. , 14, 290  
Ong, S. H. , 282  
  
Pal, N. R. , 70, 98  
Pal, S. K. , 70, 98  
Palhang, M. , 73  
Pandya, D. N. , 360  
Papert, S. , 287, 368  
Park, Y. , 7  
Patrick, J. D. , 78  
Pau, L. F. , 226, 228  
Pavlidis, T. , 26, 45, 68, 101, 102  
Pazzani, M. J. , 73, 97, 116  
Pearce, A. R. , 70, 72, 78, 85, 320  
Pearl, J. , 70  
Pentland, A. P. , 19  
Perantonis, S. J. , 276, 285  
Peterhans, E. , 340  
Peterson, M. A. , 340  
Picard, R. W. , 2  
Pinker, S. , 320  
Pintsov, D. A. , 257, 258  
Plamondon, R. , 68, 72  
Poggio, T. , 6, 7, 20  
Pollard, S. B. , 22  
Pollatsek, A. , 328, 333-335  
Pong, T-C. , 193  
Porrill, J. , 22  
Port, R. , 237  
Potter, M. C. , 330  
Prade, H. , 236  
Pratt, D. R. , 102  
  
Pratt, W. K. , 36  
Preparata, F. P. , 33  
Press, W. H. , 28, 292  
Preston, G. A. N. , 258  
Pridmore, T. P. , 22  
Psaltis, D. , 31  
  
Quinlan, J. R. , 8, 10, 12, 70, 72-  
74, 97, 98, 101, 106, 141,  
144, 198, 213, 214  
  
Rabinowitz, J. C. , 325  
Ramachandran, V. S. , 323, 327,  
330, 336, 338-342, 373,  
374  
Rao, B. , 226  
Rao, K. R. , 36  
Rayner, K. , 328, 333-335  
Reid, M. B. , 276, 285  
Reitbock, H. , 257  
Reiter, R. , 2  
Ren, Y. , 101  
Rescorla, R. A. , 374  
Rhodes, B. , 237  
Rich, E. , 95, 102, 227  
Rimey, R. , 193, 197  
Riseman, E. , 2, 31, 37, 120, 227  
Riskin, E. A. , 7  
Rissanen, J. , 73  
Rist, T. , 237  
Robinson, J. A. , 229  
Robson, J. G. , 320  
Rocha, J. , 101, 102  
Rockland, K. S. , 340, 341  
Roeder, N. , 258  
Rosenbaum, D. A. , 346  
Rosenfeld, A. , 19, 160, 161, 257  
Rosin, P. L. , 102, 103, 118  
Ross, L. E. , 328  
Ross, S. M. , 328  
Rothwell, C. A. , 21  
Rubinstein, J. , 257, 260  
Rumelhart, D. E. , 281  
Ruspini, E. H. , 234  
Russell, S. , 226

- Saito, H-A. , 314  
Saleem, K. S. , 340, 341  
Samarin, A. , 69, 106  
Samet, H. , 155  
Sammut, C. , 73, 101  
Sarle, W. S. , 282  
Schank, R. , 227  
Schirra, J. R. J. , 237  
Schlimmer, J. , 143  
Scholtes, J. C. , 346  
Schonning, U. , 211  
Schwartz, J. H. , 359, 374  
Sclar, G. , 341  
Sedgewick, R. , 100  
Segman, J. , 257, 260  
Sejnowski, T. J. , 327, 336, 339,  
    341, 342, 373, 374  
Sekuler, R. , 310, 372  
Self, M. , 6  
Shafer, G. , 234  
Shamos, M. I. , 33  
Shapiro, E. , 229  
Shapiro, L. G. , 70  
Shapley, R. M. , 340  
Sharkey, A. J. C. , 349  
Sharkey, N. E. , 349  
Shikano, K. , 346  
Shipp, S. , 314  
Shridhar, M. , 31  
Silverstein, G. , 73, 97, 116  
Simon, H. , 237  
Simpson, W. E. , 325  
Slagle, J. , 193  
Slezak, P. , 238  
Sonka, M. , 207  
Sontag, E. D. , 282  
Sowmya, A. , 73  
Spirkovska, L. , 276, 285  
Squire, D. , 255, 260, 281, 282  
Srinivasan, V. , 282  
Stacy, E. W. , 325  
Stark, L. , 331, 343–345, 351  
Stepp, R. E. , 72  
Stewart, B. S. , 73  
Strat, T. M. , 234  
Stratton, G. M. , 329  
Stricanne, B. , 368  
Stricker, M. , 193  
Stutz, J. , 6  
Suen, C. Y. , 68, 72, 101  
Sum, K. C. , 73, 101  
Sung, C. K. , 237  
Swain, S. , 193  
Szummer, M. , 2  
Takeuchi, A. , 229  
Talmy, L. , 336  
Tamminen, M. , 155  
Tanaka, K. , 314, 340, 341  
Tarjan, R. E. , 100, 148  
Taylor, W. , 6  
Teague, M. R. , 31  
Teukolsky, S. A. , 28, 292  
Thomas, D. , 234  
Thompson, P. , 330  
Thonnat, M. , 192  
Thorpe, S. , 368  
Toran, J. , 211  
Treisman, A. M. , 371  
Trojanowski, A. E. , 100, 148  
Trotter, Y. , 368  
Tsotsos, J. , 197, 231  
Turvey, M. T. , 333  
Ullmann, J. R. , 85  
Utcke, S. , 21  
Utgoff, P. E. , 143, 144, 215  
Van der Heijden, A. H. C. , 314,  
    330, 331  
Van Essen, D. , 311–313, 340  
Van Gelder, T. , 237  
Van Leeuwen, J. , 100  
Velichkovsky, B. M. , 314, 330,  
    331  
Vetterling, W. T. , 28, 292  
Vitanyi, P. M. B. , 78  
Viviani, P. , 315, 346  
Von der Heydt, R. , 340

- Von Helmholtz, H. , 333  
Wagner, A. R. , 374  
Waibel, A. , 346  
Walder, F. , 344  
Walker-Smith, G. J. , 344  
Wallace, C. S. , 65, 78, 215, 219  
Watanabe, S. , 198, 213  
Weber, J. , 226  
Wechsler, H. , 256, 257  
Weinshall, D. , 21  
Weiss, R. S. , 31, 37  
Wesley L. P , 234  
West, G. A. W. , 102  
Weyland, T. G. , 368  
White, C. C. , 73  
Whitehead, S. D. , 374  
Whorf, B. L. , 336  
Wiesel, T. , 266  
Wild, T. P. J. , 255, 282  
Wiles, J. , 290  
Wilkins, D. E. , 234  
Williams, R. J. , 281  
Wilson, H. R. , 352, 361  
Winston, P. H. , 95, 102, 320  
Wixson, L. , 197  
Wong, E. K. , 85  
Woods, R. E. , 32, 36, 51  
Wooldridge, M. , 228, 238  
Wrobel, S. , 73, 97, 116  
  
Xie, X. L. , 41  
  
Yamamoto, K. , 68, 72  
Yantis, S. , 333  
Yarbus, A. L. , 343  
Yeterian, E. H. , 360  
Yin, R. K. , 330  
Yip, P. , 36  
Yonezawa, A. , 227  
Yuille, A. , 342  
  
Zeevi, Y. Y. , 257, 260  
Zeki, S. , 314  
Zetzsche, C. , 257

# Index

- A\* search, 78, 95  
ABC  
    model, 337  
    temporal paradigm, 346  
    vision model, 358  
Abekas Digital Video system, 243  
active vision, 193, 324, 342  
actors, 229  
affine transformation, 242  
agent orientation, 228  
agent-based, 225, 228  
algebraic rules, 5  
ambiguous characters, 286, 303,  
    305, 307  
arboricity, 31, 33  
associative layer, 360  
attribute indexing, 9, 70, 106  
attribute learning, 79, 82, 95  
back-projection, 101, 131, 340  
background subtraction, 243  
Bayes' rule, 218, 234  
Bayesian belief network, 92, 226  
Bayesian conditioning, 97  
belief measures, 234  
belief pairs, 236  
best-first, 89, 95  
bidirectional link, 349  
binary  
    feature, 70, 73, 122, 138, 171  
    hypothesis, 152  
    learning, 147  
    query, 211  
binning, neural module, 283, 285,  
    306  
biological vision, 15  
bipartite matching, 148  
bottom-up, 197  
boundary region, 171  
C language, 242  
Canny edge detector, 205  
car velocities, 241  
character recognition, with ISNNC,  
    *see* ISNNC for optical char-  
    acter recognition  
CITE, 12, 120  
CLARET, 12, 68, 78, 95, 139  
clique, 127  
clique membership, 156  
clique resolution, 147, 156, 179  
closed-loop, 13  
cluster, 23, 28, 40–45, 51–53, 56,  
    57, 65  
    refinement, 42, 56  
    searching, 56, 57  
    splitting, 42, 43, 56, 57  
    validity, 41, 42  
clustering, 6, 13, 74, 83, 145, 168,  
    205, 213  
cognition, 237  
collision, 245  
colour space segmentation, 205  
concept-centred, 326  
concept-frame, 230, 233, 241  
concept-instance, 230, 242  
concurrency, 228, 252  
conditional feature space, 88  
conditional probability, 76

- conditional rule generation, 10, 72, 73, 88, 97  
cones, 310, 330, 367  
connected components, 73, 154, 207  
connection criterion, 76, 85  
constraint interpretation, 68  
contextual influence, 337  
continuous agents, 251  
cortex  
  cerebral, 359  
  mammalian, 340  
  striate, 310, 333  
  sub, 328  
  visual, 314  
CRG, 10, 17–19, 31, 40, 41, 44, 46, 51–53, 72, 73, 88, 97  
cross-modal, 338, 341  
data dimensionality, reduction of, 255, 260, 287, 289, 307  
decision boundaries, 213  
decision graph, 215, 218  
decision tree, 7, 68, 70, 78, 144, 214  
declarative knowledge, 196  
deduction, 325  
deductive learning, 5  
Dempster-Shafer, 229, 234, 235, 252  
depth-first, 89, 97  
directed binary feature, 171  
discrimination, 196  
durational agents, 251  
dynamic image sequences, 250  
dynamic programming, 78, 95, 110  
dynamic weights, 276, 277, 281, 282, 284  
edge extraction, 168  
English language interpretations, 246  
entropy, 78, 145, 214  
evidence chain, 75  
evidence-based, 68, 72  
evidential reasoning, 252  
evolutionary, 341  
exemplar, 213  
explanation-based learning, 5, 9  
explanatory least generalisation, 141, 142  
external world filter, 367  
exteroceptive sensory, 359  
FCRG, 10, 17–19, 40, 41, 43, 44, 50–53, 56, 58, 62, 65  
feature, 70, 357  
feature detectors, 318  
feature extraction, 3, 137, 169  
feature model, 318  
feature ring, 343  
feature space partitioning, 7  
feed-forward segmentation, 122  
feedback, 122, 131, 190, 199, 212, 359, 360  
figure ground, 23, 26, 340  
finite interpretation, 94  
fixation, 315, 330, 331, 343, 352  
FOIL, 106  
Fourier, 31, 34, 35, 53  
Fourier transform, 207  
fovea, 310, 330, 370  
frames of discernment, 236  
frequency domain, 207  
fuzzy cognitive maps, 237  
fuzzy conditional rule generation, 10  
ganglion, 310, 328  
Gaussian function, 207  
generalisation, 7  
geometric reasoning, 121  
geometric scene description, 226, 237  
geons, 19, 20  
give-way concept-frames, 242  
give-way relationship, 239  
global blackboard, 120  
global contour, 372  
graph matching, 73, 78, 79, 84, 95

- graph model, 73
- Hadamard, 31, 36, 37, 53, 54
- hand-drawn schematics, 12
- handwritten text, 100
- Hebbian learning, 346, 366
- hidden Markov model, 116
- hierarchical knowledge, 125
- hierarchical modelling, 92
- hierarchical processing, 322, 328
- hierarchical segmentation, 122, 127
- high-level analysis, 251
- higher order neural networks, 276, 285
- Horn clauses, 5
- Hu, 31, 54
- human vision, 15
- hypothesis generation, 151
- IGLUE, 215
- illegal turn, 248
- illusory contour, 340, 372
- image interpretation, 2–4, 9, 228
- image query language, 198, 202
- image scene interpretation, 225
- image segmentation, 3
- incremental learning, 13, 72, 98, 122, 126, 142, 215
- induction, 325
- inductive learning, 5
- inductive logic, 68, 72, 98, 106
- information, 70, 73, 192, 214
- initial segmentation, 133
- intentional states, 238
- internal screen, 332
- interoceptive sensory, 359
- interpretation paradigm, 4
- intersection, 243
- invariance, 68, 98, 100, 104, 172, 330
- Invariance Measure Density Function, *see* Invariance Signature
- Invariance Signature, 265, 273, 306
- biological plausibility, 266, 304, 306
- consistency of tangent with invariance, 266
- discrete, 273, 274, 306
- example, 273
- Invariance Measure Density Function, 266
- mathematical form, 267
- of specific contour, 269
- proof of invariance, 267
- Invariance Space, 271, 273
- uniqueness, 273
- vector fields, 272
- Local Measure of Consistency, 265
- Neural Network Classifier, *see* Invariance Signature Neural Network Classifier
- Invariance Signature Neural Network Classifier, 274, 285, 306
- character recognition with, *see* ISNNC, for optical character recognition
- classification module, 285
- dynamic weights, *see* dynamic weights
- Invariance Signature, calculation, 283
- Lie vector field generation, 276
- centroid image, 276
- local measure of consistency, calculation, 282
- local orientation extraction, 279
- discontinuities, 280
- neural module, 281
- perfect and network-estimated, 289
- Invariance Space, *see* Invariance Signature
- invariant pattern recognition desirability of, 256

- methods for, 256–259  
 cross-correlation, 257  
 invariants, algebraic, 259  
 invariants, differential, 259  
 moments, geometrical, 257  
 moments, Zernike, 257  
 parts and relationships, 257  
 transform, Fourier, 257  
 transform, Hough, 258  
 transform, integral, 257
- ISNNC, construction, *see* Invariance Signature Neural Network Classifier
- ISNNC, for optical character recognition, 285  
 ambiguous characters, 286  
 computation time, reduced, 289  
 data dimensionality, reduction of, 287, 307  
 local orientation extraction effect on performance, 294  
 perfect and network-estimated, 289  
 misclassifications, “human-like”, 293, 304  
 perfect data, 286–294, 306  
 data set, 286  
 networks used, 287  
 results for ISNNCs, network-estimated local orientation, 293  
 results for ISNNCs, perfect local orientation, 292  
 results for TNNs, 289  
 quantisation noise, 286  
 real data, 294–305  
 data set, 296  
 failure analysis, 304  
 networks used, 302  
 results for ISNNCS, 302  
 results for ISNNCS, unambiguous characters, 303  
 results for TNNs, 302
- thinning, 296
- Java, 225, 252
- knowledge, 191  
 acquisition, 196  
 background, 326  
 base, 124, 176, 198, 208  
 representation, 2, 4, 5, 9, 196  
 sources, 120  
 task-specific, 190
- konicellular, 311
- label, 70, 76, 122, 127, 128, 213, 327, 337  
 label compatibility, 85, 97, 139, 179
- lattice, 210
- least generalisation, 83, 97, 141
- Lie transformation groups, 260, 265  
 associated vector field, 261  
 definition of, 260  
 functions invariant under, 264  
 condition for, 265  
 interpretation, 265  
 generator of, 261  
 finite transformation from, 261
- invariance, and, 260  
 rotation, derivation of, 262
- Local Measure of Consistency, *see* Invariance Signature
- logic programming, 226
- low bandwidth, 190, 203
- machine learning, 5, 18, 22, 140, 190, 197, 200, 212, 325
- macula, 310
- magnocellular, 310
- MBNN, 15, *see* Model-Based Neural Network
- median filter, 243
- minimum message length, 65, 78, 97, 116, 218

- Model-Based Neural Network, 255, 274, 282–284  
MORSE, 21  
multi-media database systems, 2  
multi-scale, 26, 63  
  
network of frames, 227  
networks of frames, 14  
neural AND, 277  
neural network, 2, 15, 237  
    Kohonen, 346  
    recurrent, 346  
    self-organising, 346, 360  
non-parametric clustering, 7  
noumena, 194, 210  
  
object recognition, 120, 190  
object-centred, 326  
object-oriented, 121, 200, 229  
occlusion, 128, 338  
optical character recognition, with  
    ISNNC, *see* ISNNC for  
    optical character recogni-  
    tion  
outside memory store, 333  
overfitting, 215, 217  
  
parametric clustering, 7  
Parlog, 229  
Parlog++, 225, 230, 234, 244, 250  
part, 68, 128, 326  
part indexing, 9, 10, 70, 106, 120,  
    138  
part processing, 198  
partitioning, 83  
parts clique, 156  
PARVO, 19  
parvocellular, 310  
perception  
    bistable, 338  
    global, 337  
perceptron, 7  
perceptual concept, 337  
phenomena, 194, 210  
plausibility, 235, 242, 252  
polygonal approximation, 102  
polysemous figure, 373  
pre-compiled proposition, 107  
priming effects, 328  
procedural knowledge, 196  
Prolog, 226, 229  
proprioceptive sensory, 359  
  
quantisation noise, 286  
  
radial basis function, 6  
ratio growing, 168  
re-segmentation, 122, 134, 154, 165,  
    194, 212  
receptive fields, 310, 367, 370  
recognition, 324  
    by components, 19  
    by parts, 9, 18, 25, 62, 70,  
        193, 198  
    instantaneous, 367, 371  
    object, 120  
    symbol, 100  
    temporal, 367  
recursive 3D filter, 251, 252  
referent attractors, 337  
region growing, 168  
reinforcement, 373  
relational  
    evidence, 12, 68  
    information, 128, 148, 320  
    learning, 12, 68, 72, 79, 88,  
        95, 220, 368  
    representation, 68  
        structure, 12, 70, 79  
relaxation labelling, 122, 127, 138,  
    149, 166  
representation  
    attribute-indexed, 70  
    declarative, 196  
    Fourier, 319  
    internal, 128, 138, 209, 316,  
        331  
    part-indexed, 70  
    procedural, 196  
    relational, 68

- symbolic, 191  
 retina, 310, 330  
 road intersections, 239  
 rods, 310, 330, 367  
 rule generation, 7  
 rulegraph, 12, 76, 97  
 rulegraph matching, 76  
 saccade, 314, 330, 343, 352  
 saccadic suppression, 314  
 scan-sequence, 351  
 scanpath, 343, 351  
 scene interpretation, 122, 128, 129  
 scene understanding, 120  
 SCHEMA, 120, 227  
 schematic interpretation, 68, 98  
 SEE++, 13, 190  
 segmentation, 19–22, 25–29, 31, 61–63, 103, 122, 127, 165, 190, 205, 325  
 segmentation stack, 165  
 semantic  
     integration, 336  
     label, 337  
     memory, 336  
 semi-restricted graph, 124, 127  
 sensory system, 359  
 shape, 316, 324, 339  
 SIGMA, 120, 226  
 sinusoid, 35, 53  
 situatedness, 238  
 SOO-PIN, 14, 229, 239, 250  
 sparse depth, 17, 38, 55, 64  
 spatial fusion, 330  
 spatial integration, 333  
 spatial processing, 244  
 stereo, 19, 20, 22, 23, 25, 28–30, 38, 55, 56, 64, 65, 340  
 stochastic function, 216, 220  
 structural model, 320  
 sub-graph isomorphism, 73, 139, 148, 211  
 Sun SparcStation II, 250  
 superior colliculus, 311  
 supervised learning, 6, 7, 122, 126, 138, 190, 213  
 support, 235, 242, 252  
 symbol recognition, 98  
 symbolic picture interpretation, 14  
 symbolic processing, 244, 250  
 symbolic rules, 5  
 systems  
     CRG, 17–19, 31, 40, 41, 44, 46, 51–53  
     FCRG, 17–19, 40, 41, 43, 44, 50–53, 56, 58, 62, 65  
     MORSE, 21  
     PARVO, 19  
     SCHEMA, 227  
     SIGMA, 226  
     SOO-PIN, 229, 239, 250  
     TINA, 22  
 taxonomy, 125, 156, 173  
 teleoperation, 191, 221  
 template matching, 317  
 temporal, 346  
 temporal domain, 224  
 temporal sequences, 365  
 texture, 169, 170, 319  
 thinning, 296  
 threads, 252  
 TINA, 22  
 TNN, *see* traditional neural network  
 token correspondence, 243  
 top-down, 197  
 traditional neural network, 255  
 traffic jam, 242  
 traffic scenario, 240  
 traffic scenes, 239, 251  
 training set, reduction of size of, 255, 256, 290, 305, 307  
 trajectory, 244, 252  
 trans-saccadic fusion, 332, 333  
 Treisman figure, 371  
 unary  
     feature, 70, 73

unary feature, 122, 138, 170  
unary hypothesis, 152  
unary matching, 133, 141  
unary query, 211  
uncertainty, 234, 252  
union-find algorithm, 112  
unsupervised learning, 6  
  
vectorised, 347  
vehicle activities, 239  
velocity, 244, 252  
videotape, 243  
vision  
    goal-directed, 192  
visual interpretation, 127, 151  
  
world filter, 369  
world view, 322