

**CONJUNCTIONS IN BIOLOGICAL NEURAL ARCHITECTURES  
FOR VISUAL POSE ESTIMATION**

**TOM´AS MAUL**

**FACULTY OF COMPUTER SCIENCE AND INFORMATION  
TECHNOLOGY UNIVERSITY OF MALAYA  
KUALA LUMPUR**

**2006**

**CONJUNCTIONS IN BIOLOGICAL NEURAL ARCHITECTURES  
FOR VISUAL POSE ESTIMATION**

**TOM'AS MAUL**

**THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**FACULTY OF COMPUTER SCIENCE AND INFORMATION  
TECHNOLOGY UNIVERSITY OF MALAYA  
KUALA LUMPUR**

**2006**

# Abstract

The current thesis is concerned with how biological systems solve the computational problem of visual pose estimation. Four levels of analysis are traversed in order of decreasing abstraction: computational, algorithmic, implementational and formational. As each level is traversed, a biological plausibility argument is gradually strengthened. At the algorithmic level, several approaches for solving the pose estimation problem are compared in terms of their neural implementability. A highly parallel approach based on simple inter-map conjunctions, or correspondence distributions, is chosen and tested on synthetic and real patterns. The accuracy and robustness of the approach are demonstrated in relation to various critical environmental factors. At the implementational level, the algorithm is translated into various artificial neural architectures. Several maximum value networks are investigated and compared in this context. Combinatorial issues regarding the numbers of nodes and connections are analyzed. The analyses suggest that the architectures can satisfy biological constraints. The spatial arrangement of nodes in different architectures is optimized via an elastic network, with the goal of minimizing the total wiring length between nodes, revealing novel and interesting design principles, some of which correlate with several aspects of biological neural maps. Other revealing links to biological findings are discussed, such as the computation of conjunctions at the level of dendritic branches. Following this, at the formational level, various local mechanisms are investigated in the context of the biological development of the proposed neural architectures. It is shown that simple local rules, together with visual experience, such as that provided by dynamic images, are sufficient for the development of the neural architectures. The generalization of inter-map conjunctions is discussed in the context of other visual functions and sensory modalities. Some pointers towards methodologies for uncovering direct evidence of inter-map conjunctions are also provided. The general hypothesis supported by the thesis states that at least some biological neural systems are likely to be using inter and intra-map conjunctions for efficiently solving computational problems such as visual pose estimation.

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C/Passport No: )

Registration/Matric No:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis (“this Work”):

Field of Study:

I do solemnly and sincerely declare that:

- (1) I am the sole author/writer of this Work;
- (2) This Work is original;
- (3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
- (4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
- (5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya (“UM”), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
- (6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate’s Signature

Date

Subscribed and solemnly declared before,

Witness’s Signature

Date

Name:

Designation:

# Acknowledgments

I would like to thank my supervisor Assoc. Prof. Dr. Mohd. Sapiyan Baba for his guidance, ideas and open-mindedness. He created the perfect research atmosphere by allowing certain risks whilst keeping clear track of the work's direction. I am very grateful for the environment he created. He was also crucial in clarifying the thesis and revealing several hidden assumptions.

I would also like to thank several colleagues. In particular, Yeong Pong Meau, for his mathematical expertise and the many discussions he participated in, and in general, all other colleagues of the Artificial Intelligence Laboratory, who contributed to a friendly and inspiring work atmosphere.

This work has significantly benefited from advice and hints from a long string of researchers, via various informal communications. Specifically, I would like to thank: David Arathorn, Dave Casasent, Tansu Celikel, Peter Földiák, Geoffrey Hinton, Guosong Liu, Bartlett Mel, Yoichi Miyawaki, Mike Mozer, Riadh Mtibaa, Bruno Olshausen, Alexandre Pouget, James Reggia, Reiner Schulz, Terrence Sejnowski, Patrice Simard, Steve Skiena, Javier Traver and Laurenz Wiskott.

Most importantly I would like to express my deepest gratitude to my family, who have always supported all my endeavors and without which this work would be meaningless if not impossible. I would like to thank my father for his endless support, for his liberal attitude and trust regarding my choices, for providing an example of perfectionism and logical clarity, and for having the wisdom to advise me to never close any doors. I hope that I can make all your efforts worthwhile. I am also grateful to my mother for her support and creativity. My sister's character and career have often reminded me of what should be the main guiding principle behind any endeavor: to help others. I would also like to thank my grandparents for the archetypal inspiration and wisdom they continuously provide. Thank you also to my mother and father in law for painstakingly helping us at this crucial transition phase: babysitting is probably the toughest job in the world. Lastly and most fundamentally, I would like to say thank you to my wife, who has supported me in all possible ways, and who has brought a meaning and happiness to life which I hardly deserve. Your warmth, intelligence and humor inspire me everyday. Thank you for putting up with my stubborn obsession with time and for making all of this possible.

All shortcomings to be found within these pages are my sole responsibility. Any useful insights, on the other hand, would not be possible without the direct or indirect influence of those acknowledged above, or those that have been inadvertently omitted.

# Contents

<b>Abstract</b>	<b>v</b>
<b>Acknowledgments</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Computational Neuroscience of Vision	1
1.2 Levels of Abstraction	2
1.3 Objectives	3
1.4 Main Argument	4
1.5 Research Methodology	5
1.6 Main Contributions	6
1.7 Thesis Structure	7
1.7.1 The Chapters	7
1.7.2 A Short-Cut	8
<b>2 Thesis Overview: Four Levels of Abstraction</b>	<b>9</b>
2.1 The Framework	9
2.2 Computation	11
2.2.1 Pose Estimation	11
2.2.2 Importance	12
2.2.3 Difficulty	13
2.3 Algorithm	13
2.3.1 Types of Algorithms	14
2.3.2 Correspondence Distributions	15
2.4 Implementation	15
2.4.1 Computational Devices	15
2.4.2 Artificial Neural Networks	16
2.4.3 Artificial Neural Correspondence Distributions	17
2.4.4 Biological Neural Networks	20
2.4.5 Biological Neural Correspondence Distributions	22
2.5 Formation	23
2.5.1 Types of Formation	23
2.5.2 Development of Correspondence Distributions	24
<b>3 Algorithms for Pose Estimation</b>	<b>26</b>
3.1 Introduction	26
3.2 Template based approaches	27
3.2.1 Deformable Models	27
3.2.2 Map-Seeking Circuits	28
3.2.3 Dynamic Routing Circuits	30
3.2.4 What-and-Where Filter	32

3.3	Correspondence based approaches . . . . .	33
3.3.1	Search-based . . . . .	33
3.3.2	Vote-based . . . . .	36
3.4	Global Properties . . . . .	42
3.4.1	Center of Mass . . . . .	42
3.4.2	Symmetry Axes . . . . .	43
3.5	Conclusion . . . . .	44
<b>4</b>	<b>Single Correspondence Analysis</b>	<b>46</b>
4.1	Basic Concepts . . . . .	46
4.1.1	Inputs . . . . .	46
4.1.2	Correspondences . . . . .	48
4.1.3	Local Invariant Features . . . . .	50
4.1.4	Problem Definition . . . . .	52
4.2	Pose Estimation Algorithms . . . . .	53
4.2.1	Transformations . . . . .	53
4.2.2	Constrained Estimation . . . . .	60
4.2.3	Unconstrained Estimation . . . . .	63
	<b>Abbreviations</b>	<b>46</b>
<b>5</b>	<b>Estimation Accuracy</b>	<b>67</b>
5.1	Synthetic Patterns . . . . .	67
5.1.1	Introduction . . . . .	67
5.1.2	Feature Repeatability . . . . .	68
5.1.3	Spurious and Missing Features . . . . .	71
5.2	Real Patterns . . . . .	72
5.2.1	Pre-processing . . . . .	73
5.2.2	The PTC of Real Patterns . . . . .	75
5.2.3	Real Pattern Accuracy . . . . .	76
<b>6</b>	<b>Artificial Neural Architectures</b>	<b>79</b>
6.1	Artificial Components . . . . .	79
6.2	Feature Maps and Columns . . . . .	81
6.3	Correspondence Detectors . . . . .	82
6.4	Transformation Voting Structures . . . . .	83
6.5	Maximum Value Networks . . . . .	85
6.5.1	Lateral Inhibition . . . . .	86
6.5.2	Mean Based . . . . .	92
6.5.3	Paired Comparisons . . . . .	93
6.5.4	Advantages and Disadvantages . . . . .	94
6.6	Correspondence Vote Normalization . . . . .	97
6.7	Constrained Estimation . . . . .	97
6.8	Unconstrained Estimation . . . . .	98
<b>7</b>	<b>Architectural Analysis</b>	<b>101</b>
7.1	Combinatorial Explosions . . . . .	101
7.1.1	Nodes . . . . .	102
7.1.2	Connections . . . . .	104
7.1.3	Constraints . . . . .	105
7.1.4	Containing Explosions . . . . .	108

7.2	Structural Optimization . . . . .	113
7.2.1	Introduction . . . . .	113
7.2.2	Assumptions and Simplifications . . . . .	114
7.2.3	Optimization Methods . . . . .	115
7.2.4	Examples . . . . .	118
7.2.5	Conclusions . . . . .	126
<b>8</b>	<b>Biological Neural Architectures</b>	<b>129</b>
8.1	Biological Support . . . . .	129
8.1.1	Topographic Maps . . . . .	129
8.1.2	Features . . . . .	133
8.1.3	Neuronal Variety . . . . .	135
8.1.4	Dendritic Conjunctions . . . . .	137
8.1.5	Example Constrained Architecture . . . . .	138
8.1.6	Example Unconstrained Architecture . . . . .	140
8.2	Biological Constraints . . . . .	141
<b>9</b>	<b>Development of Connection-Patterns</b>	<b>145</b>
9.1	Introduction . . . . .	145
9.2	Main Elements . . . . .	147
9.2.1	Neural Trace . . . . .	148
9.2.2	Random Rewiring . . . . .	148
9.2.3	Fixation Levels . . . . .	149
9.3	Algorithm . . . . .	150
9.4	Experiments . . . . .	158
9.4.1	Time vs. Fixation . . . . .	159
9.4.2	Feature Sparsity . . . . .	160
9.4.3	False Correspondences . . . . .	165
9.4.4	Natural Images . . . . .	167
9.5	Rigid Transformations . . . . .	168
9.5.1	Algorithm . . . . .	169
9.5.2	Rotation . . . . .	169
9.5.3	Translation and Rotation . . . . .	170
9.6	Probabilistic considerations . . . . .	173
9.7	Structurally Optimal Development . . . . .	175
<b>10</b>	<b>Discussion</b>	<b>178</b>
10.1	Summary . . . . .	178
10.2	Strengths and Weaknesses of Correspondence Distributions . . . . .	179
10.3	Direct Evidence . . . . .	180
10.3.1	Multicolored Labeling . . . . .	180
10.3.2	Birds and Humans . . . . .	182
10.4	Expensiveness . . . . .	183
10.5	Artificial and Biological Synergy . . . . .	184
10.6	Generalizations . . . . .	187
10.6.1	Visual Applications . . . . .	187
10.6.2	Modalities . . . . .	190
10.6.3	Inter-Map Computations . . . . .	191
10.7	Future Work . . . . .	191
10.8	Conclusion . . . . .	194



<b>A Image Processing</b>	<b>196</b>
A.1 Saliency . . . . .	196
<b>B Proofs</b>	<b>199</b>
B.1 Total Connection Length for Maximum Networks . . . . .	199
B.1.1 Lateral Inhibition I and II . . . . .	199
B.2 Average Error for Random Guessers . . . . .	200
B.3 Developmental Probabilities . . . . .	201
B.3.1 Matching Synaptic Pairs . . . . .	201
<b>C Input-Clouds</b>	<b>203</b>
C.1 Introduction . . . . .	203
C.2 Types of Clouds . . . . .	205
C.3 Cloud Behaviour . . . . .	207
C.4 Applications . . . . .	210
C.5 Conclusion . . . . .	212
<b>Bibliography</b>	<b>213</b>

University of Malaya

# List of Figures

2.1	A hierarchy of problems. . . . .	10
2.2	Some image transformations. . . . .	12
2.3	A Higher Order Neural Network (b) and a specialization (a). . . . .	18
2.4	Biological neural networks implementing an exclusive OR. . . . .	21
3.1	Simplified diagram of a map-seeking circuit. . . . .	30
4.1	An input-cloud, an input-vision and some correspondences. . . . .	47
4.2	Various distributions of correspondence properties. . . . .	49
4.3	Example salience filtering. . . . .	51
4.4	Constrained transformations applied to a single point. . . . .	58
4.5	An input-cloud and a set of correspondences and votes. . . . .	61
4.6	Two voting surfaces. . . . .	64
4.7	The intersection of $T_x$ estimation surfaces. . . . .	65
5.1	Two examples of synthetic patterns. . . . .	68
5.2	Three accuracy critical factors. . . . .	68
5.3	PTC as a function of feature repeatability. . . . .	71
5.4	The effect of spurious and missing features on accuracy. . . . .	74
5.5	Example salience-filtering and thinning. . . . .	74
5.6	COIL objects and their PTCs. . . . .	76
5.7	Examples of transformations, resulting input-visions and estimations. . . . .	77
6.1	Artificial Neural Components. . . . .	80
6.2	Feature richness and feature map columns. . . . .	82
6.3	Correspondence detection. . . . .	83
6.4	Correspondences voting for translations. . . . .	84
6.5	A lateral inhibition network. . . . .	86
6.6	Node competition with rate conditioned weights. . . . .	88
6.7	$W^-$ depends on the sum of node activities. . . . .	90
6.8	Hierarchical competitive network. . . . .	91
6.9	Mean based networks. . . . .	92
6.10	Paired comparisons. . . . .	93
6.11	ANA for constrained pose estimation. . . . .	98
6.12	ANA for unconstrained pose estimation. . . . .	99
6.13	Correspondences excite/vote-for $T_x$ and $T_y$ surfaces. . . . .	100
7.1	Example equation derivation. . . . .	103
7.2	Dealing with excessive vote-cell inputs. . . . .	107
7.3	Two 3D voting structures. . . . .	110
7.4	The results of two structural optimization methods. . . . .	118
7.5	A sequence of elastic network steps. . . . .	119

7.6	The $(T_x, T_y)$ case with a single feature type. . . . .	120
7.7	The $(T_x, T_y)$ case with ten distinct feature types. . . . .	122
7.8	Elastic optimization for both the scale and rotation cases. . . . .	123
7.9	The $(T_x, T_y, \theta)$ case with ten feature types, optimized in 3D space. . . . .	125
7.10	The $(T_x, T_y, \theta)$ case with two feature types, optimized in 2D space. . . . .	126
8.1	The preservation of position and orientation relationships. . . . .	130
8.2	Biological neural architecture for a simple $(T_x, T_y)$ estimator. . . . .	139
8.3	A SCAPE BNA for a simplified unconstrained estimation case. . . . .	141
8.4	Dendrites with terminal conjunctions and inner summations. . . . .	143
9.1	Three stages of SCAPE development. . . . .	157
9.2	Changes in fixation levels. . . . .	158
9.3	Time versus overall fixation level. . . . .	160
9.4	The effect of feature sparsity on convergence time. . . . .	162
9.5	The effect of feature sparsity on development. . . . .	164
9.6	Larger speed-change intervals do not help the sparse features case. . . . .	164
9.7	The effect of different probabilities of true correspondences. . . . .	166
9.8	A natural image and its PTC-filtered version. . . . .	167
9.9	SCAPED performance on a natural image. . . . .	168
9.10	The development of rotation estimators. . . . .	170
9.11	The development of estimators for rigid transformations. . . . .	172
9.12	At least one match probabilities. . . . .	174
9.13	The development of SCAPE networks with a degree of topography. . . . .	176
10.1	Visualizing inter-map conjunctions without positional information. . . . .	181
10.2	Visualizing inter-map conjunctions with positional information. . . . .	182
10.3	A very cheap translation estimator. . . . .	185
10.4	A possible correspondence-based architecture for stereopsis. . . . .	189
10.5	Artificial neural architecture for a shape distribution. . . . .	189
10.6	A neural architecture for finding the centroid of a shape. . . . .	190
10.7	A simple architecture for sound localization. . . . .	191
A.1	Mean deviation filtering. . . . .	197
A.2	BRP filtered images. . . . .	197
C.1	Simplified illustration of a local-motion cloud. . . . .	205
C.2	Simplified illustration of a global-motion cloud. . . . .	206
C.3	Some of the forces that may underlie cloud behavior. . . . .	207
C.4	Some possible cloud behaviours. . . . .	208
C.5	Multiple cloud and object spatial relationships. . . . .	209
C.6	Part of the segmentation process using multiple clouds. . . . .	211
C.7	Applicability of input-clouds to multiple object tracking. . . . .	211

# List of Tables

3.1	A qualitative comparison of pose estimation approaches. . . . .	44
4.1	Transformation groups and their invariants. . . . .	54
5.1	The effect of feature repeatability on error. . . . .	70
5.2	The average error of a random guesser. . . . .	70
5.3	Mean chance and algorithmic estimation errors. . . . .	77
6.1	Comparing maximum-value networks. . . . .	96
7.1	Comparison of estimation errors between $4D$ and $3D$ architectures. .	113
8.1	Various human cortical constraints. . . . .	141
9.1	Standard SCAPED parameters. . . . .	159

University of Malaysia

# Abbreviations

<b>AHT</b>	Adaptive Hough Transform
<b>AMPA</b>	Alpha-amino-3-hydroxy-5-methyl-4-isoxazole propionate
<b>ANA</b>	Artificial Neural Architecture
<b>ANN</b>	Artificial Neural Network
<b>BNA</b>	Biological Neural Architecture
<b>C-Coverage</b>	Representational coverage of correspondences
<b>CDA</b>	Correspondence Distribution Analysis
<b>COIL</b>	Columbia Object Image Library
<b>C-SCAPE</b>	Constrained SCAPE
<b>DRC</b>	Dynamic Routing Circuits
<b>EPSP</b>	Excitatory Post-Synaptic Potential
<b>GHT</b>	Generalized Hough Transform
<b>HONN</b>	Higher Order Neural Network
<b>IU</b>	Image Understanding
<b>L2/3</b>	Cortical Layers 2 and 3
<b>MOR</b>	Multiple Object Recognition
<b>MOT</b>	Multiple Object Tracking
<b>MLP</b>	Multilayered Perceptron
<b>NR</b>	Number of Non-Repeating Features
<b>PFC</b>	Probability of a False Correspondence
<b>PTC</b>	Probability of a True Correspondence
<b>PTD</b>	Probability of a True Doublet
<b>PTS</b>	Probability of a True Singlet
<b>R</b>	Number of Repeating Features
<b>RANSAC</b>	Random Sample Consensus
<b>RAST</b>	Recognition by Adaptive Subdivisions of Transformation Space
<b>RUDR</b>	Recognition Using Decomposition and Randomization
<b>SCAPE</b>	Single Correspondence Analysis for Pose Estimation
<b>SCAPED</b>	SCAPE Development
<b>T-Coverage</b>	Representational coverage of transformations
<b>TWL</b>	Total Wiring Length
<b>U-SCAPE</b>	Unconstrained SCAPE
<b>VSA</b>	Visual Scene Analysis

# Chapter 1

## Introduction

*Truth in science can be defined as the working hypothesis best suited to open the way to the next better one.*

– Konrad Lorenz

### 1.1 Computational Neuroscience of Vision

Vision consists of many different but interrelated capabilities, e.g: object classification, facial expression interpretation, motion analysis, depth perception, 3D volume perception and others. The ease with which we humans perform these functions obscures us to the incredible complexity of the mechanisms underlying them. It has taken evolution hundreds of millions of years, to go from the simple visual capabilities of trilobites in the Cambrian Period, to those observed for example, in primates today. It is worthwhile to recall the often cited, possibly fictitious, yet illustrative anecdote, where a famed professor sometime around 1966, requested one of his students to solve the problem of computer vision as a summer project. Many years and brains later, the problem remains largely unsolved, except for highly-specialized applications and cases with controlled environmental parameters.

The upside to vision's complexity and apparent intractability, is that it provides us with a rich set of problems within which to conduct research on Neural Computation. Object recognition is possibly one of the problems currently attracting the greatest deal of attention. Being a highly complex and resource demanding

problem it has been investigated mostly in the context of several specializations, e.g: faces, fingerprints, handwritten characters, vehicles, histological elements, and others. One of the greatest difficulties faced by object recognition systems stems from the fact that every object has a virtually endless set of appearances. It has long been known that if such appearances can be compensated for (i.e. normalized), the recognition process can be greatly simplified. Geometric transformations constitute an important source of such appearances and provide objects with varied poses in terms of position, size, rotation and other transformations. The poses of objects are not only as useful in their own right as their identities (i.e. recognition), but as already argued can facilitate in the determination of the latter. Being a somewhat forsaken yet crucial ability, we decided that pose estimation would be a great candidate for investigating how it might be solved in biological neural systems.

Questions pertaining to the nature, location and *raison d'être* of specialized and general computations in biological neural systems are mainly the domain of Computational Neuroscience. Being by definition an interdisciplinary field it combines a host of methodologies, levels of abstraction and underlying motivations. For example, some researchers will prefer to focus on aspects such as spike dynamics and coding strategies, while others choose to focus on architectonic and developmental aspects. Our thesis belongs to the latter category. The computational problem of visual pose estimation will be investigated in the context of Computational Neuroscience, with special emphasis on representational, architectonic and developmental issues.

## 1.2 Levels of Abstraction

The problems faced by the current thesis are best understood when contextualized by a framework consisting of four levels of abstraction, i.e: computational, algorithmic, implementational and formational. Here we will briefly introduce the framework, which will be explained in greater detail in Chapter 2.

The first level is essentially concerned with formulating the computational problem that needs to be addressed, which in this case refers to pose estimation. The

pose estimation problem can be defined as follows: given two patterns, which are equivalent to each other except for a geometric transformation<sup>1</sup>, estimate the transformation that relates them. The geometric transformations considered in the thesis belong to the similarity group which consists of: 2D translation, scaling and rotation.

At the algorithmic level we are concerned with formulating information processing steps which are capable of solving the pose estimation problem. The best candidate from a neural implementability perspective was deemed to be an approach based on distributions of correspondences between local features: correspondence distributions. The chosen approach was shown to be significantly accurate and robust to environmental factors.

The implementational level is concerned with the embodiment of algorithms in particular computational devices. In this context, we translated our chosen algorithm into artificial and biological neural architectures. Structural optimization experiments were conducted, revealing useful design principles and biological predictions. Relevant and recent biological findings were discussed in support of the architectures.

The formational level is concerned with how particular computational devices implementing particular algorithms can be formed. In our particular context, this involved modeling developmental mechanisms based on Hebbian-like local mechanisms (that mediate synaptic maturation and elimination), random axonic wiring and visual stimulation.

### **1.3 Objectives**

Two general objectives, with several semi-independent sub-goals, guided the current thesis:

1. To propose a simple, accurate and robust approach for neural pose estimation.
  - (a) Search theoretical, applied and biological domains for the best candidate approach.

---

<sup>1</sup>Other possible exceptions include: noise, clutter, occlusions, illumination variations, among others.



- (b) Test various performance characteristics of the approach.
  - (c) Translate the approach's algorithms into artificial neural architectures.
2. To strengthen the hypothesis that some biological systems might be implementing the proposed approach.
- (a) Perform biologically relevant architectural analyses and optimizations.
  - (b) Form connections with recent biological findings.
  - (c) Investigate developmental algorithms capable of generating the architectures.

Note that our two main objectives can be combined into a single goal: to find a biologically plausible neural solution for pose estimation. The final section of the thesis' final chapter (section 10.8) will verify that our objectives have been met with reference to this single all-encompassing goal.

It should also be noted here that temporal/dynamic issues (e.g. spike timing, rate coding and others) are serious absentees from our objectives list. Clearly they are highly relevant and important to consider. However, we chose to focus heavily on architectural and developmental considerations, partly because this permitted stronger bridges between the artificial proposals and recent biological findings.

## 1.4 Main Argument

The overall argument running through the thesis is essentially one of accumulating consistency. A hypothesis is put forth, which states that some biological systems are likely to be using correspondence distributions for solving the pose estimation problem. First the approach is demonstrated to perform accurately and robustly. This is consistent with the behaviour of many complex animals. Then the approach is shown to have simple and feasible artificial neural architectures embodying it. Without artificial neural implementability, biological plausibility is automatically invalidated. Subsequently the architectures are shown to have elegant and compact configurations, which satisfy certain structural goals found in biological systems.

The structural optimization of artificial architectures provides a bridge leading to biological topographic maps. After that, several biological findings are discussed, which allow for and partially attest to the existence of the approach in biological neural systems. Having shown that the approach is implementable in biological neural systems is only part of the consistency story: if the architectures are too complex to be created (i.e. developed) then the whole case is lost. So the final piece of consistency shows that developmental algorithms do exist, which demonstrate how architectures can emerge as a result of simple physiological mechanisms and visual experience.

An even more compact form of the argument might be stated as follows. Correspondence distributions are biologically plausible because they: 1) are accurate, 2) are neurally implementable, 3) have structurally optimal forms that correlate with biological maps, 4) have biological machinery for their representation, and 5) can develop.

## 1.5 Research Methodology

Being an interdisciplinary thesis, the research methodology did not follow any of the familiar formats from Computer Science on one side of the spectrum, or Neuroscience on the other. Instead, several methodological stages were recurrently visited as required by the thesis' argument. The stages included: problem refinement, background research, conceptual proofs, incremental simulation prototyping, modeling with MatLab, biological hypothesizing, and others. Having said this, one can probably simplify the recursive methodology into four main sequential stages, defined according to their central concerns: 1) problems, 2) solutions, 3) artificial neural networks and 4) biological neural networks. The first stage includes problem definition and gradual refinement. The second stage includes reviewing the literature, comparing approaches, and testing algorithm performance. The third stage includes the design, simulation, analysis and optimization of artificial neural architectures. Finally, the fourth stage includes the search for biological parallels and the design, simulation and analysis of developmental algorithms.

## 1.6 Main Contributions

Some of the main contributions originating from the current work may be summarized as follows:

1. Theoretical, applied and biological domains were reviewed in the search for explicit, general and neurally implementable pose estimation approaches.
2. New limits and factors regarding the performance of correspondence distributions were investigated.
3. Artificial neural architectures were designed for the representation and analysis of correspondence distributions.
4. Structurally optimal configurations for neural pose estimators were found and general design principles were revealed.
5. A biologically plausible neural pose estimator, based on correspondence distributions, was proposed.
6. Evidence was reviewed supporting the implementability and existence of correspondence distributions in biological systems.
7. A case was made for the generality of inter-map conjunctions in biological systems.
8. Developmental mechanisms were found for the formation of pose estimating neural architectures.<sup>2</sup>
9. The generalization of correspondence distributions to different application domains and modalities was discussed.
10. An experimental methodology was proposed for uncovering direct evidence of correspondence distributions in biological systems.
11. A new form of illumination normalization was proposed.

---

<sup>2</sup>To the best of the author's knowledge, there are no existing models of the development of biological neural architectures for explicit and general pose estimation.

## 1.7 Thesis Structure

### 1.7.1 The Chapters

In the introduction, the thesis' main problems and solutions are first presented, followed by an outline of the objectives, the main argument and hypothesis, and the research methodology. A summary of the main contributions is also provided.

Following this, Chapter 2 is dedicated to providing an overview of the thesis in terms of the four levels of abstraction that it traverses, i.e: computational, algorithmic, implementational and formational. Sufficient background information is provided for understanding the various solutions adopted throughout the thesis, and the various assumptions and motives underlying it.

Chapter 3 reviews several pose estimation approaches, some of which derive from theoretical domains (e.g. point pattern matching), while others originate from applied domains (e.g. Computer Vision) or more biological domains (e.g. Computational Neuroscience). A comparison between the approaches is made in order to find the best candidate for a biological neural pose estimator.

Subsequently, in Chapter 4, pose estimating algorithms using correspondence distributions are explained in detail. Basic concepts are clarified, including: correspondences, local invariant features, transformations and constrained/unconstrained estimation.

Following this, in Chapter 5, the accuracy of the approach is investigated in some depth, using both synthetic and natural patterns.

Chapter 6 translates the algorithmic approach into artificial neural architectures.

In Chapter 7, the artificial neural networks of the previous chapter are first architecturally analyzed and then structurally optimized according to a wiring length minimization goal.

Subsequently, Chapter 8 is dedicated to finding evidence for the existence of correspondence distributions in biological systems.

The penultimate chapter, Chapter 9, is concerned with exploring developmental algorithms for the formation of the pose estimation architectures discussed in the

previous chapters.

Finally, Chapter 10 summarizes the findings, discusses several issues such as direct evidence and the generality of correspondence distributions, and proposes several directions for future work.

## 1.7.2 A Short-Cut

For the sake of those readers who have limited time, and that would like to get the essence of the thesis without having to go through the whole document, we present here a selection of chapters/sections, which encompasses approximately half of the material, and which represents the so called heart of the thesis:

- i) Chapter 1.
- ii) Chapter 4.
- iii) Chapter 6 excluding sections 6.5 and 6.6.
- iv) Section 7.2 in Chapter 7.
- v) Subsections 8.1.4 to 8.1.6 in Chapter 8.
- vi) Chapter 9.
- vii) Chapter 10.

In this chapter, we have introduced our main hypothesis and argument, briefly described our problems and solutions, outlined the levels of abstraction addressed, listed our main objectives and contributions and outlined the structure running through the thesis. In the following chapter we will overview the thesis in greater detail with reference to the four abstraction levels that it traverses.

# Chapter 2

## Thesis Overview: Four Levels of Abstraction

The aim of this chapter is to provide an explanatory overview of the thesis. Brief but sufficient background information will be given and related to the work's main findings. Since the thesis traverses several levels of abstraction, each with its own set of problems and hidden motives, it should be easier to understand the resulting structure, if this framework is made explicit.

### 2.1 The Framework

Figure 2.1 offers a global perspective of the problems being tackled in the thesis and how they fit into the broader framework defined by general intelligence. The problem of intelligence, as is well known, can be sub-divided into a vast number of sub-components: vision is one of these. In its turn, the problem of vision consists of another vast array of sub-problems. One of these is our chosen research area, i.e: pose estimation. Finally, the problem of pose estimation can be sub-divided into computational, algorithmic, implementational and formational sub-problems, presented in order of decreasing abstraction.

The computational problem is defined by the information that is being used by some system (i.e. input), and what is being done to it (i.e. output). In the case of pose estimation the input is a pair of patterns, and the output is the transformation

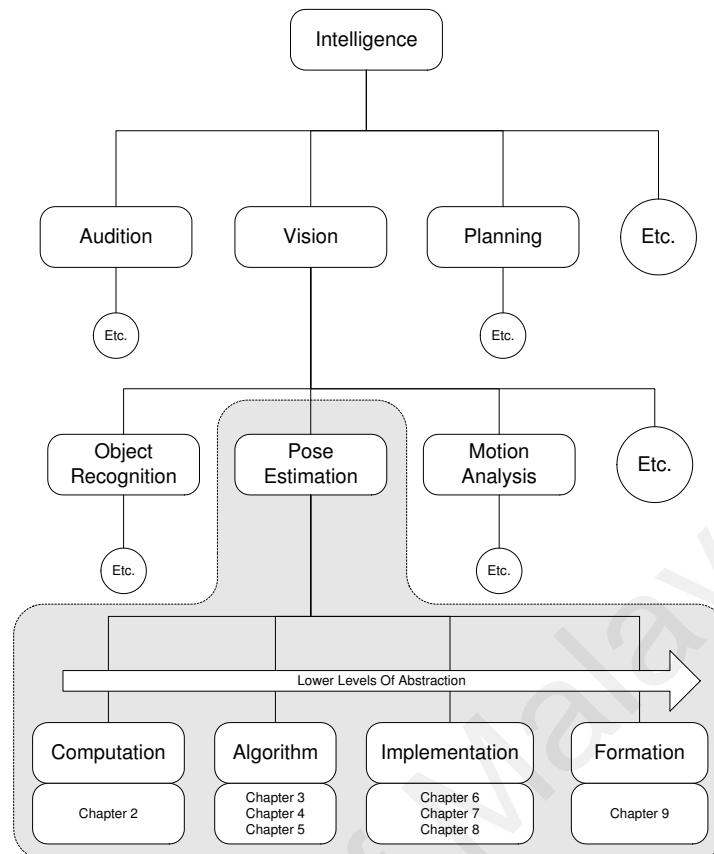


Figure 2.1: A hierarchy of problems.

that relates them. This level of abstraction specifies *what* is being computed without being concerned about *how* it is computed (see Palmer 1999).

The algorithmic problem is concerned with *how* to solve the computational problem in terms of information processing operations. In other words, it is concerned with specifying the information processing steps required for establishing the input/output relations defined at the computational level. In general, the same computational problem can be solved by many different algorithms.

The implementational problem is concerned with how to embody the algorithmic solution in some physical computational device, e.g: sequential computer, parallel computer, multilayered perceptron, avian brain, primate brain, and many others. Issues regarding representation, physical constraints, packaging, dynamics, and so on, are characteristic of this level of abstraction.

It is important to note that the above three problems or levels of abstraction (i.e. computational, algorithmic and implementational) are part of Marr's metatheory regarding the study of vision (see Marr 1982). In this context, we have added one

more problem or level, i.e: formational.

The formational problem is concerned with how a computational device implementing a certain algorithm, which in turn solves a particular computational problem, is formed or constructed. In the context of conventional computers this level refers to the engineering problems involved in the manufacture and assembly of circuit boards and other components. In the context of biological neural systems, the formational problem attempts to answer questions regarding how particular architectures can develop.

The bottom part of Fig. 2.1 gives a rough indication of where each abstraction level can be found relative to the thesis' chapters. The subsequent sections will take a deeper look at each one of the abstraction levels, providing background information wherever necessary and forming connections to the contents of the thesis.

## **2.2 Computation**

This level of abstraction attempts to answer several questions including: 1) what is the computation, 2) why is the computation needed, 3) how difficult or complex is it and 4) where is the computation known to be carried out (e.g. in what artificial and/or biological systems). Here we will focus on the first three sub-problems. The fourth question is partially answered in the context of the second one.

### **2.2.1 Pose Estimation**

The problem of pose estimation is essentially the problem of determining the pose of an object, i.e: it's position, size, orientation, depth inclination, articulated configuration, and so on. Seeing that pose is relative, one might say, it is the problem of determining the transformation that relates two patterns (e.g. a source and a target pattern). Thus, the inputs to the computation are two patterns, while the output consists of the transformation that relates them. Refer to Fig. 2.2 for some examples of transformations.

In order to better focus limited resources, the thesis will concentrate on similarity transformations, i.e: translation, scaling and rotation.



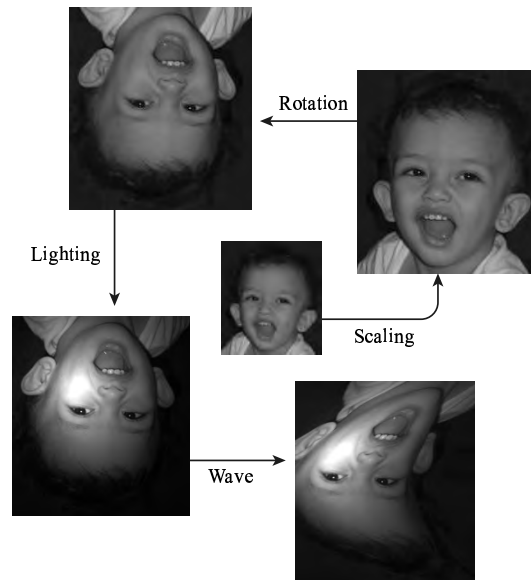


Figure 2.2: Some image transformations.

### 2.2.2 Importance

One of the reasons why pose estimation is important is that it permits organisms to interact flexibly with the environment. Object manipulation for example requires precise estimations of orientation, depth, articulation and so on. It is hard to imagine how complex object grasping and manipulation can happen without the ability to estimate the poses of target objects. Pose estimation also lies at the basis of many navigational abilities, whether it be through a cluttered room, a jungle or over a city from an aerial perspective.

Pose estimation also facilitates object recognition. Once the pose of an object has been estimated, this information can be used to normalize the object's view in order to facilitate a comparison with a stored canonical representation of it. Mounting evidence suggests that in certain instances humans do perform some sort of mental normalization (e.g. mental rotation) in order to facilitate object recognition. This can be experienced, for example, by attempting to read an upside-down newspaper.

From the perspective of the Neurosciences, and regarding the primate brain in particular, there is another reason why pose estimation is important: it may shed light on the interactions between the dorsal and ventral visual processing streams (see Ungerleider & Haxby 1994). A clearer understanding of neural pose estimation, should provide a new framework within which to investigate the interactions

between invariant object recognition (e.g. Lueschow et al. 1994) and the estimation of transformations (e.g. see Rao & Ballard 1997). It might even be possible to uncover the brain's global representational and computational strategies regarding invariant recognition and the estimation of variations in general.

### 2.2.3 Difficulty

One of the reasons for the difficulty of pose estimation lies in the combinatorial explosion of the transformations involved. For example, considering similarity transformations and assuming that the range of horizontal and vertical shifts encompasses 100 positions and that the range of scales and rotations includes 36 possibilities each, this means that 12 million and 960 thousand different transformation combinations are possible. If one were to exhaustively search for the transformation relating a source to a target pattern, by applying each transformation combination to the target and then comparing the result to the source, and assuming that each comparison requires 0.1 seconds, the estimation would take approximately 15 days to complete. Increasing the size of the transformation group (e.g. shearing, 3D depth) obviously aggravates the combinatorial explosion even further.

If the pose estimation algorithm requires local image information, then it stumbles on another heavy obstacle, i.e: invariances. How can the algorithm extract useful local information, if the transformation being estimated causes the local information to change? In other words, how can local image information be found that is discriminating and yet invariant to the relevant transformations?

## 2.3 Algorithm

Now that we have defined our target computation, we must specify how we want it to be carried out in terms of information processing operations. In other words, what algorithm is going to implement the input/output mapping mentioned in the previous section?

The algorithmic level of abstraction consists of several aspects including: 1) algorithm specification, 2) algorithm comparison, 3) performance evaluation (e.g.

accuracy, robustness, graceful degradation, temporal efficiency, among others), 4) elegance and 5) modularity. The thesis focuses on the first three problems.

### 2.3.1 Types of Algorithms

Algorithms for solving the computational problem of pose estimation can be categorized in different ways. We have chosen a simple taxonomy which categorizes algorithms based on the type of information they use: 1) templates, 2) correspondences or 3) global characteristics.

In general, template based approaches rely on search, that is, on iterative transformations and comparisons of the patterns involved (e.g. Olshausen et al. 1993). Each iteration usually involves a transformation of the target pattern and then a comparison with the template. The iterations continue until both patterns are sufficiently similar, thus indicating that a true pose has been found that relates the template to the original (untransformed) pattern.

Correspondence based approaches rely on local feature information (e.g. Ullman 1996). A correspondence is simply a vector defined by two points with matching features, one in the source pattern, the other in the target pattern. In order to allow for matches, local features must be invariant to the transformations found in the patterns. In most cases, each correspondence votes for a particular pose. When the totality of correspondences are allowed to vote, the correct poses tend to be those that receive the most votes and are thus “democratically elected”.

Global characteristics can be used for the estimation of some transformations (e.g. Hong & Tan 1988). A pattern’s center of mass is a useful example in this context. A simple center of mass algorithm averages the positions of a pattern’s constituent points. If two patterns are equivalent except for a translation, then subtracting their centers of mass recovers the correct pose estimate. Other global characteristics (e.g. symmetry axes) can be used to estimate other poses (e.g. rotation).

### 2.3.2 Correspondence Distributions

In view of the fact that our underlying motivation was neural implementability in general and biological neural plausibility in particular, our chosen algorithm needed to, among other things, rely on as simple and parallelizable computations as possible. Correspondence based approaches turned out to be the best candidates. In particular, we chose an approach based on the total distribution of correspondences between two patterns and where each correspondence votes for a specific pose, i.e: correspondence distributions.

Several experiments were conducted on synthetic and real patterns in order to confirm (and further analyze) the adequacy of correspondence distributions at the algorithmic level. Several factors were tested, e.g: feature repeatability, proportion of spurious features and proportion of missing features. It was found that correspondence distributions do indeed exhibit sufficient accuracy and robustness.

## 2.4 Implementation

Now that we have defined our target computation (or problem), indicated how to solve it algorithmically, we must discuss the next lower level of abstraction, i.e: how to implement the algorithm in a physical/computing device.

This level consists of several aspects including: 1) device selection, 2) adaptation of the algorithm to the computing device, 3) spatial optimization of the physical components and connections, 4) modeling and prototyping, 5) minimization of other costs such as energy and the cost of materials and 6) robustness. The current thesis focuses on the first four aspects of the above list.

### 2.4.1 Computational Devices

Computation is essentially the processing of information via a finite set of rules. A computational device, on the other hand, is a physical system which is capable of changing states also by a finite set of rules, and in a manner which reflects some computation. Many physical systems fit the above general definitions and are thus

computational devices, e.g: 1) classical serial computers, 2) parallel computers, 3) quantum computers, 4) artificial neural networks and 5) biological neural networks. These and other devices differ along several dimensions, e.g: 1) materials, 2) computational components (logical operations, memory, and others), 3) architectures, 4) scale, 5) efficiency, 6) generality, 7) cost, 8) feasibility and 9) origination (e.g. evolution or human invention).

From the very beginning it was the thesis' aim to use artificial neural networks in general, and focus on biological neural networks in particular. Note however, that we are not dealing with actual physical devices, but instead with models of those devices.

## 2.4.2 Artificial Neural Networks

Put briefly, artificial neural networks consist of a multiplicity of simple computational units, interconnected to each other in a highly parallel manner. They draw their main inspiration from biological neural systems, which they nevertheless greatly simplify, in order to increase mathematical tractability, learnability and feasibility, among other reasons. They are used mainly to establish complex input-output mappings, achieved almost invariably through a process of learning.

Artificial neural networks come in many varieties, a brief listing of which might include: Multilayered Perceptrons, Self-Organizing Maps, Hopfield Networks, Association Networks, Radial Basis Functions, Adaptive Structure Neural Networks, Higher Order Neural Networks, Helmholtz Machines, Cellular Neural Networks and Hybrid Networks, to cite just a few (see Haykin 1999 for a good reference).

Learning is probably the most central theme in most artificial neural network research. It is also probably one of the main factors attracting scientists and engineers to the investigation of artificial neural networks. Sub-topics in this area include: learning algorithms, speed of convergence, proof of convergence, generalization, pre-processing of input data, and re-learnability. Other artificial neural concerns include: the simplicity and elegance of architectures, the ability to scale-up, spatial optimization, temporal efficiency, stability, hardware implementability

and graceful degradation.

### 2.4.3 Artificial Neural Correspondence Distributions

How can our chosen algorithm (i.e. correspondence distributions) be embodied in an artificial neural network (ANN)? First recall that the fundamental element of our solution is a correspondence, i.e: a vector defined by two points (one at the source and the other at the target pattern) with matching features. Second, note that we are considering all possible correspondences between a pair of patterns. Therefore, every point in one pattern must be represented in relation to every point in the other pattern. This second order structure can be represented in Higher Order Neural Networks<sup>1</sup> (see Bishop 1995). Note also that a match between two features, and thus a correspondence, can be represented or computed by a logical conjunction. Seeing that the second order nodes in Higher Order Neural Networks compute products, this further strengthens their suitability, assuming that input patterns are binary. Other necessary components include: summation nodes (for the counting of votes), excitatory connections and possibly inhibitory connections. All these components are available in Higher Order Neural Networks (HONNs).

Seeing that invariant pattern recognition is known to be one of the major advantages of HONNs (see Giles & Maxwell 1987 and Wood 1996), it is interesting to discover, in this context, that these networks are also useful for pose estimation, since the latter capability can be seen as the converse of the former capability, in the sense that pose estimation is concerned with extracting the same information (i.e. transformations) that the invariance property aims to throw away.

Having justified the general suitability of HONNs, it is also necessary to point that our final architecture is actually a special case: 1) only second order structure is represented, 2) two input maps rather than one are used, 3) inter-map (between the two maps) rather than intra-map second order structure is represented, 4) connections between second order and summation nodes are restricted to a weight of 1, 5) the connectivity between second order and summation nodes is one-to-one or

---

<sup>1</sup>Note that other terms have been used in the literature to refer to HONNs, e.g.: “Sigma-Pi Networks” and “Product Unit Neural Networks”.

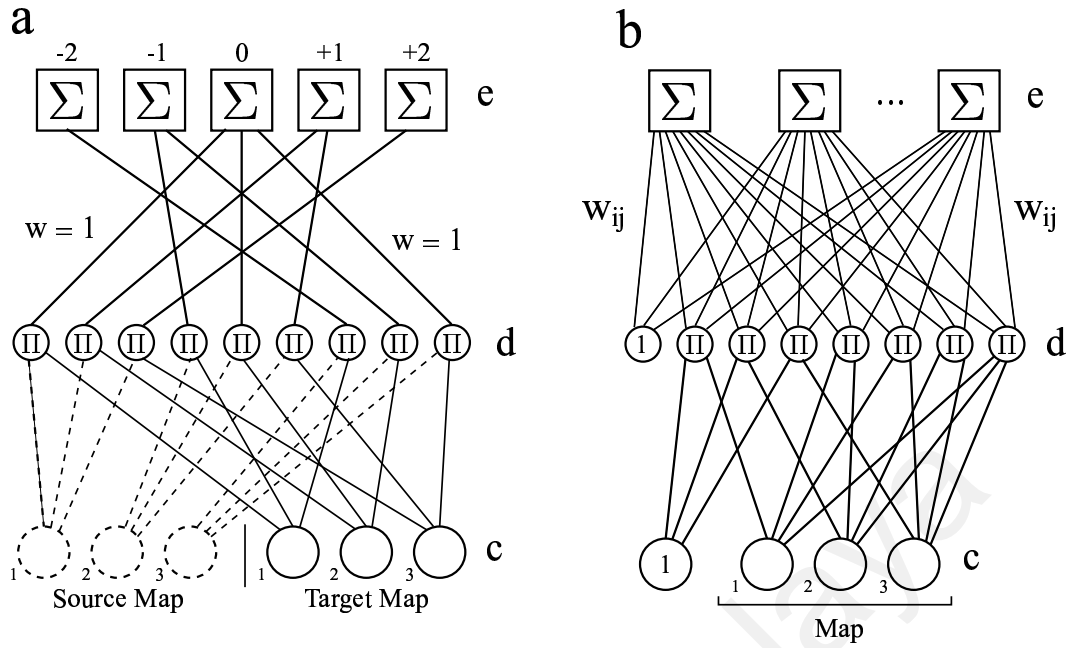


Figure 2.3: A Higher Order Neural Network (b) and a specialization (a).

one-to-many rather than one-to-all, 6) summation nodes are not biased, and 7) the connections between the second order nodes and the summation nodes are mathematically defined rather than learnt. Refer to Fig. 2.3 for a visual comparison between a HONN in (b) and a relevant second order specialization for pose estimation in (a). Input maps are denoted by (c), while multiplicative and summation nodes are denoted by (d) and (e) respectively.

One might be tempted to ask, at this point, why we do not consider neural networks with no explicit higher-order representations. To take Multilayered Perceptrons (MLPs) as an example, one might recall that they are universal approximators, and that they can thus establish any set of input-output mappings given a sufficiently large number of hidden-units. Since pose estimation consists essentially of a set of input-output mappings, then the question of the suitability of MLPs is pertinent indeed. There are probably four main reasons, why in our context, they are unsuitable: 1) an excessive amount of data is required for training (e.g.  $2^{2nt}$  binary pattern pairs are possible, where  $n$  represents map resolution and  $t$  represents the number of feature types), 2) there is no certainty that the trained network will generalize adequately for new cases, 3) training is likely to be quite slow and cumbersome and 4) the back-propagation and related supervised learning algorithms

used for training MLPs are biologically implausible.

How does our chosen architecture fair in relation to the various concerns outlined in the previous subsection on artificial neural networks? The following paragraphs provide a brief answer to this question.

As already mentioned, the architecture does not require learning in order to establish the desired input-output mappings since the architecture is mathematically predefined. Recall that correspondence votes (regarding a pose) are represented by connections between second order nodes and vote/pose nodes. Since the poses correspondences vote for are mathematically defined, so are the connections between second order nodes and vote/pose nodes. Issues regarding speed and proof of convergence are unsurprisingly also mute in this context.

Generality is an intrinsic part of the architecture. The fact that all possible correspondences between two patterns are being represented should allow poses to be accurately estimated regardless of the types of patterns and transformations involved. Pattern generality is achieved seeing that in spite of no patterns being taught to the network the latter is still capable of performing regardless of the inputted patterns. Transformation generality is achieved by virtue of the fact that the “consistency set of poses” of correspondences can be extended through simple mathematical expressions leading to an automatic redefinition of the connectivity between second order and summation nodes.

The only preprocessing required involves the extraction of invariant features from patterns. The invariances must in the very least encompass those transformations which the patterns are subjected to.

In view of the fact that the architecture consists of a few simple computational units, two processing layers, does not require recurrent connectivity, and is defined by simple mathematical equations, one may conclude, albeit with some subjectivity, that the architecture is simple and elegant.

Networks that rely on learning often encounter scaling-up issues, e.g: larger problems and/or larger networks often lead to excessively slow convergence. The fact that node inter-connectivity is mathematically defined precludes this possibility. In



this sense, scaling-up is not an issue. However, seeing that second-order structure is represented, scaling-up might be an issue in the context of hardware implementations: the number of connections and conjunctions scale with the square of the resolution of the input patterns.

The structural or spatial efficiency of components is an issue commonly found at the implementational level (e.g. packaging). In our work we explored the structural optimization of several variants of the architecture and found that interesting design principles emerged, which are useful from an engineering perspective, and which correlate with certain biological findings.

In view of the fact that the architecture is highly parallel, strictly feedforward and consists of merely two layers, it should not be difficult to appreciate its temporal efficiency. Lack of recurrence also precludes any instability issues.

The question of its hardware implementability is open to debate. The question is tied to the future of neural hardware in general and seems to hinge more on cost and perceived benefit, rather than on feasibility.

The architecture is also likely to exhibit graceful degradation, although explicit testing is required in order to uncover empirical limits. Since most pairs of patterns are likely to have multiple matching features, the elimination of a moderately small proportion of correspondence nodes, is likely to still allow, on average, for the detection of a sufficient number of matching features. This issue should be clearer in subsequent chapters.

#### **2.4.4 Biological Neural Networks**

There are significant differences between the neural networks found in biological systems (e.g. insects, birds and mammals) and those found in simplified mathematical models. Unsurprisingly the complexity of biological neural networks is vastly superior, be it in size, forward connectivity patterns, recurrence, non-linearities, dynamics and others. Furthermore, when we are considering models of biological neural networks, unlike the ideal context of most artificial neural networks, it is necessary to be aware of some fundamental constraints, e.g: 1) neurons exhibit up-

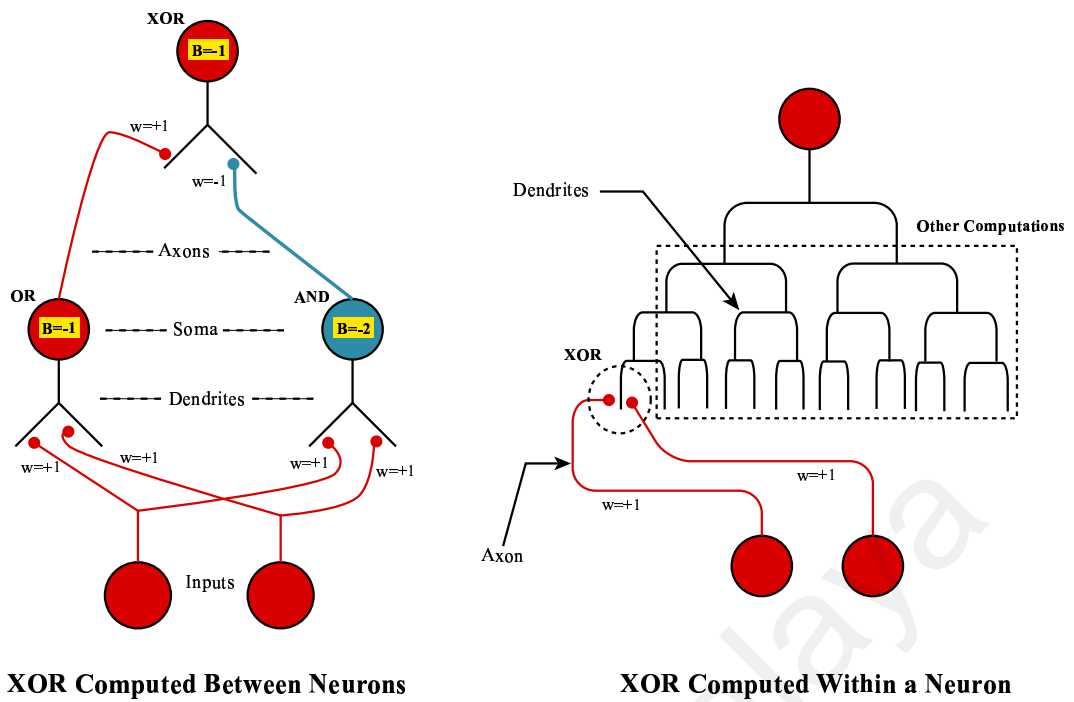


Figure 2.4: Biological neural networks implementing an exclusive OR.

per bounds to the numbers of inputs and outputs that they can receive and project respectively and 2) neurons are in the great majority of cases either excitatory or inhibitory.

A crucial question within Computational Neuroscience, which clarifies some aspects of the current thesis, asks: what is the fundamental computational unit in biological neural systems? Classically, the neuron is the fundamental unit. In this view, computation is borne out of the interconnectedness of these fundamental units, or neurons (see London & Häusser 2005). The left-hand side of Fig. 2.4 illustrates how biological neural networks, in this classical view, might implement an exclusive OR (i.e. XOR). Inputs are assumed to be binary, “B” stands for bias, “w” stands for weight, and computation is performed as in the McCulloch-Pitts model (see McCulloch & Pitts 1943). Note how several neurons are required for the computation. In view of the fact that dendritic trees are large enough to receive hundreds or thousands of synapses on average, one can conclude that this configuration ignores and thus wastes a large proportion of potential synapses.

Relatively recent research has begun to uncover certain facts which bring the classical view into question. It seems that complex computations can be carried out within neurons themselves, rather than just between neurons. Complex non-

linearities within dendritic arborizations in single neurons, capable of mediating rich computations, are increasingly being reported in the literature (see London & Häusser 2005 for an excellent review). The right-hand side of Fig. 2.4 illustrates how a single terminal dendritic branch may compute an XOR via a particular non-linearity. Not only is this configuration cheap in terms of the number of neurons required, it also avoids wasting other potential synaptic sites. The rest of the dendritic tree is free to implement other computations, since these will not interfere with the original XOR. In the “classical” configuration (on the left-hand side of Fig. 2.4), other potential computations (at other dendritic sites: not depicted) will interfere with the target computation since the XOR is borne out of the whole network.

### 2.4.5 Biological Neural Correspondence Distributions

How does this emerging view regarding the computational richness of single neurons relate to our approach? It turns out that the fact that dendrites exhibit complex non-linearities and thus computations is very advantageous. Recall that our approach relies on a large set of correspondences, and thus that we need to represent a large number of second order terms (or conjunctions). Clearly it would be very wasteful for biological systems to assign a single neuron to each conjunction. Fortunately, mounting evidence suggests that in many cases dendrites exhibit sigmoidal non-linearities (e.g. see Polsky et al. 2004), which in turn can be used for computing conjunctions. This signifies a huge saving of resources since a single neuron can be used for representing hundreds or thousands of correspondences, rather than just one as would be required by the classical view. It is interesting to note that the two main computations in our approach (conjunctions for correspondences and summation for vote counting) can be implemented in a single neuron, e.g: conjunctions can be computed by dendritic branches, the results of which can be summed linearly at proximal parts (relative to the soma) of the dendritic tree.

Note that the modeling of biological neural systems can be done at many different levels, e.g: molecules, synapses, cellular compartments, neurons, networks, dynamics, maps and systems (see Churchland & Sejnowski 1992). The choice of what to

consider essential, and thus keep, and what to consider inessential, and thus leave out, depends very much on the type of argument that is being put forth. We are concerned here mostly with patterns of connections and logical operations, rather than molecular, electrophysiological or even compartmental issues. This serves our purposes best, since our argument is essentially an algorithmic one, i.e: what types of algorithms (and thus what types of computations) are biological neural systems using for estimating poses?

## **2.5 Formation**

Now that we have dealt with the computational, algorithmic and implementational levels, we are ready to introduce the formational level. Examples of concerns characteristic of this level include: 1) the nature of formational mechanisms, 2) the robustness of the formational process to environmental factors, 3) the quality of resulting devices, 4) temporal efficiency and 5) the ability to scale-up. In this thesis, we focus mainly on the first three issues.

### **2.5.1 Types of Formation**

As already mentioned, formation pertains to the processes by which particular computational devices, implementing particular algorithms, which in turn establish particular computations, come to be.

#### **Artificial Domain**

A possibly interesting example from the artificial domain pertains to nanotechnology. It is an interesting example in the sense that in many cases, the computational devices are already theoretically defined and the necessary materials already exist, and yet problems remain in terms of the assembly of those materials, e.g: supramolecular chemistry for self-assembly. Although with less severity, formational problems are also encountered in the manufacture of compact circuit boards and other components for modern personal computers and other devices.

## Biological Domain

In the biological domain, formation pertains to development. Since our context is computational, we are here concerned with biological neural development. Topics in this area include: the growth, differentiation, placement and selective death of neurons, the guidance of axons and dendrites, and the formation, maturation and elimination of synapses. In our thesis, we will be concerned mainly with axonic guidance and synaptic formation, maturation and elimination.

The development of biological neural systems is extremely complex. Hundreds of different types of cells must grow, differentiate and migrate to their correct locations. To makes things even harder, neurons also have to be interconnected in extremely complex and highly unique ways. For example, axons must grow not only towards particular brain regions, but also to particular sub-regions, layers, networks, cells and even to specific parts of cells (e.g. a sub-region in a dendritic tree). The resulting architectures are so complex that it is impossible for DNA to explicitly code for all of it (see Churchland & Sejnowski 1992).

Clearly indirect genetic guidance through molecular gradients and other signalling mechanisms is a fundamental driving force behind the formation of biological neural systems. However, interacting with this biochemical guidance, is another force: activity dependent development. Neuronal firing dynamics can be harnessed in useful ways for the establishment of different patterns of connectivity (e.g. see Wong 1999 for an example of how spontaneous retinal firing patterns can guide the development of visual pathways even before the onset of vision). A crucial sub-type of activity dependent development is that based on environmental stimulation of sensory organs, i.e: experience dependent development (e.g. see Miller 1994). Our thesis provides an example of how experience dependent development can lead to complex and useful pose estimation neural architectures.

### 2.5.2 Development of Correspondence Distributions

How can neural architectures embodying correspondence distributions capable of estimating accurate poses develop? Not only must map neurons project to the

correct vote/pose neurons but particular synapses must pair up (in small dendritic sub-regions such as terminal branches) with their correct counterparts to form conjunctions. It turns out that “dumb” (random) axonic guidance is sufficient for generating the correct connectivity providing that certain Hebbian-like mechanisms are in place, and visual stimulation is available. We hypothesize a variable describing the stability of pairs of synapses which, in general, is increased by correlated activity (between synaptic pairs and target cells) and decreased by decorrelated activity. Synaptic pairs with a low stability are eventually eliminated and substituted by a new random pair. Gradually, the process of strengthening, weakening and substituting synaptic pairs, leads to the emergence of architectures that reflect, and which can thus estimate, the transformations being “experienced” by the network.

This chapter has overviewed the four levels of abstraction spanned by the thesis: computational, algorithmic, implementational and formational. It has provided some background information and made brief but relevant connections with the thesis in order to better clarify its contents, motives and structure. In the following chapter we will survey several algorithmic approaches for solving the problem of pose estimation and justify our chosen approach.

# Chapter 3

## Algorithms for Pose Estimation

### 3.1 Introduction

In general, computational problems can be solved by more than one type of algorithm. The problem of pose estimation is no exception. In this chapter, we aim to cover a broad spectrum of approaches applicable to pose estimation. Since, our underlying motivation is to investigate an efficient, neurally implementable and biologically plausible approach, our most likely source of algorithms lies in Computational Neuroscience and related disciplines. Surprisingly, and a few exceptions aside, the literature is scant in explicit and general models of pose estimation. For example, although the literature abounds in models of attentive visual search (see for example Chelazzi 1998, Chelazzi et al. 1993 and the review by Müller & Krummenacher 2006), these are only implicitly and non-generally performing pose estimation, i.e: the identification of the location of an object is indirectly producing a translation estimate, which however ignores other pose attributes such as orientation and scale.

This scantness of pose estimation models forced us to search wider. Most of the algorithms presented in this chapter, originate from theoretical and/or applied domains, such as Computer Vision. One of the challenges in this context, is to compare algorithms in terms of how they might fare in a particular implementational domain (i.e. biological neural networks) that is completely different to the one they were originally designed for (e.g. serial computers).

Pose estimation algorithms can be categorized along different dimensions, e.g: 1) how estimates are computed, 2) whether local, regional or global information is used and 3) the transformations which are addressed (e.g. rigid, similarity, affine and projective). We have decided to classify algorithms according to whether they use templates, correspondences or global information for computing estimates. This classification, in our view, encompasses the vast majority of approaches, exhibits limited overlap and facilitates making evaluations regarding neural-implementability.

The rest of the chapter is organized as follows: first template based approaches are discussed, followed by correspondence-based approaches and then global approaches. In the conclusion section we summarise the algorithm evaluations and select an approach for further investigation in a neural context.

## **3.2 Template based approaches**

In general, template based approaches involve an iterative process, whereby a template (or a target pattern) is continuously transformed and compared to a target pattern (or a template), until a sufficient level of correlation has been achieved. Combining the transformations up to the final match provides the transformation (or pose) that relates the template and the target pattern.

### **3.2.1 Deformable Models**

Deformable models are a very interesting approach applicable to segmentation, shape matching, motion tracking, pose estimation and other domains (e.g. Jain et al. 1998, Cheung et al. 2002 and Yena & Smith 2005). One area that is greatly benefiting from this research is the segmentation of medical images (see McInerney & Terzopoulos 1996). Anatomical structures vary greatly in location and shape and thus call for models which can deform whilst maintaining their intrinsic shape. By deforming within certain limitations, the models can latch onto these structures thus providing adequate segmentation and identification. Apart from their flexibility (they can deal with shape variations) and generality (they are applicable to several domains), deformable models have the added advantages that they are



significantly accurate and allow for the interaction of bottom-up constraints and top-down knowledge.

Although deformable models are interesting in their own right, we are here concerned not so much with the fact that models can deform, but instead on what guides the deformation process. The implicit pose estimation algorithm of deformable models lies in what guides the deformation, since it is the deformation process that compensates for the transformations that distinguish the model (in its prototypical state) and an actual instance of the structure which the model represents. In other words, the guiding force behind the deformation, seeks to apply the transformation that relates the model to the instantiation of the model. Although one might envision different types of guiding forces underlying model deformation, according to (McInerney & Terzopoulos, 1996) the most popular method comes from Approximation Theory. In this approach to deformation, the goal is to minimize an energy function consisting of a term representing internal forces (tension and rigidity) for maintaining the model's intrinsic shape, and another term representing external forces which attract models to salient features. The minimization of the energy function is most usually performed through numerical algorithms.

The main advantages of the approach include generality (e.g. it is extensible to any combination of transformations), and robustness (e.g. it can deal with distortions and noise<sup>1</sup>). On the downside, the approach is relatively inefficient in the sense that it depends on an iterative process for convergence and is likely to require an excessively convoluted architecture to be neurally implemented.

### 3.2.2 Map-Seeking Circuits

The map-seeking circuits approach is another interesting template-based approach. It was developed by an independent researcher, David W. Arathorn, and is documented in some detail in (Arathorn, 2002). In the author's view, its main advantage lies in its ability to eliminate the transformational combinatorial explo-

---

<sup>1</sup>Distortion robustness is made possible due to the fact that models are by definition deformable and noise robustness is derived from the fact that the approach, in general, maximizes the goal of maintaining the internal structure of models.

sion which is commonly faced by template-based approaches. Instead of having to deal with  $\prod_i n(T_i)$  transformations it only needs to deal with  $\sum_i n(T_i)$  where  $T_i$  stands for different types of transformations (e.g. scaling and rotation) and  $n(T_i)$  stands for the number of possible configurations within transformation  $T_i$  (e.g. if 10 scaling factors are allowed  $n(Scaling) = 10$ ). The combinatorial explosion is essentially eliminated by “pattern superpositions” and a recurrent/iterative process that manages to discover/recognize the patterns being presented and/or their poses.

By pattern superposition is meant the actual summation of patterns. It is quite interesting to see how, through the recurrent/iterative map-seeking process, an apparently homogeneous amalgam of patterns can be resolved into a very precise pattern. Refer to Fig. 3.1 for a simplified depiction of a map-seeking circuit in the context of rigid transformations. Notice the translation and rotation superpositions and notice how the competitive process recovers both the identity of the pattern and its transformations. Unfortunately, in the approach’s main strength (pattern superpositions) probably lies its main weakness, since there is a limit to how many patterns can be superposed, which depends on their nature, and which if exceeded, can lead to errors termed collusions. Furthermore, at this point, there seems to be little direct evidence for pattern superpositioning as a coding strategy used in biological neural systems (D.W. Arathorn, personal communication, 1 July 2005), although one must be reminded that “absence of evidence is not evidence of absence”.

For lack of space, the recurrent/iterative process can not be explained here in great detail. There are however some general principles that seem to cover its main functionality. First of all, transformations are segregated into different layers/stages. The memory layer/stage is, as might be expected, the last one of the forward processing chain. Secondly, the main computations being performed between adjacent layers are the summing (superpositions) and matching of patterns. Patterns are superimposed in order to break the transformational combinatorial explosion and patterns/poses are gradually extracted/discovered by matching. Patterns that match more than others are given stronger coefficients in their superpositions and gradually

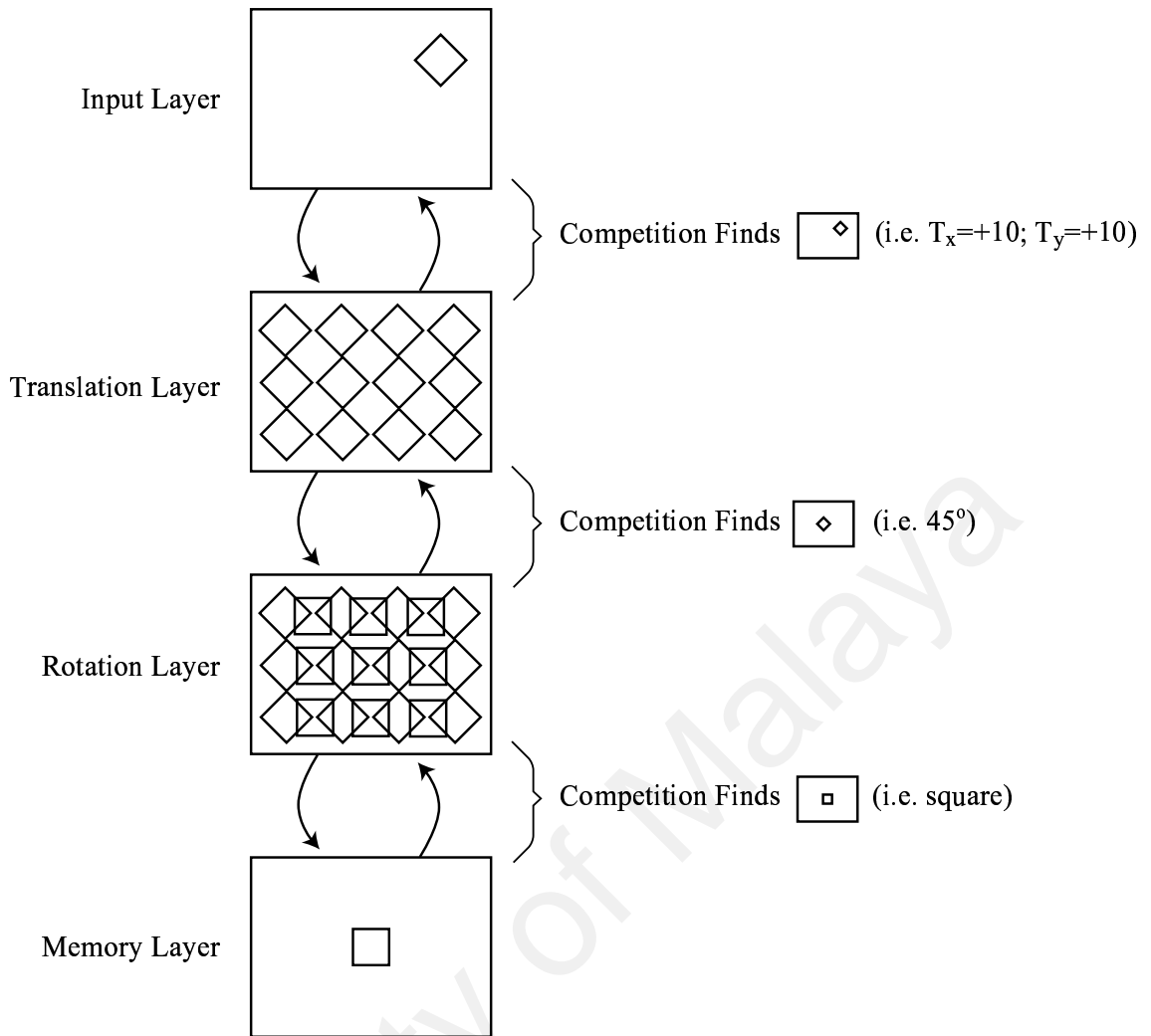


Figure 3.1: Simplified diagram of a map-seeking circuit.

get extracted.

Apart from eliminating transformational combinatorial explosions, the approach has other strong points: 1) generality (map-seeking circuits can be applied to different areas of vision and even to entirely different modalities and/or computational problems), 2) biological relevance (in Arathorn 2002 the author has gone to some length to show how the pure algorithmic side of the approach can be implemented by neural-like circuits) and 3) robustness (see Arathorn 2004). Probably the main downside of the approach, from our perspective, is its iterative nature.

### 3.2.3 Dynamic Routing Circuits

The Dynamic Routing Circuits (DRC) solution (see Olshausen et al. 1993 and Anderson et al. 2004) to the problem of information routing and visual invariance is

highly relevant to our discussion. It is a well known fact that brains utilize attention bottle-necks in order to filter out the majority of sensory information and to focus limited processing resources on interesting/useful regions of the input space. In the visual system, there is the additional problem of recognizing objects regardless of their particular transformations (e.g. position and size). The DRC model solves both problems (attention and visual invariance) in one swoop, e.g: by changing the position and size of an attention window in order to focus on a shifted/scaled object, both problems are being solved. Although the model was originally developed to deal with only translation and scale, it can be generalized to other transformations (Olshausen et al. 1993).

One of the most interesting aspects of DRC is how information routing is implemented. Briefly, this is achieved via a set of control neurons which synapse multiplicatively onto the connections that emerge from the inputs and which ultimately converge onto output units where the focused processing (e.g. object recognition) is to be performed. By enabling some connections and disabling others, the control units effectively route (shift or scale) information from a limited region of interest in the input space (e.g. retina) onto the output nodes. Although the actual mechanism of routing information by enabling/disabling connections is of great interest, what we are here interested in is “what controls the control units”. Since the control units shift attention to shifted and/or scaled objects, and since we are interested in pose estimation, we need to know what guides the control units into enabling/disabling connections.

In (Olshausen et al., 1993) the authors describe an autonomous control mechanism whereby attention is guided by Gaussian salience blobs. Essentially, each control unit represents a different transformation (i.e. each control unit, if activated, implements a particular routing of information, which corresponds to a particular transformation) and competes with every other control-unit for its particular routing to be effected. The competition is essentially an optimization process, whereby the goal is to maximize the correlation between the routed information and a template Gaussian blob. Thus, the control unit whose routing leads to the maximal corre-

lation between the template and the routed information is the winning node, and the one that ultimately performs the routing. Thus one might say that the essential algorithm being implemented is a form of template matching with a search through transformation space. Although optimization (or search) processes are by no means biologically implausible, in this case, they bring with them a temporal disadvantage which we wish to avoid. In spite of this, the majority of insights provided by the DRC research (e.g. information routing mechanisms) remain highly relevant and compatible with our chosen approach for the estimation of transformations. The strengths of the approach include: parallelizability, generality and robustness (see Olshausen et al. 1993).

### 3.2.4 What-and-Where Filter

The What-and-Where filter (see Carpenter et al. 1998) is an interesting approach originating from a biologically motivated context. The What-and-Where filter implements object recognition (i.e. “what”) and pose estimation (i.e. “where” refers not only to position, but also to size and orientation) in parallel and interactively.

In this particular context, we are more interested in the “where” aspect of the approach, i.e: how are poses estimated? The approach relies on a multitude of oblong excitatory regions (or oval blobs), at different positions, and with different sizes and orientations, each one of which is convolved with a particular region of the target image. A competitive process serves to isolate the particular filter which is maximally activated (and thus correlated) by the image, and thus which indicates the presence of a particular object with a particular position, size and orientation. In spite of the discriminatory poorness of the “templates” (i.e. oblong excitatory regions), the approach is nevertheless quite successful at locating target objects. Seeing that the filters are globally matched to image regions, a critical disadvantage of the approach lies in its lack of robustness in terms of cluttered environments (occlusions and distractors).

## 3.3 Correspondence based approaches

Correspondence based approaches have here been divided into two main categories, i.e: search-based and vote-based approaches. In a search-based approach a pose estimate is generally formed after an iterative search for a transformation that can bring a sufficient number of model points into close proximity to a sufficient number of image/data points. In a vote-based approach a pose estimate is formed on the basis of a one-shot view of the evidence at hand, where each correspondence (or subset of correspondences) represents a vote of some kind, usually in favor of a particular transformation (or subset of transformations).

### 3.3.1 Search-based

#### Generate-and-Test

Random sample consensus (RANSAC) is probably one of the most popular generate-and-test approaches. The technique is quite general being intended for the robust fitting of models (see Fischler & Bolles 1981). It can be applied to domains as varied as: shape recognition, pose estimation, feature matching and others. When RANSAC is applied to pose estimation problems, it appears almost undistinguishable from Ullman's alignment method (see Ullman 1996). Note that Ullman's alignment method allows for the incorporation of feature labeling, which reduces the amount of search required.

When RANSAC is given a set of data points, it selects a random sample as small as possible but large enough to constrain a model hypothesis (e.g. a circle can be constrained by three points). It then verifies this model against the data set by for example counting the number of points that lie on the positions defined by the model. If the number of inliers is below some threshold, RANSAC chooses another random sample of points to form a model hypothesis and performs verification again, and so on, until several or no models have been confirmed to be present in the data.

As already mentioned, in the case of circle detection, three points are sufficient to form model hypotheses. In the case of estimating a similarity transformation

between two sets of points, a sample of two points (each one a correspondence) is required. As in the alignment method, the verification step of a pose estimating RANSAC algorithm, involves applying the model hypothesis to the source points and then verifying how many of the target points coincide with the transformed points. If there is a sufficiently large number of coincident points, then the model hypothesis (a particular similarity transformation) is deemed to be correct.

The main strengths of RANSAC-based approaches are probably their robustness to outliers (e.g. spurious points), their permissibility of noise in the characterization of data points (e.g. positional noise of image points) and their generality. Their main disadvantages include difficult parallelization and the fact that they are iterative in nature (see Fischler & Bolles 1981).

## **RAST**

Thomas M. Breuel developed the Recognition by Adaptive Subdivisions of Transformation Space (RAST) algorithm which is a very interesting approach that combines ideas from the following approaches: Hough transform, multi-resolution matching, search-based recognition and bounded-error recognition.

Breuel's main concern in (Breuel, 1992b) is object recognition and defines his problem in the following terms: "Find a transformation (e.g. among all translations) that will map as many points of the model to within error bounds of image points.". Two observations should be made regarding this definition. First, there is no mention of the actual/true transformation between image and model points: the quality of the transformation is indirectly measured by how well model points are matched to image points. Secondly, one of the main advantages of the approach is included in the problem definition: error bounds around image points (note that Baird in (Baird, 1985) was one of the first authors to explore bounded error techniques in the context of pose estimation).

A correspondence between a model and an image point actually spurns a set of transformations corresponding to all the transformations that will map the model point to all the points within the error bounds of the image point. More importantly,

in certain conditions, if the error bounds around the image point are defined by a convex polygon, then as Breuel and others before him have proven (Baird, 1985), the corresponding set of transformations in transformation space defines a convex polyhedron of “feasible transformations”: constraint polyhedron.

The way evidence for transformations is then derived is not very different from most Hough-based approaches, about which more will be said in the following subsections. The main idea is that correspondences that are compatible (i.e. that have been generated from the same transformation), will create convex polyhedra in transformation space that intersect each other. Intuitively, a transformation that matches more model points to image points, leads to more intersecting polyhedra in a particular region of transformation space and thus one is more confident that that region corresponds to the transformation we are seeking.

One of the main ideas that sets RAST apart from most other Hough-based algorithms is that it does not represent the whole transformation space in one large fixed-resolution data-structure. What it does instead is rely on a query-box (see Breuel 1992a) that adaptively changes size and location in the transformation space in order to find the configuration with the most intersections with constraint polyhedra.<sup>2</sup> The main idea on which the approach hinges is that the number of intersections between the query-box and the constraint polyhedra defines an upper-bound on the number of intersections between the polyhedra themselves. Therefore if the upper-bound of some query-box is too small, we can automatically reject the region of transformation space it corresponds to, since the number of intersecting constraint polyhedra can never exceed that bound. Furthermore, if the region of transformation space encompassed by the query-box is significantly small then one’s confidence that the number of intersecting constraint polyhedra is equivalent to the the number of polyhedra intersecting the query-box increases.

So the algorithm is broadly defined as follows. First a query-box is generated that encompasses the region of transformation space that we want to consider. Then the number of constraint polyhedra intersecting the box is computed<sup>3</sup>. If the number of

---

<sup>2</sup>Cass in (Cass, 1993) provides another example of a correspondence-based approach that involves a search through transformation-space.

<sup>3</sup>Note that finding a set of “constraint polyhedra” is equivalent to finding a set of correspon-



intersections (the upper-bound) is below a certain minimum level, then this region of transformation space is abandoned and the search is resumed in a different region. If, on the other hand, the upper-bound is larger than the minimum level, then the query-box is recursively sub-divided into smaller query-boxes within the original one, until a certain minimum box size (or maximum search depth) has been reached.

The main advantages of RAST include: small memory demands, adaptive resolutions, bounded-errors, flexibility regarding evaluation functions, generality and robustness. Its main disadvantages pertain to neural implementability: the search procedure embodied in the adaptive query box is unlikely to have an elegant and fully parallel neural implementation. RAST's iterative nature is also considered a disadvantage in this context.

### **3.3.2 Vote-based**

Although vote-based approaches do exhibit certain weaknesses in comparison to some of the sophisticated search-based approaches (e.g. Breuel 1992*b*) such as memory-requirements and resolution issues, their advantages regarding neurobiological implementability far outweigh their disadvantages. Furthermore, many search-based methods, such as those found in the Point Pattern Matching literature, assume unlabeled points, which in most cases makes the problem of pose-estimation harder than it really has to be. If one assumes that local invariant features with some saliency and discriminatory power are available (there seems to be ample evidence that such features are abundant in biological neural systems), then vote-based approaches become quite accurate and useful. Vote-based approaches combine the largest number of strengths from the perspectives of neural implementability and performance.

#### **Standard Hough Transform**

The Standard Hough Transform, is probably one of the earliest vote-based methods for shape detection and pose estimation. It was originally developed for the

---

dences.

detection of lines and curves (see Hough 1962). Pose estimation is implicit to shape detection, e.g: during detection of a circle the algorithm simultaneously obtains its position and scale.

Probably the three main elements of any Hough transform are: 1) parameter space, 2) votes, 3) accumulator array (voting data structure) and 4) the analysis of peaks in the accumulator array. Problems are usually defined in terms of parameter spaces. The problem of circle detection, for example, can be defined in terms of three parameters: horizontal and vertical position and radius. Conversely, a solution consists of one or more small regions in that parameter space. Votes are usually provided by local aspects of the data (e.g. local image features), which in turn increase the probability that the sought-for solution is contained in a particular sub-region of the parameter space. The accumulator array provides a representation of the parameter space, which is discretized due to computational and memory limitations. Peaks in the accumulator array represent regions of the parameter space for which the data has significantly voted, and which are likely to contain valid solutions (e.g. shape detections).

Assuming we want to detect circles of variable radii in some image, the Standard Hough Transform might be implemented in the following manner: 1) create (and initialize to zeros) a 3D accumulator array where 2D positions and circle radii are represented, 2) extract salient image features, 3) for each salient feature, compute consistent circle-center positions and radii, 4) increment the accumulator cells corresponding to each computed circle-center and radius, 5) after going through all points analyze the accumulator array for peaks. Peaks identify the likely locations and scales of circles.

Some of the main problems of the Standard Hough Transform and other derived/related approaches include: 1) memory demands, 2) computational expense, 3) fixed resolutions and 4) false positives. Excessive memory and computational demands stem mostly from the use of accumulator arrays to represent large parameter spaces. When a single accumulator array is used, the resolution of the parameter space is fixed, which can lead to problems if finer-grained estimates are

required. The analysis of peaks in the accumulator array is not always a straightforward process leading occasionally to the detection of accidental peaks (i.e. false positives). Many of these problems have since been addressed (refer to the following subsections). Some of the great advantages of Hough-based approaches include their robustness to noise, spurious features and occlusions, and their ability to deal with multiple solutions simultaneously. Furthermore, the notion of a vote is extremely simple, easily represented and amenable to parallel implementations, which makes Hough-based approaches quite promising from a neural perspective.

### Generalized Hough Transform

The Generalized Hough Transform (GHT) grew out of the Hough Transform (see Hough 1959) in order to deal with arbitrary shapes (see Ballard 1981).

Following the example of (Hecker & Bolle, 1994), and for the sake of clarity, we will first describe how the GHT can be used to detect arbitrary shapes regardless of translation, and then we will describe how scale and rotation can be incorporated. The algorithm essentially consists of an off-line stage for model preprocessing and a subsequent on-line stage for shape and pose detection.

During the off-line stage, each model generates an R-table. The R-table is generated by selecting an arbitrary point  $p_0$  inside the model boundary, and then going through prominent points (i.e.  $p_n$ ) in the model boundary, where each point generates an index to the table ( $\theta(p_n) \rightarrow (p_0 - p_n)$ ), where  $\theta(p_n)$  is the angle of the tangent to the boundary at point  $p_n$  and  $(p_0 - p_n)$  is the vector formed between the boundary point and the center point  $p_0$ .<sup>4</sup>

During the online-stage, an accumulator (or voting) array is first generated to represent estimated  $p_0$  positions. Subsequently each point in the scene (i.e.  $k_n$ ) is used for estimating  $p_0$  positions. The orientation at each point is computed and used for indexing the R-table, which returns a  $\vec{r} = (p_0 - p_n)$  vector which is added to the position of the current scene-point in order to obtain a  $p_0$  estimate (i.e.  $p_0 = k_n + \vec{r}$ ), which determines the accumulator cell to increment. At the end, the accumulator

---

<sup>4</sup>Note that each  $\theta(p_n)$  can have more than one  $(p_0 - p_n)$  vector since, in most cases, shapes have multiple instantiations of a particular tangent angle.

array is analyzed for peaks, which should correspond to positions in the scene where the model is present.

In order to allow for rotated and scaled instantiations of models, the above algorithm needs to be extended at the online-stage by first generating a  $4D$  accumulator array representing position, rotation and scale (i.e.  $[p_0, \alpha, s]$ ). At each scene point (i.e.  $k_n$ ), the algorithm now loops through the allowed rotations (i.e.  $\alpha$ ) and scales (i.e.  $s$ ). The R-table is indexed with  $\theta(k_n) - \alpha$ , which returns a vector  $\vec{b}_0$  which needs to be re-scaled by  $s$  and rotated by  $\alpha$ , before it can be added to  $k_n$  in order to form an estimate of  $[p_0, \alpha, s]$  (i.e.  $p_0 = k_n + \vec{r}$  where  $|\vec{r}| = s \cdot |\vec{b}_0|$  and  $\theta(\vec{r}) = \theta(\vec{b}_0) + \alpha$ ).

Note that, although the GHT was devised with shape detection in mind, it is intertwined with and automatically generates pose estimates.

### Adaptive Hough Transform

The Adaptive Hough Transform (AHT) attenuates the memory, computational and resolution problems of the Standard Hough Transform (e.g. see Illingworth & Kittler 1987 and Tian & Shah 1997). It accomplishes this by using smaller accumulator arrays, and multiple processing stages, where earlier stages use coarser representations of parameter space and later stages use finer representations of parameter space. The results of each stage feed into the subsequent stage, e.g: if at stage one the coarse region R3 receives a significant number of votes, then at stage two the algorithm adaptively focuses on R3 by creating a finer parameter space within the confines of R3. The process continues until all significant peaks have been investigated up to an acceptable degree of precision/resolution. Other related approaches, which use coarse-to-fine parameter resolutions, include: the Fast Hough Transform (Li et al., 1986) and the Hierarchical Hough Transform (Espinosa & Perkowski, 1991).

### Hough Transform Network

The Hough Transform Network (see Basak & Pal 1999) is an interesting approach with special relevance to the current thesis.

The basic idea in Hough Transform Networks is to represent the transform parameters in the connection weights of an artificial neural network. The parameter values are gradually learnt so there is no need to represent the whole space of parameters (as in a conventional fully-defined voting structure). This leads to two significant advantages: firstly storage requirements are smaller than in traditional Hough Transforms and secondly the resolution of parameters is not fixed thus allowing for, in principle, results of any precision.

In the context of this thesis, the architecture presented in (Basak & Pal, 1999) does however present us with a crucial weakness, regarding biological plausibility. The concept of Hough Transform Networks is illustrated in (Basak & Pal, 1999) using a line detection problem. In order for the network to learn the parameters defining several lines in an image, it has to be presented with the points comprising those lines, in a sequential manner. This sequential presentation of points is unlikely to be realistic in biological systems, which have the resources and the need to maximally exploit parallelness.

### **Geometric Hashing**

Just as with the Generalized Hough Transform, the class of algorithms referred to as Geometric Hashing (e.g. see Lamdan & Wolfson 1988), involves an off-line stage for model pre-processing and an on-line stage for recognition. Furthermore, models and their instantiations in scenes are represented by local features or interest-points.

Early but representative Geometric Hashing algorithms were designed for the recognition of 3D rigid objects in cluttered scenes under affine transformations. Two important theoretical points on which the algorithm rests are: 1) a unique affine transformation maps a set of three non-collinear points onto another set of three non-collinear points and 2) three non-collinear points define two linearly-independent vectors which in turn define a 2D coordinate system in which any remaining points can be represented.

Assuming multiple models, the off-line stage of the basic algorithm goes as follows: 1) for each model extract all  $n$  interest points, 2) for each set of 3 non-collinear

points, define a coordinate-system in which all other  $n - 3$  points are re-defined, 3) use each re-defined coordinate to index a hash table and record the model and triplet which originated the coordinate.

In the on-line stage the algorithm works as follows: 1) extract the scene's  $n$  interest points, 2) select an arbitrary set of 3 non-collinear points, 3) redefine the coordinates of the remaining  $n - 3$  points according to the triplet's coordinate-system, 4) use each re-defined coordinate to index the hash-table, and vote for the model-triplet pairs there recorded. In the end, if there is a model-triplet pair with a sufficient number of votes, then this is evidence for a model instantiation at the position defined by the initial/scene triplet. Furthermore, the affine relationship that relates the initial/scene triplet and the winning triplet, defines the transformation between the model and its instantiation.

One of the more significant advantages of Geometric Hashing lies in its efficiency (specially in the context of serial computers), however, its utilization of multiple correspondences (in the above example, triplets were used) is most probably too expensive for most neural implementations.

## **RUDR**

Recognition using Decomposition and Randomization (RUDR) is an interesting approach that combines search and voting, in an attempt to avoid the weaknesses of both (e.g. see Olson 2001).

In brief, RUDR decomposes a problem into multiple sub-problems each one of which is randomly selected (this is the search part) for further analysis using a Hough-based method (this is the voting part). The meaning of sub-problem will be clarified next by using the estimation of a similarity transformation as an example. If a match is defined as a sub-set of model to image point correspondences and the cardinality of a match is defined as the number of correspondences involved (see Olson 2001), then one can say that the constraining match-cardinality of a similarity transformation is 2, since two correspondences provide four equations which can constrain the four unknowns of a similarity transformation (i.e. 2D position,

scale and rotation). If the constraining cardinality is  $k$  (in this case  $k = 2$ ) then a sub-problem in this context corresponds to a match with cardinality  $g < k$  (in this case  $g = 1$ ). Any single correspondence partially constrains the parameter-space and thus represents a sub-problem in the sense that it partially (but incompletely) solves the problem. After a random correspondence (subproblem) has been selected, voting is performed by combining the selected correspondence with remaining correspondences in order to form constrained estimates. If the initial correspondence (sub-problem) is false, then the voting stage is unlikely to form significant peaks. Conversely, if the initial correspondence is true, the voting stage should produce a significant peak with the correct estimate. Note that in general  $g$  and  $k - g$  can both be larger than one.

In spite of RUDR's advantages (i.e. generality, accuracy and computational/storage efficiency), sub-problems (and implicit cardinalities larger than one) and randomized search, are likely to be representationally unfeasible in a neural context. Regarding the first impracticality, parallel computation in general calls for a complete representation of all sub-problems which by definition contradicts the solution. If on the other hand sub-problems are represented serially, then some very complex and unfeasible machinery is required in order to change the underlying representation for each serially tested sub-problem. Regarding the second impracticality, that is, randomized search, again unfeasible machinery is required for random selection, serial testing, and dynamic termination of sub-problems.

## 3.4 Global Properties

The following approaches use neither template matching nor correspondences, instead they rely on objects' global characteristics for generating pose estimates.

### 3.4.1 Center of Mass

The center of mass approach uses global information for computing an object's "center". In one of its simplest forms the technique averages the positions of an object's constituent points. Variations might perform an averaging which is weighted

by point characteristics such as brightness or entropy. Regardless of variations, the core idea of the technique is a normalization relative to the object's "center", which essentially constitutes a translation estimate. One of the advantages of the approach is that it can be used as a fast preliminary stage for a more complex pose estimate (see Hong & Tan 1988 for an example of where this is done). In the case of a similarity transformation, an initial center of mass normalization can provide the translation estimate, allowing the subsequent computational stage to concentrate only on the target's scale and orientation. The main disadvantages of the approach include lack of robustness (i.e. it exhibits sensitivity to noise, occlusions and spurious points) and lack of generality (i.e. it is not easily extended to more complex transformations). The main strength of the approach lies in its neural implementability (refer to the thesis' Discussion chapter for a brief explanation and illustration of the neural implementability of the approach).

### **3.4.2 Symmetry Axes**

Another global approach related to the center of mass technique involves detecting an object's symmetry axis. Obviously the technique is not general in that it is applicable only to symmetric objects (e.g. human body). By finding the symmetry axis of an object and computing its angle, a rotation estimate is generated which relates the object in a canonical configuration and its image instantiation. As in the center of mass case, the main weaknesses of the symmetry-axis approach include lack of generality and robustness. Neural implementability is one of the strong points of the approach.

If the center-of-mass and symmetry-axes techniques are combined with an accurate human-head detector, the result is a similarity transformation estimator for human bodies consisting mostly of global methods: 1) the center of mass technique provides the translation estimation, 2) the symmetry-axis technique provides the rotation estimate and 3) the distance between the center of mass and the head (from the head detector) provides the scale estimate.



Table 3.1: A qualitative comparison of pose estimation approaches.

Approach	Algorithm	P	A	G	E	R	Score
Template	Deformable	1	0	1	0	1	3
	Map-Seeking	1	1	1	0	1	4
	DRC	1	1	1	0	1	4
	What-and-Where	1	1	1	1	0	4
Correspondence	Search	0	0	1	0	1	2
	Vote	1	1	1	1	1	5
	Search & Vote	0	0	1	0	1	2
Global		1	1	0	1	0	3

### 3.5 Conclusion

Table 3.1 summarises how we have evaluated the main approaches in the literature. Note that the evaluations must be seen in the context of our main objectives: neural implementability and performance. How these objectives are defined moreover, can vary from author to author. We have chosen to focus neural implementability on parallelizability and architectural simplicity. Regarding performance, we have chosen to focus on generalization, efficiency and robustness (overall accuracy is assumed). Furthermore, for the sake of simplicity and combining evaluations, we have adopted a qualitative approach which reduces each evaluation to a binary variable (one for “sufficient” and zero for “insufficient”). Note also that only differentiable categories have been represented (e.g. the vast number of Hough-based approaches have been grouped into the single “vote” category since the former are mostly indistinguishable in terms of the main evaluation factors being used).

The following abbreviations have been used in Table 3.1: P (parallelizability), A (architectural simplicity), G (generalization), E (efficiency) and R (robustness). It is important to note that evaluations are relative. For example, both map-seeking and DRC approaches have been evaluated as “insufficiently” efficient. This does not mean that they are inefficient, instead this reflects the fact that they require several recurrent iterations before finding a solution, whereas other approaches such as the voting ones can find a solution in one step (assuming parallel implementations).

According to the taxonomy, objectives, and evaluations embodied in Table 3.1, the most promising approach (with a total score of 5) for a neural investigation

involves correspondences and votes. Since correspondences are easy to represent using conjunctions, and vote-accumulation can be computed by one feed-forward layer, the approach is parallelizable, efficient and architecturally simple. It is also general (i.e. it can be extended to many complex transformations and other domains) and robust. Additionally, accuracy can always be further improved by simply increasing the discriminatory power of local invariant features.

Note that the map-seeking and DRC approaches are close behind, differing only in terms of efficiency, which some authors might choose to ignore, contrary to our present decision. The What-and-Where approach, on the other hand, falls short of the maximum score mainly due to its lack of robustness in cluttered environments. Note also that the global approaches, although still further behind on account of their lack of generalization and robustness, nevertheless exhibit maximum scoring in terms of neural implementability, i.e: they can be implemented in simple, fully-parallel and efficient neural architectures. Regarding vote-based approaches, we will be focusing on the simplest (and still accurate) forms, since this will permit us to fully explore the broader argument which includes an analysis of biological implementability, structural optimization and developmental algorithms.

In this algorithmic chapter (see Fig. 2.1) we have described and discussed some of the main pose estimation approaches based on templates, correspondences and global properties. Several criteria, such as neural implementability, were used for selecting the best candidate for further investigation, i.e: approaches based on correspondence votes. In the following chapter we will describe the chosen approach in sufficient detail for its applicability to real problems.

# Chapter 4

## Single Correspondence Analysis

### 4.1 Basic Concepts

As a useful reference for the rest of the thesis, and for the sake of a rapid contextualization, we will start by briefly explaining some of the most fundamental concepts.

#### 4.1.1 Inputs

**Input-Clouds.** The current thesis assumes that, in most cases, the amount of visual information received by a sensor exceeds processing resources, thus calling for an attentional bottleneck, i.e: a window which captures a sub-set of the available visual information (see Ullman & Koch 1999). Our attentional window consists of a finite set of semi-independent inputs, each holding a specific image position, and altogether forming what is here termed an input-cloud. The function of an input is to convey image information that is local to its position: in the simplest case each input transmits the brightness value underlying its position (see Fig. 4.1). Input-clouds can exhibit global dynamics, where the positions of inputs are changed according to a global rule (e.g. right-shift all inputs by 4 pixels), or local dynamics, where each input changes position semi-independently of all other inputs. The existence of multiple, parallel and interacting clouds poses many interesting questions, which unfortunately lie outside the scope of the current thesis. Refer to Appendix C for a more detailed discussion on input-clouds (or dynamic-inputs) and their general

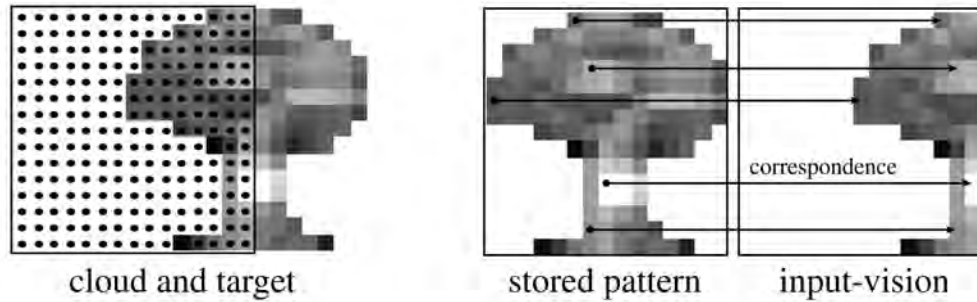


Figure 4.1: An input-cloud, an input-vision and some correspondences.

practicality.

**Input-Vision.** An input-vision consists of the totality of the information (e.g. brightness values) gathered, in a topographic fashion, by all inputs. In most cases, this information is gathered in a regular array format, which is easier to interpret (see the right-hand side of Fig. 4.1).

Constraining input-clouds to having a pre-determined (user-defined) fixed number of inputs, simplifies matters, e.g: there is no need to process information for subsequent stages and sampling is performed automatically. However, the scheme fails when clouds are significantly scaled-up and test-images consist of thin line-drawings. For these cases, more sophisticated schemes such as those found in (Olshausen et al., 1993) are recommended. For our purposes however (refer to the accuracy tests in Chapter 5), the simplicity and flexibility of fixed numbers of inputs, outweigh any disadvantages.

**Model and Image.** The fundamental pose-estimation problem to which this thesis is dedicated, involves two patterns, and more fundamentally, the transformation that relates one pattern to the other. We will follow the common usage found in most of the point pattern matching literature (e.g. Gold et al. 1998), and denote these two patterns by *model* (or source) and *image* (or target). In the case where the pose estimation involves a pattern retrieved from memory via some process of recognition, the model is assumed to be the stored form of the pattern (stored-pattern), while the image is assumed to be the input-vision. In the case of pose estimation for motion analysis, the model is assumed to be the input-vision of the

previous time-step while the image is assumed to be the input-vision of the current time step.

**Cyclops and Attractors.** A model point/input will occasionally be referred to as a *cyclop*, while an image point/input will at times be referred to as an *attractor*.

### 4.1.2 Correspondences

**Correspondence.** A correspondence is essentially the vector formed between two matching (sufficiently similar) features, one in the model and the other in the image. In order for this vector to validly reflect the transformation that relates the model to the image (or vice-versa), both patterns must be superimposed on each other. A singlet refers to a single pair of matching points (one image point and one model point). A doublet refers to two pairs of matching points (two image points matching with two model points). Although doublets contain more information they are also more expensive<sup>1</sup>. The fact that doublets are expensive, which is specially evident when considering neural implementations, and the fact that singlets still provide sufficient information for our pose estimation purposes, both explain why we here focus primarily on singlets. Moreover, singlets are also general<sup>2</sup> and have clear and relatively cheap parallel implementations<sup>3</sup>. From this point onwards, the terms “correspondence” and “singlet” will be used interchangeably.

**Correspondence Distributions.** In most cases, two patterns will have more than one match, and thus more than one correspondence. The set of all correspondences between two patterns provide a rich source of visual information, be it about pose, shape, motion, or other aspects of the environment. Correspondence distributions are thus sets of correspondences, each with their individual properties (e.g. angle and length), which if analyzed adequately can give us important clues about the external environment. See Fig. 4.2 for several examples of scattergrams/distributions of correspondence properties resulting from an input-cloud

---

<sup>1</sup>If there are  $n$  model nodes (and  $n$  image nodes), then there are  $n^2$  singlets and  $\left[\frac{n(n-1)}{2}\right]^2$  doublets.

<sup>2</sup>Within vision, they are applicable to at least pose, motion, shape and stereopsis (refer to Chapter 10 for a more detailed exposition).

<sup>3</sup>Refer to Chapter 6 for an elucidation of parallel architectures representing singlets.

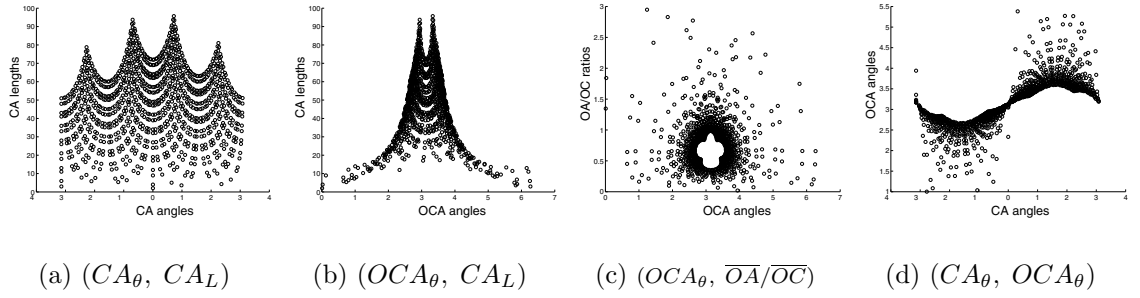


Figure 4.2: Various distributions of correspondence properties.

shifted to the left by 12 pixels, scaled by a factor of 1.5 and rotated by  $\pi$  radians, relative to a pattern replete with unique features. The following list summarizes the various properties used to produce the depicted distributions:

1.  $\vec{CA}$ . The vector obtained from subtracting cyclop coordinates from corresponding attractor coordinates, i.e: an actual correspondence vector.
2.  $CA_\theta$  and  $CA_L$ . The angle and length of a correspondence.
3.  $OCA_\theta$ . The angle between vectors  $\vec{OC}$  and  $\vec{OA}$  where  $\vec{OC}$  is the vector formed by subtracting the origin coordinates from a cyclop, and  $\vec{OA}$  is the vector formed by subtracting the origin from the corresponding attractor.
4.  $\overline{OA}$  and  $\overline{OC}$ . The lengths of  $\vec{OA}$  and  $\vec{OC}$  respectively.

Our approach can be generalized into what we term *Correspondence Distribution Analysis* (CDA) which consists of two main aspects: 1) representation and 2) computation. Representation encompasses aspects such as the cardinality of correspondences (e.g. singlets or doublets) and the properties that define them (e.g. angle and length). Computation on the other hand refers to what ones does to the distributions, e.g: filtering, sharpening, accumulation/counting of properties, and others. In terms of representation, we will be focusing on singlets and the positions of the features involved, while in terms of computation, we will be concentrating mainly on the accumulation/counting of properties.

**Probability of a True Correspondence (PTC).** The converse of *PTC* is what we term the *probability of a false correspondence (PFC)*, where  $PFC = 1 -$

*PTC*. Recall that the definition of a correspondence states that it involves two matching features. The fact that two features match, or are sufficiently similar, does not imply that they are caused by the same physical entity, e.g: if our feature is “green” then one must accept the possibility of a point belonging to a green leaf matching with a point belonging to a green door. If a correspondence matches the same physical entity it is denoted a “true correspondence”, otherwise it is referred to as a “false correspondence”. The less discriminating a feature-type is, the more likely it is to cause false correspondences, and thus we can say that for that particular feature type the *PTC* is smaller (or the *PFC* is larger). Image properties also affect the *PTC*, e.g.: a pattern with many rare and variable features tends to produce cases with larger *PTC<sub>s</sub>*.

### 4.1.3 Local Invariant Features

As was made clear in the previous subsection, in order to detect a correspondence one needs a match between two features. Furthermore, the probability of a correspondence being true depends heavily on the quality of the type of feature being used. Thus, some attention must be given to the issue of what feature-types to use.

**Locality.** A local feature is one that describes the visual content derived from a relatively small receptive-field centered at a particular image point. A global feature on the other hand describes the content of a receptive-field that encompasses the whole pattern. The fact that we are searching for multiple distinct correspondences amongst two patterns precludes the use of global features.

**Discriminatory Power.** The discriminatory power of a feature is related to how many distinctions it can make (see Swiniarski & Skowron 2003). A feature-type that consists of only two possibilities/values is likely to discriminate far fewer patterns than one with a range of 100 values.<sup>4</sup> The more discriminating a feature-type is the lower the probability of a false correspondence.

---

<sup>4</sup>Note however that two different feature-types with the same expressive range can have very different powers of discrimination.

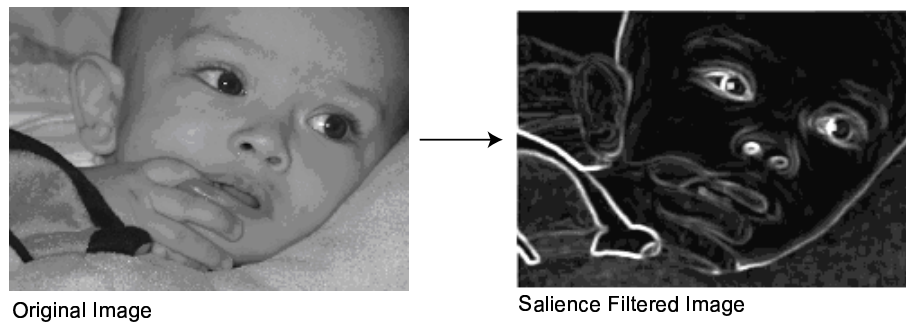


Figure 4.3: Example salience filtering.

**Invariance.** Invariant features are features that remain constant regardless of the transformations applied to the input, e.g: illumination, noise, translation, rotation and others (e.g. Tuytelaars 1999). It is fundamental that our local features be invariant to the transformations that we are trying to estimate, otherwise the same physical entity both in the model and in the image may not match after a transformation. Though it might seem at first that invariance and discriminatory power are opposing forces, in fact it is possible to have features that maximize both simultaneously. It is this type of feature (i.e. that maximizes discriminatory power and invariance) that we need, since it increases *PTC* whilst remaining robust to the transformations (i.e. correspondences can be found regardless of the transformation).

**Salience.** Salience-filtering pertains to the highlighting of salient/interesting features and the ignoring of common/uninteresting features (e.g. Jurie & Schmid 2004). This is very useful because it increases the *PTC*: common features produce many false correspondences, hence their removal increases the proportion of true correspondences. Another advantage is that it decreases the number of features that need to be considered (this is advantageous in the case of a sequential implementation). A simple example of a salience-filter might be a variance-filter, i.e: accept those features whose brightness variances supersede some threshold, and ignore the rest. Figure 4.3 provides an example of an image which has been salience-filtered. Appendix A briefly explains one possible algorithm for variance-based filtering.



#### 4.1.4 Problem Definition

The way we define a problem deeply affects the types of solutions we come up with, therefore some diligence is required in this regards. Three problem definitions are presented next, the last one of which underlies most of this thesis:

1. Find the optimal transformation that maps the largest number of unlabeled model points to unlabeled image points.
2. Find the optimal transformation that maps the largest number of labeled model points to labeled image points, whilst satisfying the label-matching constraint.
3. Find the actual transformation that maps the model to the image.

The first definition is a common one in the point pattern matching literature (e.g. Gold et al. 1998). One important aspect about it is that it mentions an “optimal” rather than the “actual” transformation between model and image points, where optimality can be defined by different criteria. This type of definition usually implies that pose-estimation is not the main concern, but both the means and a possibly useful side effect for some other function such as object recognition. Since in our thesis we are mainly concerned with features that have some discriminatory power, we are thus interested in point labeling, and thus in the second definition we provide for this extension. In some cases, this definition might be useful, however, pose-estimation is still a secondary/indirect function. Therefore, the third definition, greatly simplifies our problem, but carries the heavy assumption that an “actual transformation” exists against which solutions can be verified. Information regarding the “actual transformation” can come in several forms, e.g: 1) after transforming the model (or the image) by the estimate, the degree of match between the patterns serves as a measure of the accuracy of the estimate (or the degree to which the actual transformation was found), 2) in an ecological context, the quality of an estimator can be verified by its survival value (safely assuming that a faulty estimator can have disastrous consequences) and 3) in an artificial/experimental

context an experimenter can manipulate the transformations and thus information regarding the “actual transformation” is always available.

If not stated to the contrary, all patterns used in this chapter have a  $PTC = 1$ .

## 4.2 Pose Estimation Algorithms

### 4.2.1 Transformations

#### Transformation groups

Determining the pose of a target pattern is essentially equivalent to determining the set of geometric transformations that relates a source pattern in some canonical configuration to the target pattern.

Table 4.1 presents several groups of transformations in increasing order of generality and distinguishes them in terms of the properties they preserve (i.e. their invariants) between source and transformed patterns (see Hartley & Zisserman 2004). The “lines” (collinearity) property refers to the fact that lines in the source pattern remain as lines in the transformed pattern. The “parallelness” property means that parallel lines in the source pattern remain as parallel lines in the transformed pattern. The “angles” property refers to the relative angles between lines and finally, the “lengths” property refers to the lengths of lines. The group of rigid transformations, which is the least general of all, preserves all the geometric properties presented in the table. Transformed patterns are translated and/or rotated. The similarity group of transformations, preserves all properties except the length of lines: patterns can be translated and/or rotated and/or scaled (isotropically<sup>5</sup>). The affine group of transformations preserves collinearity and parallelness but does not necessarily preserve line lengths and relative angles: transformed patterns can exhibit translation, rotation, scaling and shearing. The projective group of transformations is the most general of all the groups presented in Table 4.1 and the only property it preserves is collinearity.

We have chosen to focus on similarity transformations because they encapsulate

---

<sup>5</sup>A single factor for all coordinates, e.g.: no horizontal vs. vertical scaling.

Table 4.1: Transformation groups and their invariants.

	Lines	Parallelness	Angles	Lengths	Example
Rigid	✓	✓	✓	✓	
Similarity	✓	✓	✓	⊗	
Affine	✓	✓	⊗	⊗	
Projective	✓	⊗	⊗	⊗	

the largest differences between source and target patterns. Affine, projective and other groups of transformations are fundamental, and need to be dealt with, but the differences (beyond the similarity transformations that they already embody) between source and target patterns that they represent are more subtle<sup>6</sup>, and therefore, one might argue that for a first rapid approximation, most organisms only need to estimate a similarity transformation. Subsequent stages or more general computations can then provide the organism with more refined estimates. Regardless of the validity of this argument, the approaches being discussed in this thesis can be naturally extended to other more general groups of transformations. By restricting ourselves to the similarity group we also simplify the analysis and presentation of the algorithms and architectures.

### Transformation ordering

Narrowing our scope to similarity transformations allows us to be very precise about our aim. Our hardest<sup>7</sup> aim is thus to estimate four unknowns: horizontal shift, vertical shift, rotation and scaling. At this point it is useful to recall that the order in which transformations are applied to patterns or points (to simplify our discussion) can lead to different results. Intuitively one can accept that translating a point first and then rotating it leads to a completely different position than if the point is rotated first and then translated. At this stage we will introduce some simple notation for expressing transformations and their relative orderings:  $T_x$  and  $T_y$  stand

<sup>6</sup>If  $D_\tau = |Img - Transform_\tau(Img)|$  then  $\frac{\partial D_{T_x}}{\partial T_x} = \frac{\partial D_{T_y}}{\partial T_y} \approx \frac{\partial D_s}{\partial s} \approx \frac{\partial D_\theta}{\partial \theta} \gg \frac{\partial D_{Project}}{\partial Project}$ .

<sup>7</sup>We will also be looking at easier component problems such as pure translation estimation.

for horizontal and vertical translations respectively,  $S$  refers to scaling,  $R$  refers to rotation,  $(X_1 \rightarrow X_2)$  means that transformation  $X_1$  precedes transformation  $X_2$ ,  $(X_1 \rightleftharpoons X_2)$  means that transformations  $X_1$  and  $X_2$  can be placed in any order,  $(X_1 \rightarrow X_2) \equiv (X_2 \rightarrow X_1)$  means that placing  $X_1$  and  $X_2$  in different orders leads to the same result (so  $(X_1 \rightleftharpoons X_2) \Leftrightarrow (X_1 \rightarrow X_2) \equiv (X_2 \rightarrow X_1)$ ),  $(X_1 \rightarrow X_2) \not\equiv (X_2 \rightarrow X_1)$  means that placing  $X_1$  and  $X_2$  in different orders leads to different results,  $B = R_\theta(A)$  represents the fact that  $B$  results from rotating  $A$  by  $\theta$  and  $s \cdot A$  represents the scaling of vector  $\vec{OA}$  by a factor  $s$ , where  $O$  represents the origin.

One result which will be useful in subsequent sections proves that scaling and rotation can be placed in any order (i.e.  $(S \rightleftharpoons R)$ ). Let  $A$  be a point defined by  $(x_1, y_1)$ ,  $A'$  be a point defined by  $(x'_1, y'_1)$  and  $B$  be a point defined by  $(x_2, y_2)$ .

Considering that  $(S \rightarrow R)$  implies that  $B = R_\theta(s \cdot A)$  and

if  $A' = s \cdot A$  then  $B = R_\theta(A')$  so

$x_2 = x'_1 \cos\theta - y'_1 \sin\theta$  and  $y_2 = x'_1 \sin\theta + y'_1 \cos\theta$  and because

$A' = (x'_1, y'_1) = (s \cdot x_1, s \cdot y_1)$  we finally get

$$\begin{cases} x_2 = (s \cdot x_1) \cos\theta - (s \cdot y_1) \sin\theta \\ y_2 = (s \cdot x_1) \sin\theta + (s \cdot y_1) \cos\theta \end{cases} \equiv \begin{cases} x_2 = s \cdot (x_1 \cos\theta - y_1 \sin\theta) \\ y_2 = s \cdot (x_1 \sin\theta + y_1 \cos\theta) \end{cases} \quad (4.1)$$

On the other hand,  $(R \rightarrow S)$  implies that  $B = s \cdot R_\theta(A)$  and

if  $A' = R_\theta(A)$  then  $B = s \cdot A'$  and

$x'_1 = x_1 \cos\theta - y_1 \sin\theta$  and  $y'_1 = x_1 \sin\theta + y_1 \cos\theta$  then

$B = s \cdot A'$  leads to

$$\begin{cases} x_2 = s \cdot (x_1 \cos\theta - y_1 \sin\theta) \\ y_2 = s \cdot (x_1 \sin\theta + y_1 \cos\theta) \end{cases} \quad (4.2)$$

which is equivalent to the results in Eq. 4.1 which proves that  $(S \rightleftharpoons R)$  is indeed correct.

Having shown how the relative ordering of scaling and rotation is inconsequential,

it should be useful to provide a simple demonstration of how different orderings can lead to different results. For example, in the case of horizontal translation and scaling, order is crucial, i.e:  $(S \rightarrow T_x) \neq (T_x \rightarrow S)$ .

If  $(S \rightarrow T_x)$  then  $B = s \cdot A + T_x$  or

$$x_2 = s \cdot x_1 + T_x \quad (4.3)$$

On the other hand, if  $(T_x \rightarrow S)$  then  $B = s \cdot (A + T_x)$  or

$$x_2 = s \cdot (x_1 + T_x) = s \cdot x_1 + s \cdot T_x \quad (4.4)$$

which is not equivalent to Eq. 4.3 thus proving that the order in this case is not inconsequential.

Having shown that the relative order in which transformations are applied cannot be ignored, it is now important to determine what type of ordering is exhibited by a cloud of inputs when it changes configuration according to the group of similarity transformations.

### Cloud vs. attractor transformations

When a cloud changes configuration (i.e. when it is translated and/or rotated and/or scaled) it does so relative to itself. Therefore the relative order in which transformations are applied is inconsequential: no matter how we apply the transformations the final cloud configuration will be the same.

The question we are asking however is not about the order of cloud transformations, but its consequence on attractors. This is so, because we want to use the displacements of attractors relative to cyclops to infer transformations. Since we know that, when talking about changing the configuration of a cloud, transformations can be applied in any order, it is sufficient to look at one particular order, say  $(S \rightarrow R \rightarrow T)$ . Brief introspection will show that this cloud sequence corresponds to the attractor sequence of transformations  $(S \rightarrow R \rightarrow T)$ , which can further be generalized to  $[(S \rightleftharpoons R) \rightarrow T]$  since the relative order of rotation and scaling have

already been shown to be inconsequential<sup>8</sup>.

### Constrained and unconstrained cases

A single correspondence (or a singlet), involves two points (a cyclop and an attractor), and thus provides us with two equations. Similarly, a pair of correspondences (or a doublet), involves four points and thus provides us with four equations. This is crucial because, since two equations can give exact solutions only when two unknowns are involved, singlets can provide constrained solutions only when the transformation space is defined by two dimensions, e.g:  $(T_x, T_y)$ ,  $(s, \theta)$ ,  $(T_x, s)$ , etc.

Similarity transformations, as already mentioned, consist of horizontal and vertical shifts, isotropic scaling and rotation, and therefore result in four unknowns. Hence, they are constrained only by doublets. This is where the challenge lies: how can one estimate the four unknowns using singlets, that is, in severely unconstrained conditions? There are various reasons for insisting on single correspondences. Firstly, they are simpler/cheaper to represent in neural architectures (doublets involve a more severe combinatorial explosion and it is probably impossible to pack sufficient wiring to represent them in a small enough amount of space). Secondly, the probability of a true singlet ( $PTS$ ) is larger than the probability of a true doublet ( $PTD$ ), assuming the same features are being used ( $P(TS) > P(TS)^2 = P(TD)$ ).

The following subsection derives all necessary transformation equations, starting from the simpler constrained cases and working its way up to the the unconstrained ones. Note that the constrained cases are still interesting to investigate mainly because of two facts: 1) in most cases the probability of a true correspondence is significantly smaller than one ( $PTC \ll 1$ ) and hence many if not most of the correspondences are misleading and 2) the similarity transformation problem can be broken into two stages, the first one, for example, performing a translation estimation with center of mass or other methods, the second one solving the rotation and scale estimation problem, which is a constrained case.

---

<sup>8</sup>Note that a cloud transformation sequence of  $(T \rightarrow S \rightarrow R)$  does not correspond to the same  $(T \rightarrow S \rightarrow R)$  sequence at the attractor level.

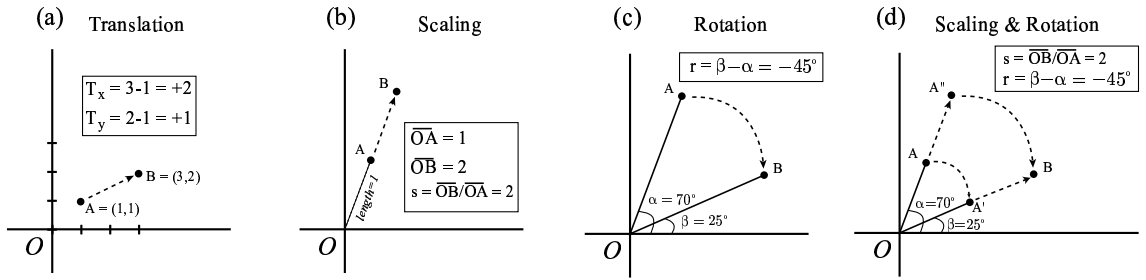


Figure 4.4: Constrained transformations applied to a single point.

The application of single correspondences to the problem of pose estimation will from this point onwards be abbreviated to SCAPE, i.e: single correspondence analysis for pose estimation. When SCAPE is applied to constrained problems it is abbreviated to C-SCAPE. Conversely, when it is applied to unconstrained problems it is abbreviated to U-SCAPE.

## Equations

The following cases will be derived in this section, the first four of which are constrained: 1) translation, 2) scaling, 3) rotation, 4) scaling and rotation, 5) translation and scaling, 6) translation and rotation and 6) translation, scaling and rotation.

The following conventions are used in this subsection.  $O$  corresponds to the origin of a cartesian coordinate system.  $A$  is a source point with coordinates  $(x_1, y_1)$  which is transformed into a point  $B$  with coordinates  $(x_2, y_2)$ . Sometimes an intermediary point  $A'$  with coordinates  $(x'_1, y'_1)$  will be used. The expression  $R_\theta(A)$  represents the rotation of point  $A$  by the angle  $\theta$ .

When point  $B$  results from transforming point  $A$  by only a translation (refer to Fig. 4.4(a)), estimation of this transformation is quite intuitive

$$\begin{cases} T_x = x_2 - x_1 \\ T_y = y_2 - y_1 \end{cases} \quad (4.5)$$

If point  $B$  results from scaling point  $A$  (refer to Fig. 4.4(b)) then the scaling can be estimated by

$$s = \overline{OB}/\overline{OA} \quad (4.6)$$

If a rotation is applied to  $A$  resulting in point  $B$  (refer to Fig. 4.4(c)), the transformation can be estimated by  $\theta = \tan^{-1}(y_2/x_2) - \tan^{-1}(y_1/x_1)$  or more simply

$$\theta = \beta - \alpha \quad (4.7)$$

where  $\beta$  stands for the angle between the  $\vec{OB}$  vector and the x-axis and  $\alpha$  represents the angle between the  $\vec{OA}$  vector and the x-axis.

Regarding the last constrained case involving both rotation and scaling (refer to Fig. 4.4(d)), as was previously shown, the order of the transformations is inconsequential, therefore it is sufficient to consider one particular ordering, say  $(R \rightarrow S)$ . It is not difficult to see that, rotation needs to transform point  $A$  to a point  $A'$  that lies on the line containing the vector  $\vec{OB}$ . From this, one can conclude that rotation estimation consists of  $\theta = \beta - \alpha$  where  $\beta$  refers to angle between  $\vec{OB}$  and the x-axis and  $\alpha$  corresponds to the angle between  $\vec{OA}$  and the x-axis. Then, considering the fact that the length of  $\vec{OA}'$  is the same as the length of  $\vec{OA}$ , the scaling factor can easily be computed by  $s = \overline{OB}/\overline{OA}$ . So, to summarise,  $(s, \theta)$  estimation can be performed by the following pair of equations

$$\begin{cases} \theta = \beta - \alpha \\ s = \overline{OB}/\overline{OA} \end{cases} \quad (4.8)$$

Now for the first unconstrained case: translation and scaling. As has already been shown, the relative ordering of attractor transformations resulting from cloud transformations is  $[(S \rightleftharpoons R) \rightarrow T]$ , and therefore in this particular case we need to consider  $(S \rightarrow T)$ . Since scaling precedes translation,  $(T_x, T_y) = B - s \cdot A$  and since we choose to leave  $s$  unconstrained, the transformation can be expressed as

$$\begin{cases} T_x = x_2 - s \cdot x_1 \\ T_y = y_2 - s \cdot y_1 \end{cases} \quad (4.9)$$



For the case of translation and rotation, and following the same argument used for the previous case, we need to consider  $(R \rightarrow T)$ . Since rotation precedes translation  $(T_x, T_y) = B - R_\theta(A)$ . If  $A' = R_\theta(A) = (x'_1, y'_1)$  and  $x'_1 = x_1 \cos\theta - y_1 \sin\theta$  and  $y'_1 = x_1 \sin\theta + y_1 \cos\theta$  then  $(T_x, T_y) = B - A'$  and thus

$$\begin{cases} T_x = x_2 - (x_1 \cos\theta - y_1 \sin\theta) \\ T_y = y_2 - (x_1 \sin\theta + y_1 \cos\theta) \end{cases} \quad (4.10)$$

Finally for the full similarity transformation case, we can consider  $(S \rightarrow R \rightarrow T)$  or  $(R \rightarrow S \rightarrow T)$  since they lead to the same result. Let us consider the first ordering  $(S \rightarrow R \rightarrow T)$  and leave both  $s$  and  $\theta$  unconstrained. If  $A' = (x'_1, y'_1) = s \cdot R_\theta(A)$  then  $(T_x, T_y) = B - A'$ . Since  $x'_1 = s(x_1 \cos\theta - y_1 \sin\theta)$  and  $y'_1 = s(x_1 \sin\theta + y_1 \cos\theta)$  the final estimation equations are

$$\begin{cases} T_x = x_2 - s(x_1 \cos\theta - y_1 \sin\theta) \\ T_y = y_2 - s(x_1 \sin\theta + y_1 \cos\theta) \end{cases} \quad (4.11)$$

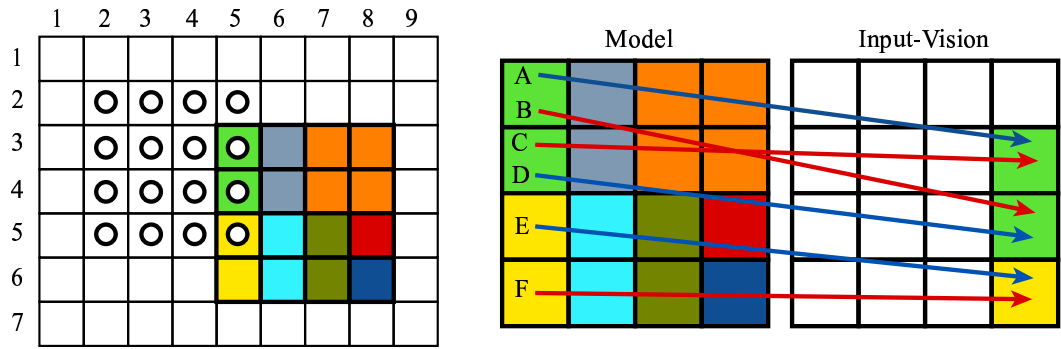
Having derived some fundamental equations for estimating poses involving translations and/or scalings and/or rotations we will now demonstrate some of the ways in which they can be used.

## 4.2.2 Constrained Estimation

As already discussed, when using single correspondences, it is possible to constrain at the most two unknowns. These two unknowns might be, for example,  $T_x$  and  $T_y$  in a translation-estimation problem or  $s$  and  $\theta$  in a scale and rotation estimation problem. By themselves these constrained cases are already useful, but one might also find them in conjunction/interaction with other modules that solve other transformations.

### Translation

Recall that our approach involves extracting information from correspondence distributions. In this particular example we are interested in the horizontal and



Correspondence	A	B	C	D	E	F
<b>Veracity</b>	<b>True</b>	<b>False</b>	<b>False</b>	<b>True</b>	<b>True</b>	<b>False</b>
<b>Cyclop (x, y)</b>	(2, 2)	(2, 2)	(2, 3)	(2, 3)	(2, 4)	(2, 5)
<b>Attractor (x, y)</b>	(5, 3)	(5,4)	(5, 3)	(5, 4)	(5, 5)	(5, 5)
<b>T<sub>x</sub> Estimate</b>	5-2 = +3	5-2 = +3	5-2 = +3	5-2 = +3	5-2 = +3	5-2 = +3
<b>T<sub>y</sub> Estimate</b>	3-2 = +1	4-2 = +2	3-3 = 0	4-3 = +1	5-4 = +1	5-5 = +0

T <sub>x</sub>		T <sub>x</sub>		T <sub>x</sub>		T <sub>x</sub>		T <sub>x</sub>		T <sub>x</sub>		T <sub>x</sub>															
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Voting Structure Winner: T<sub>x</sub> = 3 T<sub>y</sub> = 1

Figure 4.5: An input-cloud and a set of correspondences and votes.

vertical vector-components of each correspondence: see Equation 4.5. In the case of translation estimation, the horizontal component of a correspondence is equivalent to the  $T_x$  estimate, while the vertical component of a correspondence is equivalent to the  $T_y$  estimate.

A 2D voting structure is used where each cell represents a different  $(T_x, T_y)$  combination. Each correspondence votes for a particular cell (each vote increments a particular cell's counter by one). After all (or a sub-set of all) correspondences have voted, the translation estimate corresponds to the cell with the largest number of votes: refer to Fig. 4.5 for a simple example.

If  $PTC = 1$  then one correspondence is sufficient for the estimate. However, if  $PTC < 1$ , more correspondences are required to build a more robust estimate, due to the ambiguating effect of false-correspondences. Note that even if  $PTC \ll 0.5$ , estimates tend to be accurate because false-correspondences vote for many different

cells while true-correspondences always vote for the same cell.

Algorithm 1 summarizes the approach. Note that  $r(T_x)$  refers to the resolution of  $T_x$ . A simple approach for constructing a set of correspondences is presented in Algo. 2. This is a somewhat naive yet explanatory algorithm in the sense that it falls prey to a basic combinatorial explosion: all model points are compared to all input-vision points. One practical alternative entails constructing feature lists for models and input-visions, where each feature is referenced to a list of points (with coordinates) that instantiate it. This naturally permits correspondences to be found automatically, without the need to search through an extensive list of potential candidates (where the majority are non-matches).

---

**Algorithm 1**  $T_x T_y$  Estimation

---

```

procedure ESTIMTXTY( $C$ )                                ▷ Input a set of correspondences ( $C$ )
   $Votes \leftarrow 0$                                        ▷ Initialize vote structure:  $[r(T_x) \ r(T_y)]$ 
  for all  $c \in C$  do                                       ▷ Scan correspondences
     $(x_1, y_1) \leftarrow getCyclop(c)$                        ▷ Coordinates of cyclop
     $(x_2, y_2) \leftarrow getAttractor(c)$                    ▷ Coordinates of attractor
     $aT_x \leftarrow x_2 - x_1$                                ▷ Compute a horizontal estimate
     $aT_y \leftarrow y_2 - y_1$                                ▷ Compute a vertical estimate
     $Votes(aT_x, aT_y) \leftarrow Votes(aT_x, aT_y) + 1$      ▷ Increment vote
  end for
   $(T_x, T_y) \leftarrow maxCell(Votes)$                    ▷ Estimates from the strongest cell
end procedure

```

---



---

**Algorithm 2** Collecting Correspondences

---

```

1: procedure COLLECTCORRESP( $I, M$ )                        ▷ Input-cloud and model
2:    $C \leftarrow 0$                                        ▷ Initialize correspondence set
3:    $z \leftarrow 0$                                        ▷ Initialize correspondence counter
4:   for all  $m \in M$  do                                       ▷ Scan model points
5:      $(m_x, m_y) \leftarrow getPosition(m)$                  ▷ Canonical point position
6:      $(\tilde{m}_x, \tilde{m}_y) \leftarrow imgCoord(m_x, m_y, I)$      ▷ Convert to image frame
7:      $m_f \leftarrow getFeature(m)$                        ▷ Model point feature
8:     for all  $i \in I$  do                                       ▷ Scan cloud inputs
9:        $(i_x, i_y) \leftarrow getPosition(i)$                ▷ Input position
10:       $i_f \leftarrow getFeature(i)$                        ▷ Input feature
11:      if  $m_f \equiv i_f$  then                                       ▷ Match implies correspondence
12:         $C(z) \leftarrow [\tilde{m}_x \ \tilde{m}_y \ i_x \ i_y]$            ▷ Store correspondence
13:         $z \leftarrow z + 1$                                ▷ Increment correspondence counter
14:      end if
15:    end for
16:  end for
17: end procedure

```

---

## Scaling and Rotation

This constrained case works much in the same way as the previous case. The voting structure is still two-dimensional, but now represents scale ( $s$ ) along one dimension and rotation ( $\theta$ ) along the other. The information that is extracted from each correspondence is also different now. In this case, the model/image origin needs to be taken into consideration. More specifically, the vectors  $\vec{OA}$  and  $\vec{OC}$  provide the necessary information, where  $\vec{OA}$  represents the vector defined by subtracting the origin from a correspondence's attractor and  $\vec{OC}$  represents the vector defined by subtracting the origin from a correspondence's cyclop. Scaling information is obtained by dividing the lengths of the  $\vec{OA}$  and  $\vec{OC}$  vectors, while rotational information is obtained by subtracting their angles (see Equation 4.8). The approach taken here is summarized in Algo. 3.

---

### Algorithm 3 Scale and Rotation Estimation

---

<b>procedure</b> ESTIMSCALROT( $C$ )	▷ Input a set of correspondences
$Votes \leftarrow 0$	▷ Initialize vote structure: $[r(s) \ r(\theta)]$
<b>for all</b> $c \in C$ <b>do</b>	▷ Scan correspondences
$(\vec{OA}, \vec{OC}) \leftarrow getOrigVecs(c)$	▷ Origin vectors
$s \leftarrow \overline{OA/OC}$	▷ Scale estimate
$\theta \leftarrow \alpha(OA) - \beta(OC)$	▷ Rotation estimate
$Votes(s, \theta) \leftarrow Votes(s, \theta) + 1$	▷ Increment vote
<b>end for</b>	
$(Scale, Rotation) \leftarrow maxCell(Votes)$	▷ Final estimates
<b>end procedure</b>	

---

### 4.2.3 Unconstrained Estimation

As we have already mentioned, single correspondences provide information sufficient only for constraining two unknowns. The similarity transformation group, with its four unknowns  $(T_x, T_y, s, \theta)$  is thus severely underconstrained. So how can we estimate transformations under these conditions (i.e. using single correspondences)?

In Equations 4.11, we have left  $s$  and  $\theta$  undefined, and we have defined  $T_x$  and  $T_y$ . It is not difficult to see that each one of these equations defines a surface. Refer to Fig. 4.6 for an example pair of surfaces, resulting from a single correspondence comprised of a cyclop at coordinates  $(x=5, y=5)$  and an attractor at coordinates

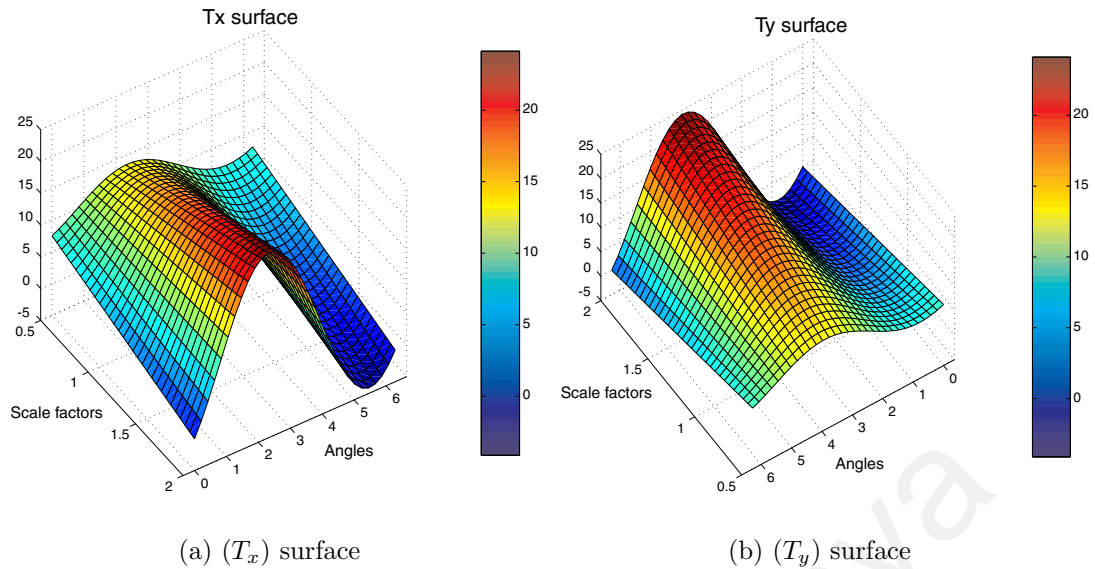


Figure 4.6: Two voting surfaces.

( $x=10, y=10$ ). The question is then, how might such surfaces be used to estimate the parameters behind a transformation?

One possibility is based on the notion that the surfaces resulting from a set of correspondences will tend to intersect more around the region corresponding to the actual transformation. Figure 4.7 illustrates how the approach works. For the preparation of this figure three distinct cyclops were used, each one of which was subjected to the same transformation (a horizontal shift of 8 units, a vertical shift of 5 units, a scaling factor of 1.5 and a rotation of  $\pi$  radians), thus leading to three distinct attractors. Sub-figure 4.7(a) on the left depicts the  $T_x$  estimation surface corresponding to one of the cyclop-attractor pairs (i.e. one of the correspondences). Sub-figure 4.7(b), in the middle, illustrates the intersection of the previous surface and another one resulting from a different correspondence. The sub-figure on the right, illustrates the intersection of all three surfaces. As one can see, the more surfaces that are added, the narrower the resulting intersection is. Furthermore, if one looks closely, one can see that the intersection corresponds approximately to the transformation parameters originally applied to the cyclops. One of the attractions of this approach, which is crucial to one of the fundamental motives underlying this work, is its simplicity, which in turn translates into a greater neural implementability and thus in greater biological plausibility, as will be soon demonstrated.

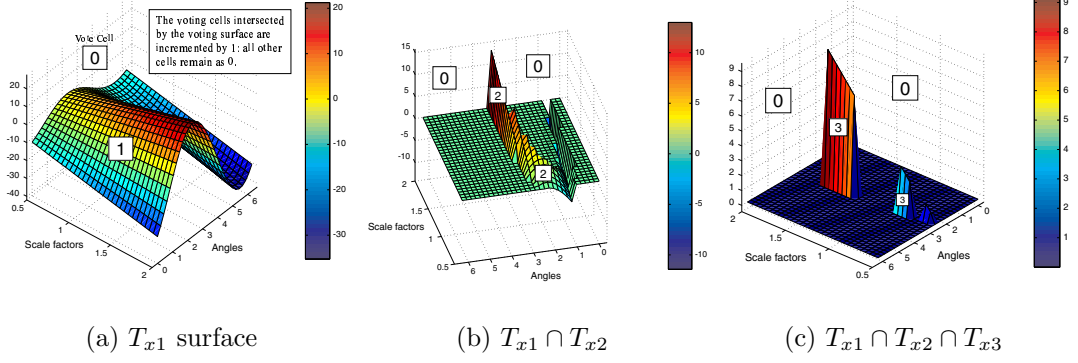


Figure 4.7: The intersection of  $T_x$  estimation surfaces.

The approach presented here, in its simplest form, essentially equates a correspondence to a “voting surface”. As new correspondences are considered, more “voting surfaces” pile up on each other, until eventually, a peak<sup>9</sup> should form, corresponding to the correct estimate. Since a single-correspondence and a particular  $(s, \theta)$  pair determine both a  $T_x$  and a  $T_y$  value, the voting structure is actually four-dimensional (i.e.  $[s \ \theta \ T_x \ T_y]$ ), rather than three-dimensional as in Fig. 4.7. Algorithm 4 condenses the main aspects of the approach.

---

**Algorithm 4** U-SCAPE algorithm

---

```

procedure U-SCAPE( $C$ )
   $Votes \leftarrow 0$  ▷ Vote structure:  $[r(s) \ r(\theta) \ r(T_x) \ r(T_y)]$ 
  for all  $c \in C$  do ▷ Correspondences
     $(x_1, y_1) \leftarrow getCyclop(c)$  ▷ Coordinates of cyclop
     $(x_2, y_2) \leftarrow getAttractor(c)$  ▷ Coordinates of attractor
    for all  $s$  do ▷ Scales
      for all  $r$  do ▷ Rotations
         $T_x \leftarrow x_2 + s(y_1 \sin\theta - x_1 \cos\theta)$ 
         $T_y \leftarrow y_2 - s(x_1 \sin\theta + y_1 \cos\theta)$ 
         $Votes(s, r, T_x, T_y) \leftarrow Votes(s, r, T_x, T_y) + 1$ 
      end for
    end for
  end for
   $(T_x, T_y, s, \theta) \leftarrow maxBin(Votes)$  ▷ Estimates from the strongest bin
end procedure

```

---

Essentially, in Fig. 4, a four-dimensional data structure is used for voting (or piling surfaces). The voting structure is initially filled with zeros. The notation  $r(x)$  denotes the resolution of the variable  $x$ . The algorithm loops through all of the

<sup>9</sup>The largest peak corresponds to the location with the most intersections and thus with the most votes.

correspondences, each one of which is used to generate  $(T_x, T_y)$  voting surfaces (using the equations in 4.11), which increment the cells/buckets that they intersect in the voting data structure. Once all correspondences have been considered, the largest (most voted for) cell is chosen as the estimate of the transformation parameters.

In this algorithmic chapter (see Fig. 2.1) we have defined core concepts such as correspondences and local invariant features, discussed several issues surrounding transformations in general and described how single correspondences can be used for estimating poses in constrained and unconstrained contexts. In the next chapter we will investigate how the approach performs in terms of estimation accuracy and robustness.

University of Malaysia

# Chapter 5

## Estimation Accuracy

The current chapter aims to demonstrate the accuracy and robustness of correspondence/vote based algorithms. In order to accomplish this, it is divided into two main sections, the first one dealing with synthetic patterns and the second one dealing with real patterns. By using synthetic patterns, it is possible to manipulate accuracy critical factors, and thus obtain a very concrete idea of the performance of the algorithms.

### 5.1 Synthetic Patterns

#### 5.1.1 Introduction

In its simplest form, the pattern-generator creates patterns consisting of a set of points randomly distributed in a 2D space. Apart from a pair of coordinates, each point also possesses a value representing its feature. The pattern-generator accepts three input parameters:  $NR$  (the number of non-repeating features),  $R$  (the number of repeating features) and  $maxPos$  (maximum coordinate modulus). Figure 5.1 depicts two examples of synthetic patterns. In both cases  $maxPos = 5$ . The left pattern consists of ten non-repeating features (circles with blue edges and faces with multiple colors) and three repeating features (squares with red edges and a single face color). The right pattern consists of ten non-repeating features and ten repeating features.



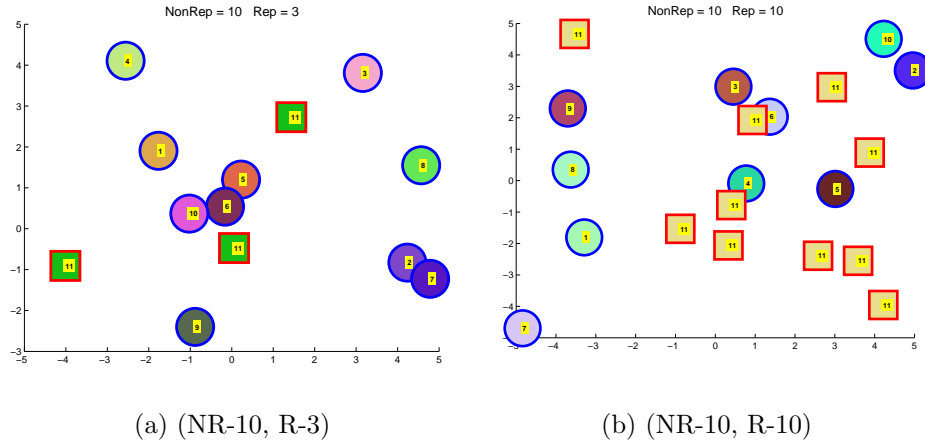


Figure 5.1: Two examples of synthetic patterns.

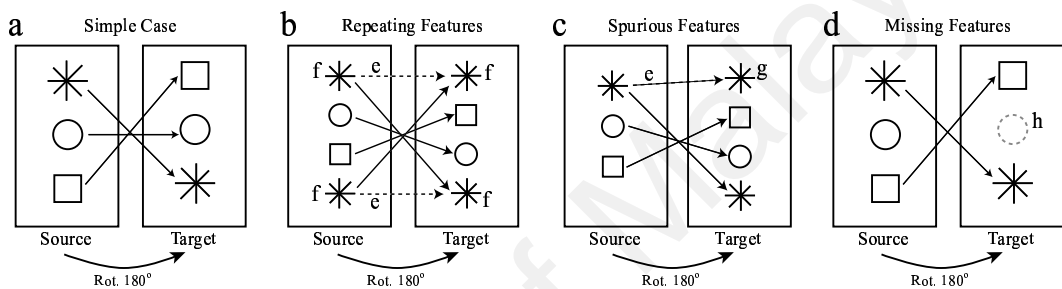


Figure 5.2: Three accuracy critical factors.

Probably the three most accuracy-critical factors for correspondence based approaches are: feature repeatability and the number of spurious and missing points in the target. Refer to Fig. 5.2 for an illustration of why these factors are critical to the accuracy of estimates. When patterns contain repeating features (note the star features (f) in case (b)) this leads to false correspondences (e) which in turn can create ambiguity. Observe also how spurious features (g) in the target pattern (case (c)) can also lead to false correspondences. Finally, missing features (h) in the target pattern (case (d)) are critical because they lead to a reduction in the number of true correspondences available.

### 5.1.2 Feature Repeatability

True correspondences are informative while false correspondences are misleading. The problem with repeating features lies exactly here: repeating features produce false correspondences and thus increase the probability of error. The question then is: how robust is a basic correspondence-based algorithm such as Algo. 4 (see

Chapter 4) to different proportions of non-repeating and repeating features? It turns out that it is significantly robust, as will be demonstrated below.

The general testing approach taken consisted of generating a pattern (source points), applying a similarity transformation to it (resulting in target points), and then using these two sets of points to estimate the relevant transformation. The estimation error was defined as the modulus of the difference between the actual transformation (known to the experimenter and/or test program) and the resulting estimate, e.g: if the actual  $T_x$  is 4 and the estimated  $T_x$  is 2 then the error is  $|2 - 4| = 2$ .

For each parameter setting (number of non-repeating features -  $NR$  - and number of repeating features -  $R$ ) five tests were run in order to reduce the possible effect of extraneous factors (e.g. shape). For each test, a new pattern was generated (where points acquired new random positions) and a new random transformation was applied. A  $maxPos = 5$  was chosen for all patterns. Transformations were randomly selected from the following sub-sets of transformations:

$$T_x \in [-5 \ -4 \ \dots \ +4 \ +5]$$

$$T_y \in [-5 \ -4 \ \dots \ +4 \ +5]$$

$$s \in [0.5 \ 0.6 \ \dots \ 2.0]$$

$$\theta \in [0 \ \pi/8 \ \dots \ (15\pi)/8]$$

The voting data-structure used, mirrored the above sets of transformations, except for the translation dimensions which were extended to  $\pm 8$ .

Table 5.1 provides a very concrete image of how feature repeatability affects the algorithm's accuracy while Table 5.2 summarizes the accuracy of a random guesser<sup>1</sup>, under the same circumstances. Table 5.1 depicts the mean errors for each transformation dimension (i.e.  $T_x$ ,  $T_y$ ,  $s$  and  $\theta$ ), for different parameter combinations.

---

<sup>1</sup>The accuracy of the random-guesser was determined by 100,000 Monte Carlo simulations and confirmed analytically via the expression  $Err = d \sum_{n=1}^r \frac{2n}{r^2}(r - n)$ , where  $d$  represents the transformation's resolution and  $r$  represents the number of positions available. See section B.2 in Appendix B for a proof/explanation.

Table 5.1: The effect of feature repeatability on error.

		NR=0	NR=5	NR=10	NR=15
R=5	$T_x$	0	0	0	0
	$T_y$	0	0	0	0
	$s$	0.04	0.06	0.06	0.02
	$\theta$	0	0	0	0
R=10	$T_x$	0	0	0	0
	$T_y$	0	0	0	0
	$s$	0	0	0.02	0.02
	$\theta$	0	0	0	0
R=15	$T_x$	0	0	0	0
	$T_y$	0	0	0	0
	$s$	0	0.04	0.04	0.02
	$\theta$	0	0	0	0

Table 5.2: The average error of a random guesser.

	$T_x$	$T_y$	$s$	$\theta$
Mean	3.6	3.6	0.5	2.1
Std. Dev.	2.6	2.6	0.4	1.5

As one can see, the algorithm is significantly robust to all parameter combinations. The scale dimension was the only one to manifest some inaccuracy, which was practically negligible (smaller than the scale resolution), and thus most probably caused by factors relating to the resolution of the voting structure, rather than the truth or falsehood of the correspondences.

As already mentioned, feature repeatability is critical because of how it affects the resulting numbers of true and false correspondences. Related to this is the probability of a true correspondence (i.e. PTC). Given the complete set of correspondences between a source and a target pattern, the PTC defines the probability of the first randomly selected correspondence being true. Figure 5.3 depicts how PTC varies as a function of the number of repeating and non-repeating features.<sup>2</sup> Comparing Fig. 5.3 with Table 5.1 one can see that even low PTCs lead to accurate results. The main cause for this lies in the fact that true correspondences are consistent and thus intersect their votes continuously on the same correct es-

<sup>2</sup>If  $nT = NR + R$  is the number of true correspondences and  $nC = NR + R^2$  is the total number of correspondences, then  $PTC = nT/nC$ .

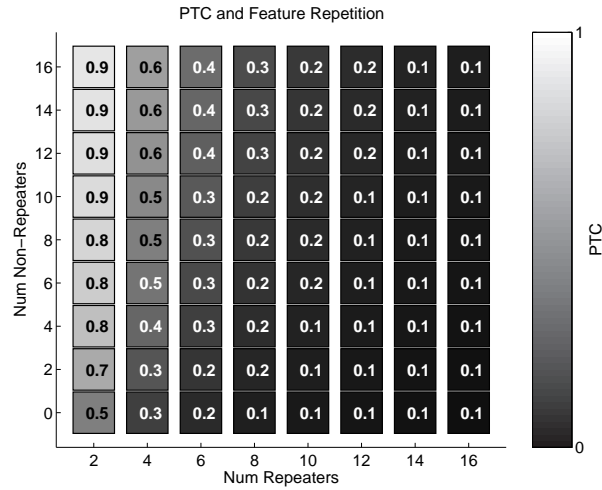


Figure 5.3: PTC as a function of feature repeatability.

estimate, while false correspondences dilute their votes across multiple contradictory estimates. Furthermore, we are using complete sets of correspondences, which is an acceptable assumption in the fixed/parallel context of neural implementations. When random (relatively small) sub-sets of correspondences are used, PTC becomes a critical factor to consider.

### 5.1.3 Spurious and Missing Features

Because correspondence-based approaches depend on the matching of features, they are likely to be affected by spurious and/or missing features, both of which can result from the ubiquitous condition of clutter, e.g: when a distracting object is partially placed in front of a target object, the occlusion leads to the loss of target features and the addition of distracting features.

Similarly to the feature repeatability experiments, multiple tests were run for different combinations of parameters, i.e: the number of spurious features and the number of missing features. For each test a new set of source points was generated. In order to generate the target points, the following operations were applied to the source points: points were randomly selected for removal (i.e. missing points), then several new and randomly positioned points were added and finally a random transformation was applied to all of the resulting points. In order to make the estimation problem more difficult the spurious points were given a feature label from one of the

source points, thus increasing the proportion of false correspondences.<sup>3</sup> To make the estimation problem even more difficult, and the algorithm’s performance boundaries more evident, the number of repeating features in the original pattern was made relatively high and the number of non-repeating features was made relatively low or zero.

Figure 5.4 summarizes the results. The figure consists of two columns of four graphs, each graph of which represents a different transformation dimension. The column on the left represents simulations run with  $R = 11$  and  $NR = 0$  while the column on the right represents simulations run with  $R = 11$  and  $NR = 5$ . Concerning the graphs, the y-axis represents the number of spurious features while the x-axis represents the number of missing features. Mean errors are represented by the brightness of circles (refer to the vertical “colorbar”) while the standard deviations of the errors are represented by the brightness of squares (refer to the horizontal “colorbar”).

One conclusion to be drawn from Fig. 5.4 is that the algorithms are significantly robust to many combinations of spurious and missing points. Predictably, estimation error is directly proportional to both the number of missing points and the number of spurious points. The right-hand column in Fig. 5.4 shows how the inclusion of even a small number of non-repeating features (i.e.  $NR = 5$ ) can have quite a strong effect on estimation accuracy. Notice how, contrariwise to the results in the left-hand column, the largest errors in the right-hand column are significantly smaller than those defined by chance-level (see Table 5.2), e.g: notice how the maximum  $T_x$  and  $T_y$  errors in the  $NR = 5$  case are both 0.6 while those defined by chance are both 3.6.

## 5.2 Real Patterns

Now that the robustness of the algorithms has been demonstrated in the context of synthetic patterns and several accuracy-critical factors, it is necessary to ask

---

<sup>3</sup>Note that the spurious features in real patterns are likely to have a certain proportion of non-matching unique features, thus creating less ambiguity.

whether this still holds true for real patterns?

### 5.2.1 Pre-processing

Before correspondences can be usefully extracted from real patterns some pre-processing is required. The first preprocessing step involves the detection of interesting/salient features. This eliminates a vast proportion of indistinct features and thus leads to a significant reduction in the resulting number of false correspondences. This also leads to an overall reduction in the number of correspondences in general, which is critical in the context of sequential computers and real-time processing. The second pre-processing step involves the characterization (or differentiation) of the salient features in order that discriminating matches, and thus correspondences, may be found between them.

#### Saliency

Many different types of saliency measures can be found in the literature: see for example (Mikolajczyk & Schmid, 2004) and (Sebe et al., 2003). We found that a simple brightness-variance measure was sufficient for our purposes. Briefly, our filter scans an input image with a small window, and defines the saliency of the central point as the mean deviation of the brightness values in the window. A threshold is subsequently applied, whereby only features whose mean deviations are large enough are considered salient. To reduce the resulting number of features even further, the saliency-filtered images are then processed with a “naive thinner”, which eliminates salient points with an excessive number of salient neighbors. See Fig. 5.5 for an example of saliency filtering and thinning. For the results presented below a window radius of 1 (i.e. dimensions  $3 \times 3$ ) and a threshold of 10 were used.

#### Features

As with saliency measures, the literature abounds in local invariant features: see for example (Jurie & Schmid, 2004) and (Schmid et al., 2005). Again, for the sake of simplicity and sufficiency we chose a simple feature with considerable robustness to

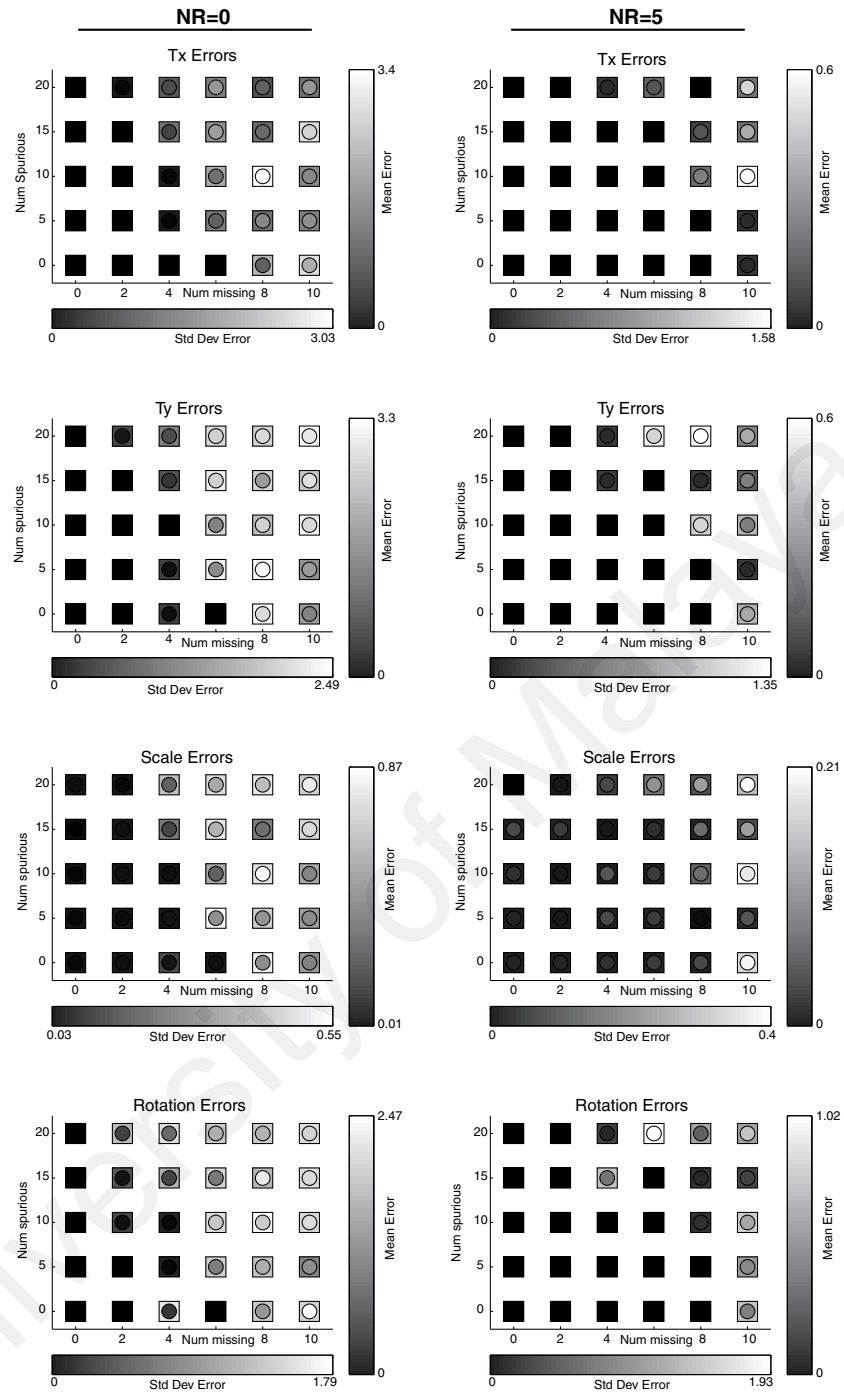


Figure 5.4: The effect of spurious and missing features on accuracy.

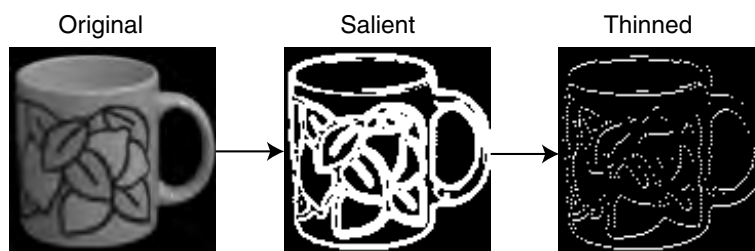


Figure 5.5: Example salience-filtering and thinning.

affine transformations, i.e: brightness histograms (see for example Brunelli & Mich 1999). The feature of a salient point was defined to be the brightness histogram of the data captured by a window centered on that point. All our tests involved a window of radius 3 (i.e. dimensions  $7 \times 7$ ). Note that an assumption of illumination constancy is required in the context of brightness-histograms, which can obviously be relaxed for features that incorporate illumination invariance.

The distance measure between two brightness histograms was defined to be the minimum overlap between them, i.e: if the minimum overlap of brightness values between two salient points was above a certain threshold (here defined as  $0.4 \cdot (2r+1)^2$  where  $r$  is the window-radius<sup>4</sup>), the points were deemed to have matching features.

## 5.2.2 The PTC of Real Patterns

An exercise which should allow us to predict and understand the performance of SCAPE in the context of real patterns, involves processing the latter with the above mentioned saliency measure and local descriptor, and observing the numbers of true and false correspondences that the patterns exhibit relative to themselves.

Figure 5.6 illustrates ten objects used in our experiment, the numbers of true and false correspondences that they exhibit (abbreviated to T and F respectively), and their resulting PTCs (abbreviated to P).<sup>5</sup> Note that the average is approximately 0.08, which although low, is not an impediment to accurate estimates, as was shown in the section on synthetic patterns (see Fig. 5.3 and Table 5.1). In that section it was also shown how even a very small number of true correspondences was enough to counteract the deleterious effects of spurious and missing points: note the large numbers of true correspondences in Fig. 5.6. In summary, a preliminary analysis of real patterns in regards to correspondence veracity suggests that accurate estimates are very likely.

---

<sup>4</sup>An overlap level of 40% allows for a degree of discriminability whilst retaining a sufficient amount of robustness regarding illumination and geometric variations.

<sup>5</sup>The objects represent a subset from the Columbia Object Image Library (COIL-100): see (Nene et al., 1996).



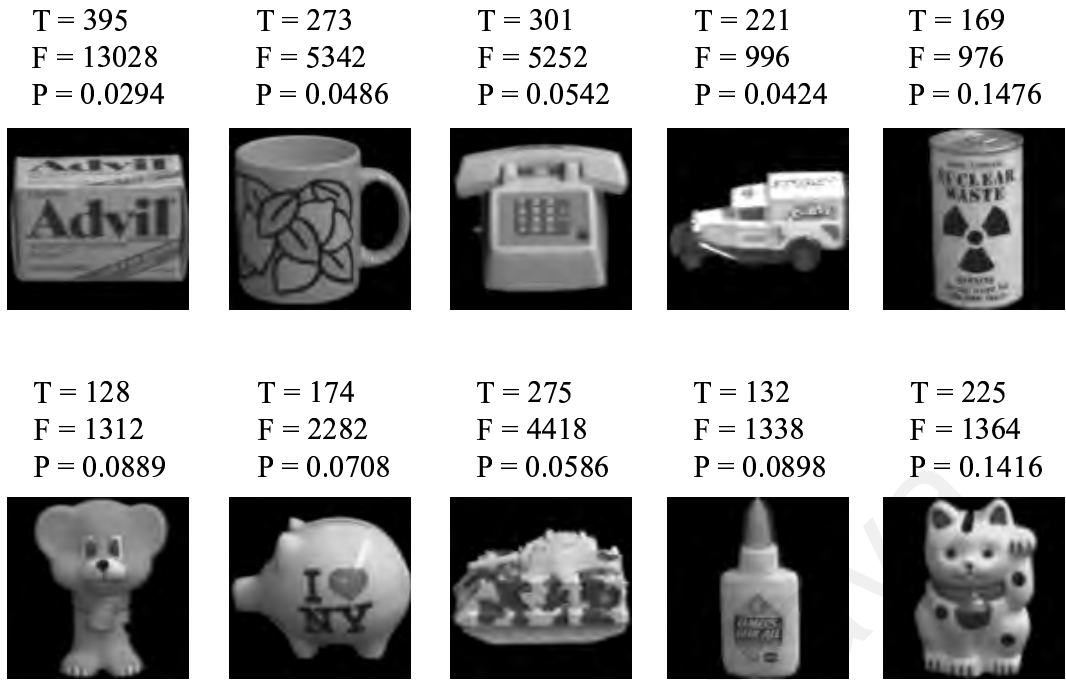


Figure 5.6: COIL objects and their PTCs.

### 5.2.3 Real Pattern Accuracy

In order to increase the realism of our tests, target objects were placed over heavy clutter consisting of multiple instances of non-target objects. The clutter provided a significant amount of spurious points and the limited-view of input-clouds often led to a significant number of missing points. Up to 960 separate tests were conducted, where for each test, a new random target object was selected, a new cluttered test image was generated and a new random transformation was applied to the input-cloud.<sup>6</sup> Figure 5.7 provides an intuitive feel for the clutter, the transformations applied and the resulting estimation accuracy.

Table 5.3 provides a global view of estimation accuracy: both the means and the standard deviations of the errors for a random guesser and SCAPE are presented. As can be concluded from the table, the algorithm is significantly accurate for real patterns. Note that many of the errors produced were artifacts of the data structure used for voting, i.e: a data structure with a higher resolution should reduce the number and size of errors further. It was also observed that the largest errors were

<sup>6</sup>The transformations applied exhibited the following ranges:  $T_x = -50|5|50$ ,  $T_y = -50|5|50$ ,  $s = 0.5|0.1|2$  and  $\theta = 0|\pi/4|7\pi/4$ , where  $x|i|y$  denotes a set of values starting at  $x$ , and increasing in increments of  $i$  up to  $y$ . The voting data structure exhibited the same ranges as the transformations with the exception of the rotational dimension which used  $\theta = 0|\pi/8|15\pi/8$ . The model patterns exhibited dimensions  $101 \times 101$ .

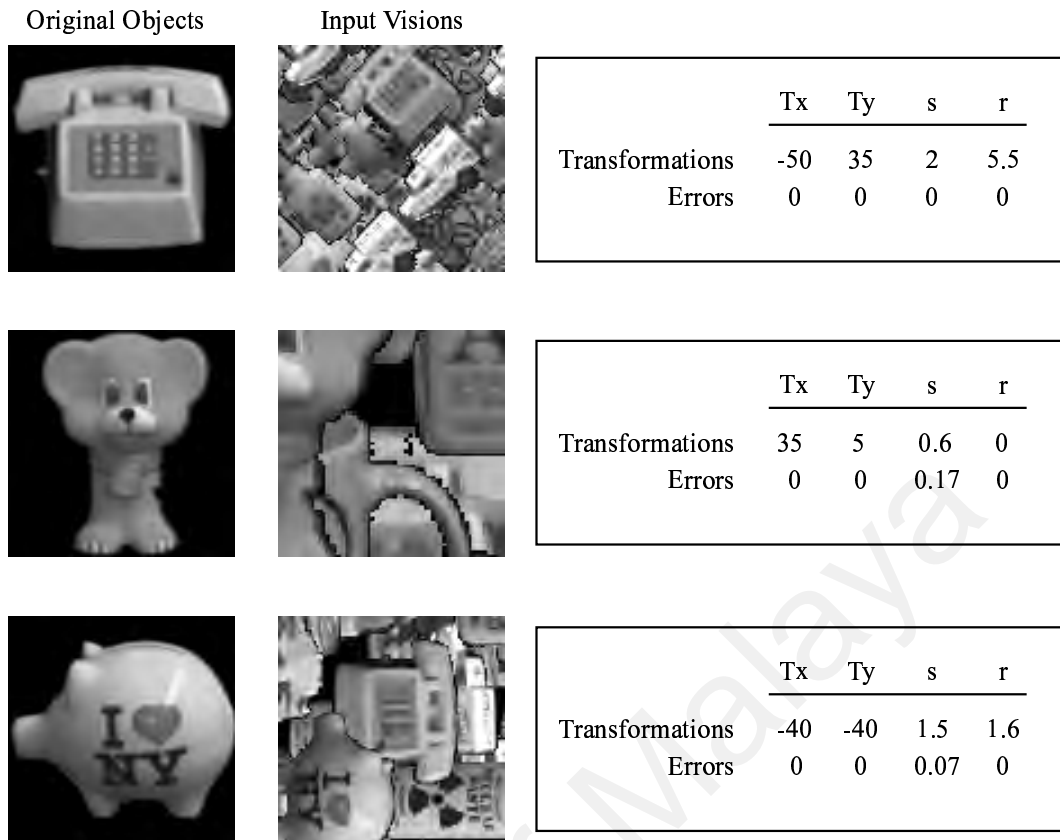


Figure 5.7: Examples of transformations, resulting input-visions and estimations.

Table 5.3: Mean chance and algorithmic estimation errors.

		$T_x$	$T_y$	$s$	$\theta$
Mean	Chance	34.9	34.9	0.5	2.06
	SCAPE	2.1	2.1	0.1	0.03
StdDev	Chance	24.8	24.8	0.38	1.49
	SCAPE	6.26	6.47	0.17	0.3

produced when all the dimensions of the cloud transformation were simultaneously at their largest magnitude (e.g.  $T_x = -50$ ,  $T_y = +50$ ,  $s = 0.5$  and  $\theta = \pi$ ). This situation is less common, and leads to a significant reduction of useful information, which understandably reduces accuracy, even likely so for human estimators.

In this algorithmic chapter (see Fig. 2.1), SCAPE's accuracy was demonstrated in the context of synthetic and real patterns. Several accuracy critical factors were isolated and studied, e.g: feature repeatability, spurious and missing points and PTC. The importance of preprocessing was also demonstrated. The demonstration of the robustness of correspondence distributions constitutes a first step in the

argument regarding their plausible existence in biological neural systems. In the following implementational chapter the algorithmic solution will be translated into an artificial neural architecture.

University of Malaya

# Chapter 6

## Artificial Neural Architectures

This chapter is concerned with embodying the algorithms of the previous chapter in artificial neural architectures (ANA). Aside from strengthening our general hypothesis, there are several other reasons why this might be interesting, e.g: 1) ANAs are parallel computing devices and thus can converge onto solutions much faster than their sequential counterparts, and 2) ANA implementations of general algorithms are often not exactly equivalent to the latter and can thus bring forth some extra advantages such as graceful degradation.

### 6.1 Artificial Components

Before moving on, and for the benefit of clarity, all of the artificial neural components required by some or all of the artificial neural architectures in this chapter, are here depicted and explained in Fig. 6.1. The selection of artificial neural components was based on the rough maximization of the following factors: computational flexibility, structural organization and biological plausibility.

Most of the components depicted are quite common to the artificial neural network literature (refer to Haykin 1999 for a good introduction). The only ones which probably appear less frequently and require some clarification are the “Rhythmic” and “Dyn-Connect” components. The first one of these, is a component that fires at regular intervals, which is controlled by a frequency component, and whose main purpose is to synchronize computations (e.g. some computations must “wait” for the













	Name	Parameter(s)	Function
	Input	-	Represents data from the environment
	Reset Sum	Bias	Resets to zero and then sums inputs and bias
	Temporal Sum	Bias	Sums inputs, bias and previous state
	Linear	Bias	Copies the input
	Semi-Linear	Bias	Outputs the input if the latter is greater than 0, otherwise outputs 0
	Threshold	Bias	Outputs 1 if input is greater or equal to 0, otherwise outputs 0
	Sigmoid	Steepness, Bias	Squashing function
	Rythmic	Frequency	Fires at a particular frequency
	Pair-Comp	-	Outputs 1 if the left input is greater than or equal to the right input (otherwise 0)
	Pos-Connect	Weight	Input multiplied by a positive weight and passed on
	Neg-Connect	Weight	Input multiplied by a negative weight and passed on
	Dyn-Connect	-	The connection weight is set by another node

Figure 6.1: Artificial Neural Components.

results of other stages/areas if incongruent outputs are to be avoided). The second component, short for “dynamic connection”, is defined by a weight which is dynamically determined by the output of some other component (e.g. a sigmoid node). This type of connection, as will be seen, is useful for certain types of maximum-value networks, among other computations (e.g. neuronal models of attention). It might also be useful to clarify at this point the meaning of the steepness parameter of the sigmoid unit. If the steepness parameter is  $s$  then the sigmoid is defined as:

$$f(x) = \frac{1}{1 + e^{-sx}} \quad (6.1)$$

As can be seen in Equation 6.1, as  $s$  increases, so does the steepness of the sigmoid function. When the actual value of the sigmoid steepness parameter is not specified in the descriptions below, it is assumed to be 3.

## 6.2 Feature Maps and Columns

The first question to ask when considering ANAs of pose estimation algorithms based on correspondence distributions is: how should visual patterns be represented? It is a well known fact that the primate visual system exhibits an abundance of topographic maps (e.g: Allman 1999). Topographic maps refer to neural representations that preserve the relationships (spatial or other) of features found in earlier, for example sensory, areas (see for example Goodhill et al. 1995). Although topographic relationships can be quite abstract (e.g. edge orientation) we are here concerned with spatial/neighborhood relationships. If these neighborhood preservation maps are relative to the retina then the maps are denoted as “retinotopic maps”: neighboring features in the retina, are represented by neighboring nodes at higher-level representations.

This abundance of maps in biological neural systems must not be without reason. It is likely that the usage of maps that preserve topographical (in this case, positional) relationships amongst features, confers certain structural and functional advantages, which would be unwise to forsake in the artificial domain. Therefore, objects are here represented as 2D topographical maps of features.

Most objects manifest a large variety of visual features. These features can be classified along different dimensions (e.g. color, shape, texture, among others), and each dimension can exhibit a vast range of sub-dimensions and gradations (e.g. different shades and combinations of red, green and blue wavelengths). Refer to Fig. 6.2(a) for an example of an object manifesting a multitude of features. How can this feature richness be incorporated into 2D map representations? One possibility is to have feature variations represented within feature columns. This being so, patterns can be represented by columnar maps, where feature position varies along the  $x$  and  $y$  axes, while other feature characteristics (e.g. colour) vary along the  $z$  axis. Figure 6.2(b) summarizes the representation just described. Note that this solution deviates from what current evidence suggests is happening in most biological systems: the response properties of cells vary parallel to the cortex whilst remaining mostly invariable in the perpendicular direction (e.g: Goodhill & Carreira-Perpinan

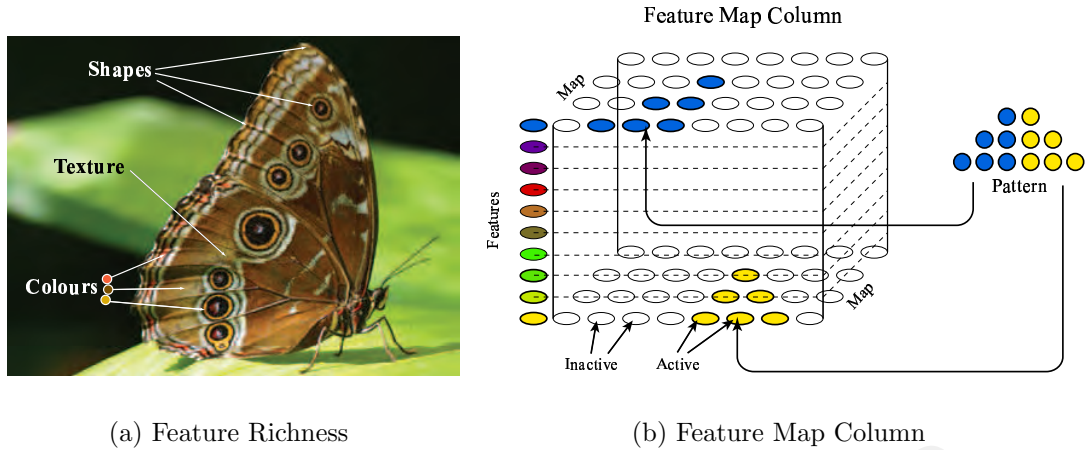


Figure 6.2: Feature richness and feature map columns.

2002).

### 6.3 Correspondence Detectors

Having described a simple representation that encompasses both feature type and feature position, the basis for correspondence detection has been set. Recapitulating, a correspondence refers to a match between a feature at the model (memory) side and a feature at the image (environment) side, and the pose (or other) information that the features imply from their relative positions.

So, using the feature representation depicted in Fig. 6.2(b), how can correspondences be represented? The simplest way is through conjunctions (logical-AND). In this scheme, each possible correspondence has a conjunction-unit dedicated to it. Each correspondence detector (conjunction node) has two inputs: one from a feature node at the model side (cyclop) and one from a feature node at the image side (attractor). If both feature-nodes of that correspondence detector are firing then so will the correspondence detector, otherwise it will remain silent. Figure 6.3 summarises this representation for the case of a single feature-type. Note that, due to space limitations, only the most relevant correspondence detectors have been depicted.

It is evident from the above scheme, that we are dealing with a representation that manifests a moderate combinatorial-explosion, e.g.: if both image and model

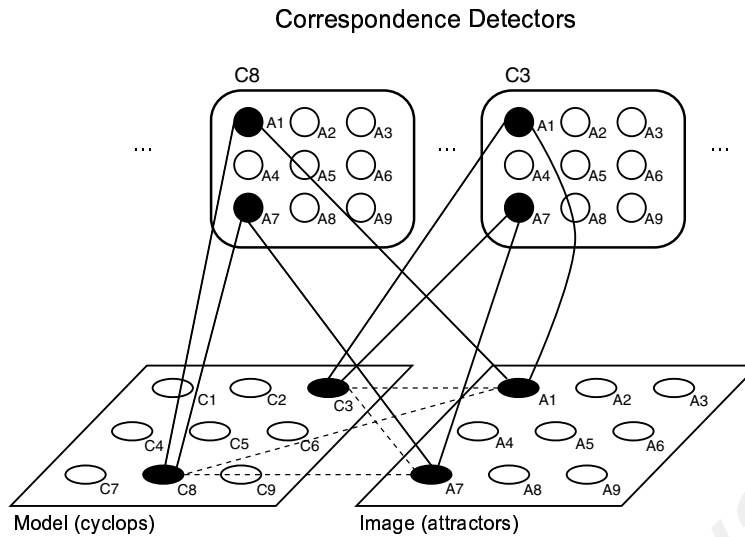


Figure 6.3: Correspondence detection.

patterns consist of  $t$  feature-types and each feature can be found in  $n$  different positions, then  $tn^2$  correspondence detectors are required. For now, however, we will just contain the explosion, by somewhat restricting the resolution of patterns. Later, in Chapter 7 we will pursue this issue further.<sup>1</sup>

## 6.4 Transformation Voting Structures

Having described a representation for feature types, feature positions and correspondence detectors, the final step in the sequence refers to the question on how to compute and represent pose estimates. As has been demonstrated in Chapter 4, the core of our approach consists in each correspondence voting for poses/transformations that are consistent with it. Each correspondence has one (constrained case) or more (unconstrained case) poses with which it is consistent. These poses are determined by the various equations discussed in Section 4.2.1 and therefore are known a-priori. The fact that they are known a-priori means that we can embody them in the pattern of connections between correspondence detectors and some voting structure where each node represents a different transformation. In short, each node in the

<sup>1</sup>Note that this combinatorial explosion, is one of the reasons why singlets were chosen over doublets for pose-estimation within the similarity transformation group. If models/images consist of  $t$  feature-types and  $n$  different positions,  $t \left[ \frac{n(n-1)}{2} \right]^2$  doublet-detectors are required. The structural demands (number of nodes and connections) that this combinatorial explosion entails are most probably too prohibitive for any practical applications or biological systems.



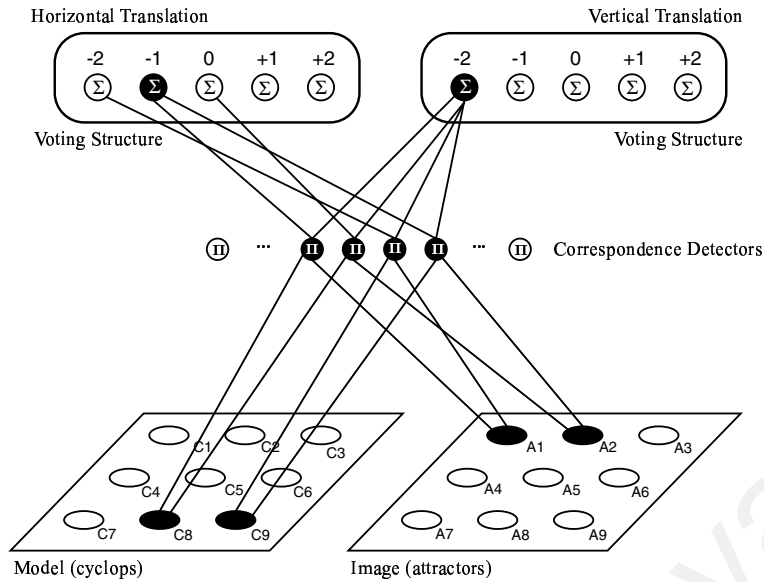


Figure 6.4: Correspondences voting for translations.

voting structure sums up all its incoming activity, and the node which accumulates the most input represents the winning pose.

Figure 6.4 summarizes this representation. The diagram illustrates the simple case where we have one feature type and 9 possible feature positions. The constrained translation case was chosen for sake of simplicity. For further sake of clarity, we divided the  $2D$  voting structure ( $T_x T_y$ ) into two separate  $1D$  structures ( $T_x$  and  $T_y$ ), one for horizontal translations and the other for vertical translations (in spite of the resulting accuracy reduction). Since there are two active features on the model side, and two active features on the image side, this means that four correspondences can be detected. Each one of these correspondences is connected to a specific node in both voting structures. As can be seen, there is a single node both in the horizontal translation and the vertical translation voting structures, which receives a majority of input, and thus represents the transformation between the model and image patterns (in this case:  $T_x = -1$  and  $T_y = -2$ ).

Recall that the architecture described so far, which includes input maps, correspondence detectors and summation voting nodes, can be seen as a special case of Higher Order Neural Networks. In Chapter 2 we outlined the main specializations, which we repeat here for convenience: 1) the higher order information represented at the first layer (after the input maps) is limited to second order structure, 2) two in-

put maps rather than one are used, 3) inter-map (between the two maps) rather than intra-map second order structure is represented and 4) the connections between the second order nodes and the summation or voting nodes are mathematically defined rather than learnt.

## 6.5 Maximum Value Networks

Even the simple example in Fig. 6.4 demonstrates the potential problem of nodes which represent incorrect transformations exhibiting some activation: the nodes representing horizontal translations of  $-2$  and  $0$  both exhibit non-zero activity. In general, transformation-nodes can be “falsely activated” by false correspondences and/or by voting-manifolds (in the unconstrained case). Because of this phenomenon, at least in the neuronal case, it is advantageous to “sharpen” the voting results. The ideal “sharpening” is one that maximizes the activity of the node with the highest activity, and completely suppresses the activities of all other nodes (e.g.  $[0.9 \ 0.4 \ 0.8] \rightarrow [1 \ 0 \ 0]$ ). The literature on neural architectures for such sharpening is quite vast, and various approaches are possible (e.g. Koutroumbas 2004). We have investigated several of these approaches which we have categorized into the following three groups: 1) lateral inhibition networks, 2) mean based networks and 3) paired-comparison networks. We have chosen to consider all of these networks, because each one offers certain unique advantages in particular situations.<sup>2</sup>

Before continuing, it is important to mention an important assumption. All input data to the maximum-value networks is assumed to contain one only maximum-value (e.g.  $[0.9 \ 0.5 \ 0.6]$  is a valid input-vector while  $[0.9 \ 0.6 \ 0.9]$  is an invalid input-vector). Most of the networks discussed below can be easily adapted to deal with this situation (e.g. a random fluctuation can be injected at a particular stage of processing so that a single random decision is made among the several maximum-values), however, for the sake of simplicity, and assuming that voting structures will

---

<sup>2</sup>The computation of a maximum-value is a common necessity in most problem domains, and therefore it is reasonable to assume that biological neural systems employ it in several processing areas/stages, and so it is an interesting hypothesis (possibly daring) to suggest that biological systems (even within the same organism) do not always compute it in the same way.

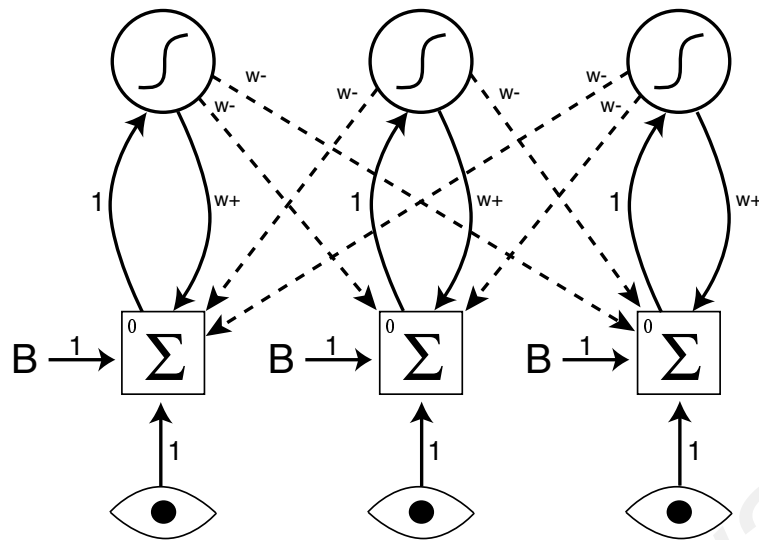


Figure 6.5: A lateral inhibition network.

in the great majority of cases produce data with a single maximum-value, we will keep to the above mentioned assumption.

### 6.5.1 Lateral Inhibition

Lateral inhibition networks, usually referred to as competitive networks, are probably the most common approach to neural maximum-value networks (see Majani et al. 1989 for an early example). The general idea is that each node inhibits its neighbors whilst auto-exciting itself. After several iterations, it is expected that the node with the maximum-value has completely suppressed all other nodes, whilst attaining a maximum activation level. Figure 6.5 summarizes the general architecture.

Within this scheme, we have investigated three possibilities. The first one uses fixed values for the excitatory and inhibitory connection weights. The second one allows the inhibitory and excitatory weights to vary, but the way they vary is predetermined by several rate parameters. The third one again allows the excitatory and inhibitory weights to vary, but this variation is primarily conditioned by the nodes' activation levels.

## Fixed Weights

This solution is completely described by the architecture in Fig. 6.5. Both the auto-excitatory weight  $w^+$  and the inhibitory weight  $w^-$  are shared by all competing nodes. So the main question follows: what values should be given to these weights? As should be expected from such a simple architecture, there is no single set of values which can satisfy all situations. The choice of weights depends primarily on the following four factors: 1) the number of nodes, 2) the statistics of the input data, 3) how fast one wants the network to converge and 4) how precisely one wants the network to distinguish the largest values.

Unfortunately, the simplicity of the network comes at a cost. The network exhibits a pronounced inability to converge onto a single maximum-value, i.e: the network tends to not distinguish between several large values, exciting them equally, and thus converging onto a solution consisting of several maximum-values (e.g.  $[0.9 \ 0.8 \ 0.5 \ 0.2] \rightarrow [1 \ 1 \ 0 \ 0]$ ). For situations when this behavior is not critical, or when it might even be desirable, this simple network is beneficial.

## Rate Conditioned Weights

The main difference between this solution and the previous one, is the addition of two new parameters which control the way the excitatory and inhibitory weights change at each new iteration. Since the bias that feeds into the summation node is assumed to be zero, the set of parameters that characterizes this solution is:  $w^+$ ,  $\delta^+$ ,  $w^-$  and  $\delta^-$ . At each new iteration, the weights are updated in the following manner:

$$\begin{cases} w_{new}^+ = \delta^+ \cdot w_{old}^+ \\ w_{new}^- = \delta^- \cdot w_{old}^- \end{cases} \quad (6.2)$$

The main advantage of this solution is that it is much more successful in converging onto a single maximum-value than the previous solution. Figure 6.6 depicts an example simulation, using 100 nodes, with random inputs lying within the range  $[+2.4, +2.8]$ , with a sigmoid steepness of 1, where  $w^+$  and  $w^-$  were initially set to 1

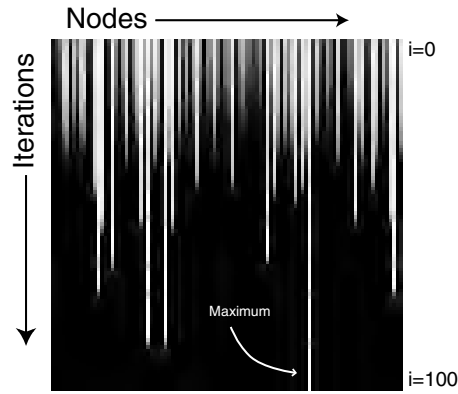


Figure 6.6: Node competition with rate conditioned weights.

and  $-1/n$  (where  $n$  refers to the number of nodes), and the  $\delta^+$  and  $\delta^-$  parameters were set to 1.046 and 1.096 respectively.

It is interesting to note that, in some cases, all node activities eventually become extinguished (i.e. become zero), but in spite of this, it is still possible to know the maximum-value by simply observing which was the last node to become extinguished.

In all these solutions that depend on one or more parameters (e.g.  $\delta^+$  and  $\delta^-$ ) there is the fundamental problem of how to find adequate values for them. As mentioned before these values depend on various factors such as input-statistics (e.g. mean and variance), so when these factors change, this forces us to search for new parameter settings. As always there are many ways to do this, ranging from more sophisticated optimization techniques all the way down to trial-and-error. Some cases might even lend themselves to a rigorous analysis that can lead to well defined expressions for generating the adequate parameter settings. The approach taken here, resulted from observing the behavior of the network in different situations and with different parameter settings. It was observed that setting  $w^+$  and  $w^-$  to 1 and  $1/n$  respectively, and then searching for adequate  $\delta^+$  and  $\delta^-$  values yielded effective solutions. Experimentation revealed the following two crucial observations: 1) increasing  $\delta^-$  tends to increase the resolution with which the network is capable of separating large values but with the added risk of instability, 2) increasing  $\delta^+$  tends to stabilize the network. From these two observations, a simple search algorithm was devised.

---

**Algorithm 5** Search-algorithm for  $\delta^+$  and  $\delta^-$ 

---

```
procedure SEARCH(searchIncr, maxNumMax, maxRecover)
   $\delta^+ \leftarrow 1$  ▷ Initialize  $\delta^+$ 
   $\delta^- \leftarrow 1 - searchIncr$  ▷ Initialize  $\delta^-$ 
   $recoveries \leftarrow 0$  ▷ Recoveries from instability
   $stable \leftarrow true$  ▷ Initially, stability is assumed
  while  $recoveries \leq maxRecover$  do
    while  $stable$  do
       $\delta^- \leftarrow \delta^- + searchIncr$  ▷ Increment  $\delta^-$ 
       $[stable, numMax] \leftarrow simulNet(\delta^+, \delta^-)$  ▷ Simulate
      if  $(stable) \& (numMax \leq maxNumMax)$  then
        store( $\delta^+, \delta^-$ )
      end if
    end while
    while  $\neg stable$  do
       $\delta^+ \leftarrow \delta^+ + searchIncr$  ▷ Increment  $\delta^+$ 
       $[stable, numMax] \leftarrow simulNet(\delta^+, \delta^-)$  ▷ Simulate
      if  $(stable) \& (numMax \leq maxNumMax)$  then
        store( $\delta^+, \delta^-$ )
      end if
    end while
     $recoveries \leftarrow recoveries + 1$ 
  end while
end procedure
```

---

Essentially, what Algo. 5 is doing is to continuously increment  $\delta^-$  until the network exhibits unstable/oscillatory behaviour, at which point it starts to increment  $\delta^+$  until the network is stable again, and then continues incrementing  $\delta^-$ , and so on. If at any time, a pair of  $\delta^+$  and  $\delta^-$  leads to an acceptable convergence (if the number of active nodes at the final iteration is smaller than or equal to some user-defined limit:  $maxNumMaxVal$ ), then that pair is stored.

### Activity Conditioned Weights

One of the main problems with the previous solutions is that they take too long to converge. The main reason for this lies in the fact that the connection weights (excitatory and inhibitory), and/or their rates of change, are not sensitive to the statistics of the node activities. In the current solution, we have implemented this dependence solely in the inhibitory weight, which we recompute at every new iteration using the following formula:

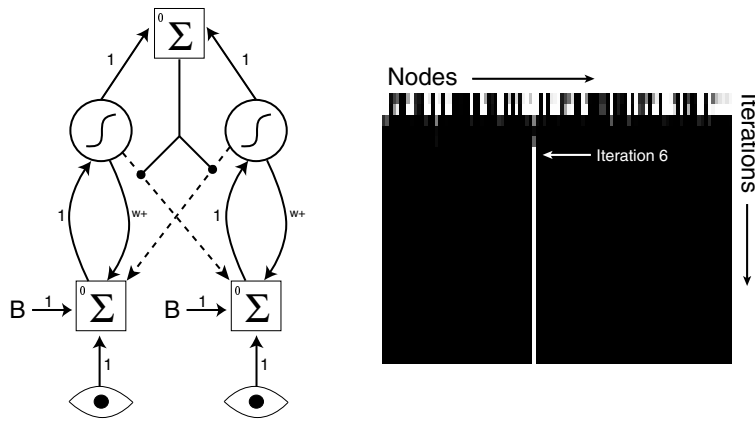


Figure 6.7:  $w^-$  depends on the sum of node activities.

$$w^- = \frac{c^-}{\text{sum}(\text{Activities})} \quad (6.3)$$

where  $c^-$  is a constant which depends primarily on the number of nodes and  $\text{sum}(\text{Activities})$  represents the sum of node activities. One effect of making  $w^-$  inversely proportional to  $\text{sum}(\text{Activities})$  is that as the activity of non-winners gradually decreases,  $w^-$  increases, thus keeping the inhibitory influence strong and accelerating convergence. So, the current solution depends on two parameters:  $w^+$  (an auto-excitatory connection weight, which remains constant throughout the competition) and  $c^-$ . The left-hand side of Fig. 6.7 illustrates a simple network that embodies the above mentioned dependence.

An example of the pronounced improvement this solution offers in terms of convergence speed is provided on the right-hand side of Fig. 6.7: the maximum-value is fully isolated by the sixth iteration. The network consists of 100 nodes and the following settings are used:  $w^+ = 4$ ,  $c^- = 7$ , a sigmoid-steepness of 3 and random inputs within the range  $[-3, +3]$ .

### Hierarchical Solution

One inconvenience with networks that isolate maximum-values by lateral inhibition is that they require significant amounts of wiring. If a network consists of  $n$  nodes then  $n(n - 1)$  inhibitory connections are required. For large values of  $n$  (e.g. large voting structures) this will often be impractical. One way around this problem is to implement the competitive solution in multiple layers. A two-layered solution

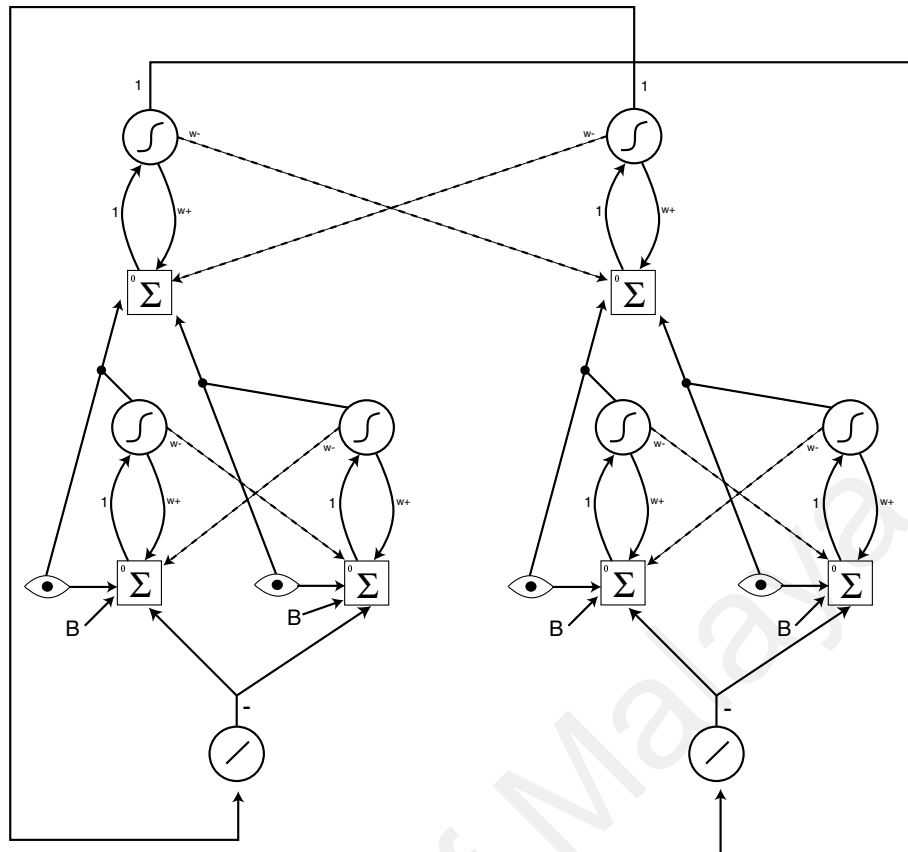


Figure 6.8: Hierarchical competitive network.

might divide the  $n$  nodes into  $g$  groups at the first layer. At the first layer then, there will be  $g$  different competitive processes<sup>3</sup> taking place in parallel. After each group of nodes has isolated its maximum, the results are passed on to the second layer (of  $g$  nodes), where a final competition takes place in order to isolate the absolute maximum. A simplified<sup>4</sup> network that implements this idea is demonstrated in Fig. 6.8. Note how the activations of the first layer set the weights of the connections from the input-nodes to the second layer: in this way, when the activation of a first-layer node is zero, the relevant input is effectively not passed, the converse being true when a first-layer node is fully active. Note also how the results of the second layer influence the first layer: the winning node of the second layer inhibits all the first-layer groups corresponding to the losing nodes.

<sup>3</sup>These processes can be implemented by any of the lateral inhibition networks described above.

<sup>4</sup>The first layer consists of four nodes only and synchronization information is omitted (it might be useful to delay the second layer's computation until the first layer has almost reached convergence).



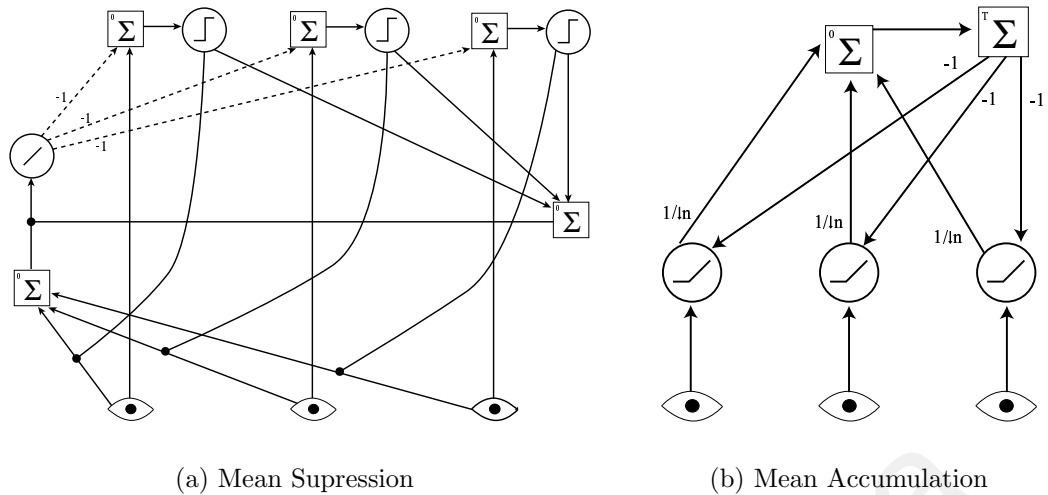


Figure 6.9: Mean based networks.

### 6.5.2 Mean Based

Lateral-inhibition is not the only way to isolate maximum-values in neural networks. An entirely different approach is based on the mean of node activations (see Huang et al. 1995 for a related approach).

One mean-based approach, which might be entitled mean-suppression, finds the mean activation, suppresses those nodes whose activations are below the mean, finds the mean of the surviving (non-suppressed) activations, and so on. After several iterations, the network converges onto the maximum-value. The diagram in Fig. 6.9(a) illustrates a network which implements this idea. The summation node on the right effectively counts the number of threshold units which are firing (i.e. counts the number of nodes whose activations are superior to the mean), and uses this value for setting the weight of the connection leaving the summation unit on the left, so that the mean is calculated (e.g. if three nodes are non-suppressed, then the weight should be  $1/3$ ).

Another mean-based approach, which we wish to propose and which cannot identify the particular node with the maximum-value, but can identify what that maximum-value<sup>5</sup> is might be denoted as mean-accumulation. The idea here is to compute the mean activation, add this result to some accumulating sum, suppress the activations that fall below the accumulating mean, calculate a new mean, add

<sup>5</sup>This might be useful for normalization purposes.

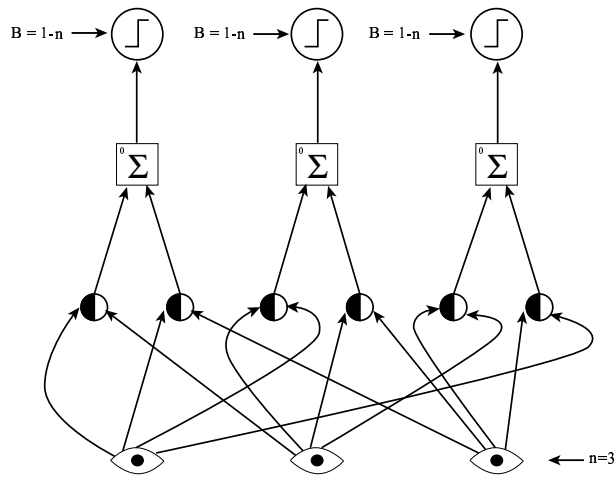


Figure 6.10: Paired comparisons.

this to the accumulator, and so on. After several iterations, the accumulator should converge to the maximum-value. Figure 6.9(b) depicts a network that implements this concept. The downward arrow at the  $1/n$  averaging weight indicates that  $n$  is decayable. In this context, an extension of the network can be envisaged where the averaging weight is considered to be  $1/c$ , where  $c$  represents the number of active units at the first layer (i.e. semi-linear nodes), which in turn can be computed from the sum of thresholded (i.e.  $1/0$ ) semi-linear inputs.

### 6.5.3 Paired Comparisons

We end our brief discussion of maximum-value networks with possibly the simplest approach of all: paired comparisons (see Huang et al. 1995 for a particular version). In this type of solution, each activation is compared to every other activation, and for each one that is found to be smaller, a count for the node under consideration is incremented. The node that exhibits the most increments, and therefore that has the largest number of nodes with smaller activations compared to it, will be the only node matching the negative bias (determined by the size of the network) and will thus fire, indicating the maximum-value. Figure 6.10 illustrates a simple and fast (single iteration) architecture implementing this idea.

#### 6.5.4 Advantages and Disadvantages

All of the maximum-value networks presented above have advantages and disadvantages. They were all presented based on the notion that each one of them has at least one context in which it supersedes the others. Table 6.1 compares the networks along various dimensions, namely: 1) number of nodes required, 2) number of connections, 3) convergence speed, 4) parameters, 5) stability and 6) usage of dynamic weights.

A brief look at Table 6.1 will probably reveal to us that, overall, the “Mean Suppression” network seems to be the most advantageous since it is relatively cheap regarding nodes and connections, it converges in a small number of iterations, requires no parameters, is not sensitive to input-range and has no instability issues. One possible down-side is that it requires dynamic weights. On the other hand, if we are looking for the fastest network, then we would have to choose the “Paired Comparisons” network since it invariably converges in one iteration. The downside of this network is that it is the most expensive of all in terms of nodes and connections. It might be argued by some, that the mean-based and paired-comparison networks are unrealistic from a biological perspective, and that if biological plausibility is what is being sought, then the lateral inhibition networks are the most suitable. The mean accumulation network has a special place in this discussion seeing that it does not isolate the node that represents the maximum, but rather just computes the maximum value itself. In its naive form it converges quite slowly, but when the averaging weight is allowed to decay, convergence is greatly accelerated. This network is particularly useful for normalization purposes and shares many of the strengths of the “Mean Suppression” network, i.e.: it is cheap in terms of nodes and connections, requires no parameters, is not sensitive to input-range and has no instability issues. It also has the extra advantage that it does not require dynamic weights.

Considering the various lateral-inhibition networks alone, one might observe that the third network (weights dependent on activity nodes) is the overall winner, e.g.: it is the fastest and most stable network. However, the second network has the

advantages that it requires fewer connections, and its dynamic weights are not dependent on inputs, but instead depend on intrinsic factors ( $\delta^+$  and  $\delta^-$ ), which from a biological perspective, might be the product of evolution. These two different types of dynamic-weights (on one side, weights that change according to external input and on the other side, weights that change due to internal/intrinsic dynamics) may have profound effects on their implementability, whether in artificial hardware or in biological networks. Finally, the first lateral-inhibition network (no dynamic weights) has simplicity as its main advantage.

University of Malaya

Table 6.1: Comparing maximum-value networks.

Network <sup>a</sup>	Num. Nodes <sup>b</sup>	Num. Connections <sup>c</sup>	Converge <sup>d</sup>	Parameters	Instability	RS <sup>e</sup>	Dyn. Weights
Lat. Inhib. I	$2n$	$n^2 + n$	$\approx 5^f$	2	High	Yes	No
Lat. Inhib. II	$2n$	$n^2 + n$	$\approx 25$	4	Medium	Yes	Yes
Lat. Inhib. III	$2n + 1$	$2n^2 + n$	$\approx 6$	2	Low	Yes	Yes
Mean Suppress.	$3n + 3$	$6n + 2$	$\approx 6$	None	None	No	Yes
Mean Accumul.	$n + 2$	$2n + 1$	$\approx 270^g$	None	None	No	No
Paired Comp.	$n^2 + 2n$	$3n^2 - 2n$	1	None	None	No	No

<sup>a</sup>Lateral Inhibition Networks: I) constant weights, II) constant weight change factors and III) weights dependent on the sum of activations.

<sup>b</sup>This column refers to the total number of nodes in the network, excluding the input nodes. The letter  $n$  refers to the number of input nodes.

<sup>c</sup>Considering Lat. Inhib. networks I and II, and assuming that nodes are regularly spaced out, where  $d$  corresponds to the horizontal/vertical distance between the centers of two neighboring nodes, the total wiring length used by this architecture can be neatly expressed by  $L = 2dn + d \sum_{z=1}^n \sum_{i=1}^n \sqrt{1 + |z - i|}$ . Refer to Appendix B for an explanation.

<sup>d</sup>For  $nodes = 100$ , uniformly distributed random inputs within the range  $[-3, +3]$  and *good* parameter choices. The values in this column refer to the iteration at which convergence takes place.

<sup>e</sup>Range sensitivity. Range refers here to the minimum and maximum input values. A network with range sensitivity is one that performs well for some ranges and poorly for other ranges.

<sup>f</sup>Although convergence is fast here, the network is incapable, in most cases, to isolate a single maximum-value.

<sup>g</sup>If the averaging weight is allowed to decay (e.g.  $w \leftarrow 3w/4$ ) convergence can occur much earlier (e.g. iteration  $\approx 17$ )

## 6.6 Correspondence Vote Normalization

Before presenting the full neural architectures for pose estimation, there is one last detail that needs to be addressed: normalization. When models and images are being put “side by side” in order to determine the transformation that relates them, the ensuing number of detected correspondences can vary significantly. In some cases, maybe only a handful of correspondences are detected, while in other cases, maybe thousands or more are detected. This is an important issue for the lateral-inhibition networks discussed above, since they are not completely independent of input-statistics. More specifically, as indicated in Table 6.1 the lateral-inhibition networks are sensitive to input-range. If the data which a network is processing lies outside its preferred range the result might for example be oscillatory instability. An obvious solution to this is to modulate the transformation specific nodes that sum correspondences, according to the maximum number of correspondences in any one of the nodes. This is one of the situations where the “Mean Accumulation” network is useful. An alternative to using a lateral-inhibition network with a pre-normalization step, is to simply use either a “Mean Suppression” or a “Paired Comparisons” network.

Having described our neural representations for images/models, correspondences, voting structures and vote “sharpening”, we can finally proceed with the complete architectures.

## 6.7 Constrained Estimation

To recapitulate, constrained estimation consists of those cases where single correspondences provide sufficient information to vote for one particular transformation consisting of two unknowns. So if we are dealing with objects which only translate, then each correspondence votes for one particular  $(T_x, T_y)$  pair. Likewise, if objects only scale and/or rotate, then a single correspondence votes for one particular  $(s, \theta)$  pair. Figure 6.11 depicts a simplified but general<sup>6</sup> network for the constrained case

---

<sup>6</sup>Recall that what makes a particular network specific to a particular group of transformations (e.g.  $(T_x, T_y)$  or  $(s, \theta)$ ) is the specific pattern of connections between correspondence detectors and

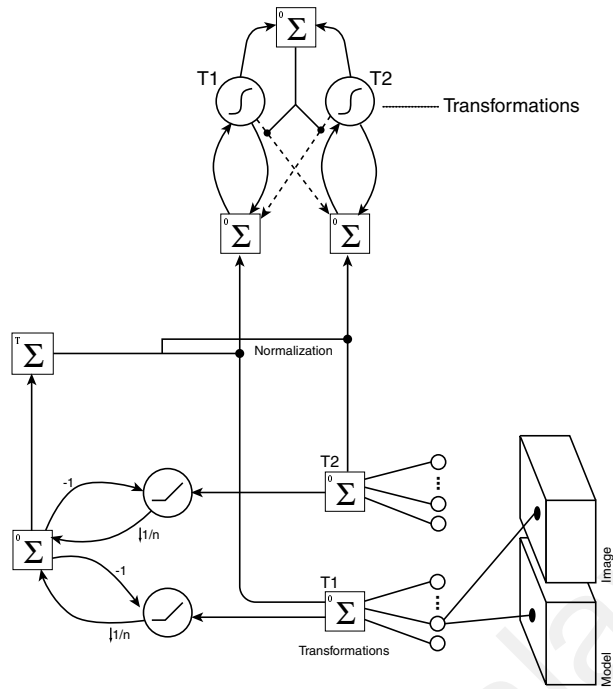


Figure 6.11: ANA for constrained pose estimation.

(notice how each correspondence detector outputs to only one transformation) which makes use of Lateral-Inhibition (activity-sum dependent) and Mean Accumulation networks. The Mean Accumulation network normalizes the output of the nodes that sum correspondences by dividing this output by the accumulative mean. Even when non-decaying averaging weights are used in the Mean Accumulation network, which results in a slow convergence towards the real maximum-value, the lateral-inhibition network can function virtually insensitive to input-range. Experiments were performed with parameters  $w^+ = 4$  and  $c^- = 5$ , which function adequately for a range of  $[0, +1]$ . It was observed that even for ranges as low as  $[0, +10^{-200}]$  the network still finds the maximum value in a small number of iterations and without any signs of instability.<sup>7</sup>

## 6.8 Unconstrained Estimation

As already mentioned, the unconstrained case originates because similarity transformations consist of four unknown parameters, while single correspondences pro-

---

<sup>7</sup>Note that the largest range that needs to be considered is  $[0, +1]$  seeing that one can assume that the weights from correspondence detectors to summation nodes reflect the largest number of correspondences a summation-node can receive ( $1/\max\text{Corresp}$ ).

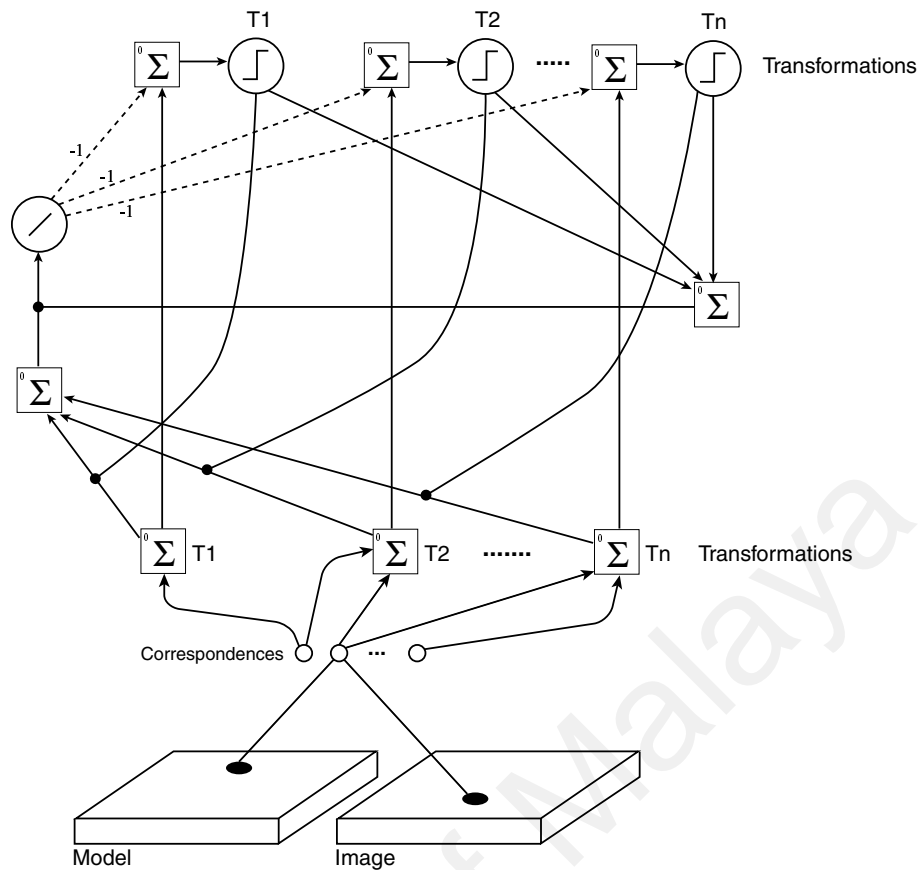


Figure 6.12: ANA for unconstrained pose estimation.

vide sufficient information to constrain only two unknowns. Figure 6.12 provides a simplified illustration of an ANA that is applicable to this case. Notice how each correspondence detector can vote for more than one transformation node (the so called “voting manifolds”). The vote sharpening is implemented through a Mean Suppression network.

Figure 6.13 depicts voting-manifolds more realistically, for the similarity transformation case, when two voting structures (one for  $T_x$  and the other for  $T_y$ ) are used. A single correspondence detector is shown, which activates two voting manifolds, one in the  $T_x$  structure and the other in the  $T_y$  structure. The shape of the manifolds is determined by Eq. 4.11. The voting-structure cells consist of summation nodes and the maximum-value network for vote sharpening is not depicted.

In this implementational chapter (see Fig. 2.1) the pure algorithmic approaches of the previous chapter were translated into artificial neural architectures, which included: feature columns, logical conjunctions, summation nodes, competitive layers and other components. Various architectures for vote sharpening (maximum value



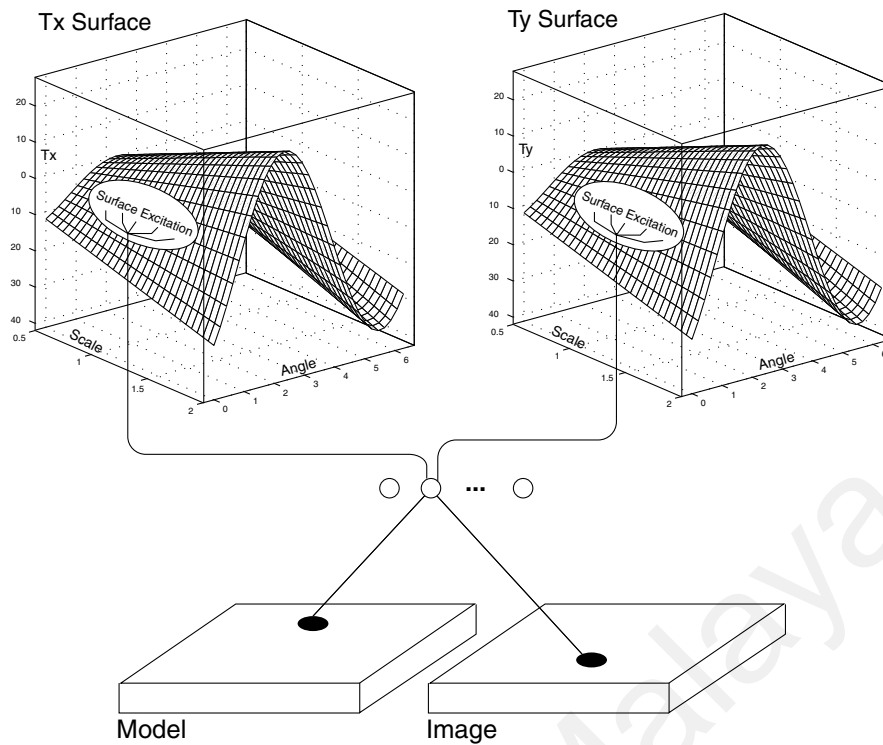


Figure 6.13: Correspondences excite/vote-for  $T_x$  and  $T_y$  surfaces.

networks) were tested and compared. Different neural architectures were designed for constrained and unconstrained estimation. Since in this chapter, we have demonstrated the neural implementability of the correspondence distribution approach, the biological plausibility argument has been somewhat strengthened. In the following chapter, several architectural (spatial and combinatorial) issues regarding the architectures will be investigated in greater depth.

# Chapter 7

## Architectural Analysis

Having described our algorithmic approach to pose-estimation, having studied its estimation accuracy, and having considered its embodiment in artificial neural architectures, we are in a good position to analyze several architectural issues surrounding these neural implementations. The investigations will be centered on the various combinatorial explosions that affect the feasibility of the architectures and on optimizing architectural layouts for minimizing relevant cost functions.

### 7.1 Combinatorial Explosions

The term combinatorial explosion denotes the fact that small increases in one factor (e.g. map resolution) lead to disproportionately large increases in another factor (e.g. number of outputs from map-nodes). The types of combinatorial explosions considered here are critical because of how they affect the feasibility/usefulness of solutions. This can take two forms, depending on whether sequential or parallel computers are being used. In the case of sequential machines, the combinatorial explosions have a critical effect on the time it takes for a pose to be computed: in most cases, solutions are of little or no practical value, if poses are computed in anything but real-time. For parallel machines, the combinatorial explosions have a critical effect on the amount of space that is required for the hardware that implements the algorithms.<sup>1</sup>

---

<sup>1</sup>Parallel machines do not solve the combinatorial problems of their sequential counterparts, instead they convert these problems from temporal to spatial domains.

### 7.1.1 Nodes

The only nodes whose total numbers may raise combinatorial issues are correspondence detectors. The number of outputs that a correspondence detector can possess influences the number of detectors that are required. For simplicity we will consider two cases: 1) detectors that can have any number of outputs and 2) detectors with one only output. Another influential factor is whether we are dealing with constrained or unconstrained estimation. The following three equations summarize the main cases:

$$\text{Multiple Outputs} \quad \text{Constrained/Unconstrained} \quad c = tn^2 \quad (7.1)$$

$$\text{Single Outputs} \quad \text{Constrained} \quad c = tn^2v \quad (7.2)$$

$$\text{Single Outputs} \quad \text{Unconstrained} \quad c = tn^2v \prod_{i=1}^z r(u_i) \quad (7.3)$$

where  $t$  refers to the number of feature-types,  $n$  refers to the number of nodes in any map (for one feature-type),  $v$  represents the number of voting structures (assumed to be of equal dimensions). In Equation 7.3, voting-manifolds are considered, where  $z$  refers to the number of unconstrained variables,  $u_i$  refers to the unconstrained-variable  $i$  and  $r(x)$  refers to the resolution of variable  $x$ .

Figure 7.1 illustrates how Equation 7.1 is intuitively derived from the corresponding architecture. For the sake of clarity, only some nodes and connections have been depicted. Notice how some of the correspondence detectors have multiple outputs (e.g. those labeled by *ii* and *iii*) and how this explains why single-output detectors lead to a situation where more nodes are required. A multiple-output correspondence detector with  $x$  outputs in one case, calls for  $x$  single-output correspondence detectors in the other case. All other equations concerning the combinatorial issues of nodes and connections were derived in a similar manner to what is illustrated in Fig. 7.1.

It is interesting to note that, when dealing with similarity transformations and their four unknowns, the worst-case scenario expressed in Equation 7.3 leads to the

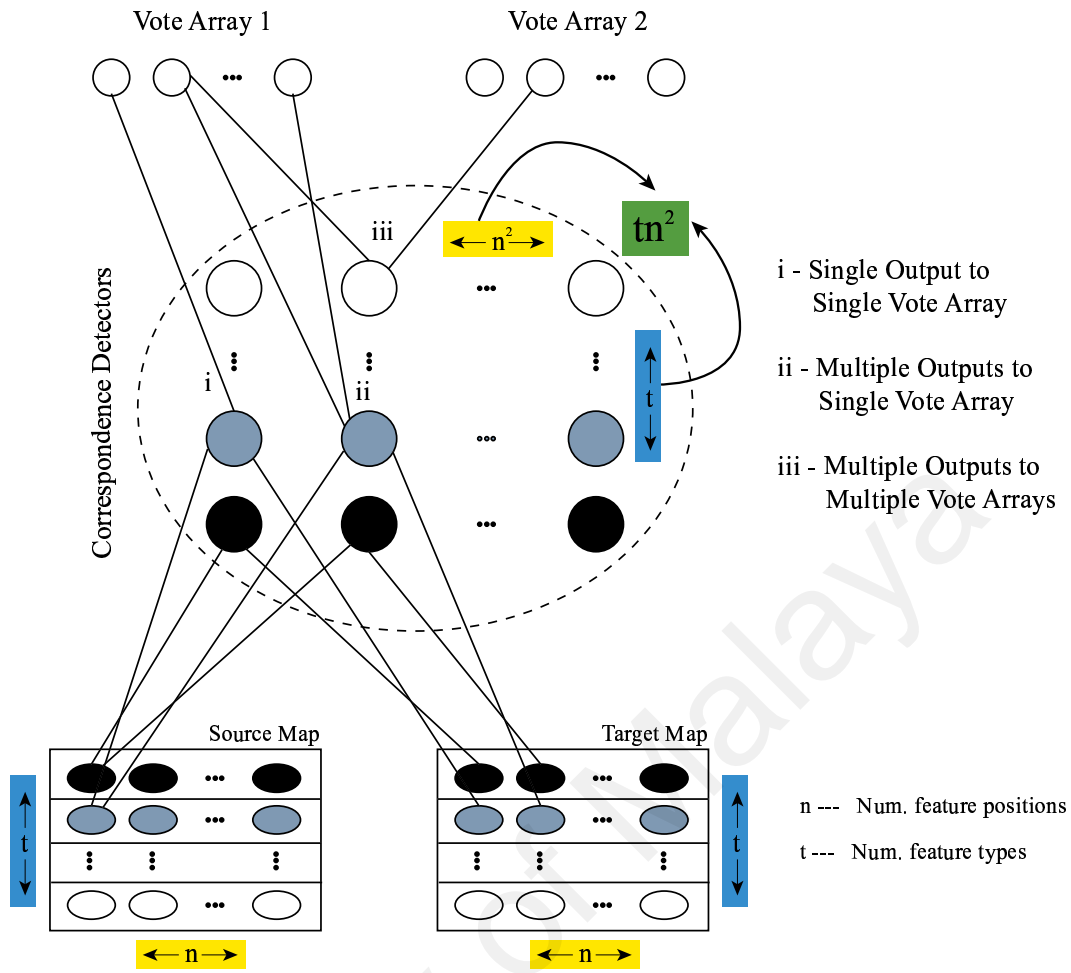


Figure 7.1: Example equation derivation.

same combinatorial problems suffered by solutions that use doublet-correspondences (when correspondence detectors can have single outputs only). If we are using doublets for estimating similarity transformations, there are no unconstrained variables, thus the number of detectors required is:

$$d = t \left[ \frac{n(n-1)}{2} \right]^2 v \quad (7.4)$$

which is smaller than what results from Equation 7.3 if  $z = 2$  and  $r(u_1) = r(u_2) = n$  (i.e.  $tn^4v$ ). From this we can conclude that, for cases involving unconstrained variables, it is quite crucial that correspondence detectors be able to have multiple outputs. This leads to an interesting hypothesis regarding biological implementations as we shall see in Chapter 8, or more specifically in Section 8.1.6.

As can be seen by the equations above, various factors ( $t$ ,  $n$ ,  $v$  and the number and resolution of unconstrained variables) can conspire towards making the

number of required correspondence detectors prohibitive. If so, the correspondence-distribution approach is called into question. Fortunately, these factors can be kept within acceptable bounds without sacrificing estimation accuracy.

Though, as we have seen here, there are certain implementability issues concerning correspondence detectors, the connections between nodes also provide a rich set of combinatorial problems.

### 7.1.2 Connections

Node connections can essentially be seen from two perspectives: outputs from nodes and inputs to nodes. It is important to consider these two cases, rather than just discuss the total number of connections between “layers” because there are physical constraints regarding the number of outputs/inputs a node can send/receive. The discussion will include all connections starting from map outputs and ending at vote inputs. The connections involved in sharpening networks (e.g. lateral-inhibition networks) can be treated independently and so will not be discussed in this section.

#### Outputs

Regarding map-node outputs there are three cases to consider: 1) correspondence detectors with multiple outputs, 2) constrained estimation using correspondence detectors with single-outputs and 3) unconstrained estimation using detectors with single-outputs. These three cases are expressed in the following equations:

$$\begin{array}{ll} \text{Multiple Outputs} & \text{Constrained/Unconstrained} \end{array} \quad o = n \quad (7.5)$$

$$\begin{array}{ll} \text{Single Outputs} & \text{Constrained} \end{array} \quad o = nv \quad (7.6)$$

$$\begin{array}{ll} \text{Single Outputs} & \text{Unconstrained} \end{array} \quad o = nv \prod_{i=1}^z r(u_i) \quad (7.7)$$

where  $n, v, z, u_i$  and  $r(x)$  are as in Equations 7.1, 7.2 and 7.3.

When considering the outputs of correspondence detectors, by default it is only of interest to consider detectors with multiple outputs. If so, there are two conditions

to consider: 1) constrained estimation and 2) unconstrained estimation. In the first case the number of outputs is determined by  $o = v$ . In the second case the number of outputs is

$$o = v \prod_{i=1}^z r(u_i) \quad (7.8)$$

where  $v$ ,  $z$ ,  $u_i$  and  $r(x)$  are as before.

## Inputs

The inputs of correspondence detectors are by definition limited to 2. The number of inputs that vote-nodes receive is a slightly more complex issue since not every node receives the same number. This is mainly due to border effects, e.g: in the case of constrained  $(T_x, T_y)$  estimation, the larger  $T_x$  and  $T_y$  are, the fewer correspondences exist that represent them because they tend to require features that lie beyond the image map boundaries. Having said this, it is possible to compromise on an average which can be expressed by a quotient between the total number of votes that can be cast and the total number of voting cells. For the constrained case, this is expressed by

$$p \approx \frac{tn^2}{\prod_{i=1}^d r(y_i)} \quad (7.9)$$

where  $t$ ,  $n$  and  $r(x)$  are as before,  $d$  refers to the number of dimensions in the voting structure and  $y_i$  refers to a particular voting dimension  $i$  (e.g.  $T_x$ ). For the unconstrained case, this is expressed by a generalization of Equation 7.9:

$$p \approx \frac{tn^2 \prod_{i=1}^z r(u_i)}{\prod_{i=1}^d r(y_i)} \quad (7.10)$$

### 7.1.3 Constraints

As already mentioned, the main reason for considering combinatorial explosions is that they affect the feasibility of solutions. Various factors can be included in the

notion of feasibility, e.g.: volume of space, number of inputs/outputs per node, number of layers, and so on. In this subsection, we are solely concerned with feasibility from the point of view of a node's inputs and outputs.

### Vote Inputs

At a first glance of Equation 7.10, one might think that the number of vote-cell<sup>2</sup> inputs is problematic. However, a closer look shows that the resolutions of unconstrained variables in the numerator are canceled out by denominator factors. Furthermore, some of the quadratic  $n$  term is canceled by the resolutions of voting-structure constrained variables. Using the similarity transformation group as an example, the average number of inputs for a vote node is:

$$\begin{aligned}
 p &\approx \frac{tn^2r(\mathcal{X})r(\theta)}{r(\mathcal{X})r(\theta)r(T_x)r(T_y)} \\
 p &\approx \frac{tn^2}{r(T_x)r(T_y)} \quad \text{and if } r(T_x) = r(T_y) = 2\sqrt{n} \\
 p &\approx \frac{tn^2}{2\sqrt{n}2\sqrt{n}} \\
 p &\approx \frac{tn^2}{4n} = \frac{tn}{4} \tag{7.11}
 \end{aligned}$$

since a reasonable choice for  $r(T_x)$  and  $r(T_y)$  is  $2\sqrt{n}$  (to approximately allow the range  $[-\sqrt{n}, +\sqrt{n}]$ ). Though it might appear that the problem has vanished,  $p \approx (tn)/4$  is actually excessive in many instances, and so we are back to our initial suspicion. If physical constraints dictate that the maximum number of inputs any node can receive is  $10^4$ , and if  $n = 10^4$  and  $t = 10^2$  then  $p = 10^6/4$ , which greatly exceeds the constraints. How can these constraints and parameters be reconciled? One simple, albeit somewhat expensive solution, is to have multiple nodes for each vote cell, so that the total number of correspondences belonging to each cell can be divided/distributed among the multiple nodes. The activities of these nodes can then be summed at a subsequent "layer" in order to give the final vote. Using

---

<sup>2</sup>It is important for this discussion to keep in mind the difference between "vote cell" and "vote node". A vote cell refers to a particular transformation (e.g.  $T_x = +5$  and  $T_y = -8$ ) regardless of whether it is represented by one or more neural nodes.

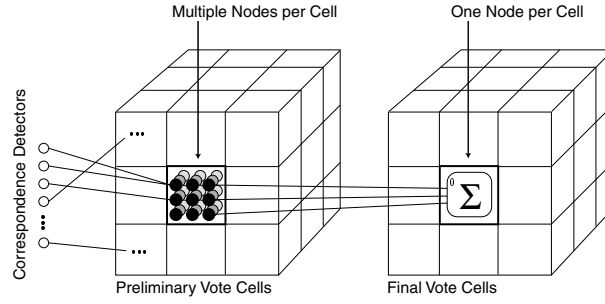


Figure 7.2: Dealing with excessive vote-cell inputs.

the above example where the maximum number of node-inputs is  $10^4$ , and where  $p = 10^6/4 = 25 \cdot 10^4$ , one can see that the average number of vote-inputs is 25 times larger than the constraint. Therefore, on average, in order to satisfy the constraint and the choice of  $n$  and  $t$ , each voting-cell requires about 25 nodes. Figure 7.2 depicts this two-layered solution to the problem of excessive vote-node inputs.

In this “multiple nodes per vote-cell” solution, and assuming unconstrained estimation, an approximation of the total number of voting nodes required can be calculated by:

$$\begin{aligned}
 \text{Nodes} &\approx \text{ceil} \left( \frac{\text{avrgIn}}{\text{maxIn}} \right) \prod_{i=1}^d r(y_i) \\
 \text{Nodes} &\approx \text{ceil} \left( \frac{tn^2}{\text{maxIn}} \frac{\prod_{i=1}^z r(u_i)}{\prod_{i=1}^d r(y_i)} \right) \prod_{i=1}^d r(y_i)
 \end{aligned} \tag{7.12}$$

where  $\text{avrgIn}$  refers to the average number of inputs per voting-cell,  $\text{maxIn}$  refers to the constraint regarding the maximum number of inputs a node can have,  $\text{ceil}(x)$  refers to the ceiling function (e.g. if  $x = 1.2$  then  $\text{ceil}(x) = 2$ ), and the other terms are as in previous equations. For the constrained case, substitute  $\prod_{i=1}^z r(u_i)$  for 1 in Equation 7.12.

### Map-Node Outputs

The same problem can occur in the context of map-node outputs. For the case where correspondence detectors are restricted to single-outputs and when we are concerned with unconstrained estimation, the number of map-node outputs ( $\text{out}$ ) is de-



terminated by Equation 7.7. If the physical constraint  $maxOut$  is such that  $maxOut < out$  then there is a clash between the requirements and the constraints. Similarly to the previous adaptation, this conflict can be overcome by having multiple nodes per map-coordinate, which can be determined by  $nodes = ceil(out/maxOut)$ . If so, the total number of nodes required for representing both model and image maps is:

$$Nodes = 2 \text{ ceil} \left( \frac{nv \prod_{i=1}^z r(u_i)}{maxOut} \right) tn \quad (7.13)$$

Having shown how certain physical constraints can clash with the number of required nodes and connections, and how architectural adaptations can be devised to deal with them, it is time to consider another approach: containment. If we reduce the number of nodes and connections that are required, then maybe we can satisfy the physical constraints without having to resort to multiple nodes per vote-cell or other adaptations.

#### 7.1.4 Containing Explosions

Fortunately, some containment of the combinatorial problem can be achieved by simply using a degree of moderation in regards to the resolutions of feature maps ( $n$  and  $t$ ) and voting structures ( $r(u_i)$  and  $r(y_i)$ ). Resolutions can be kept within feasible bounds whilst maintaining a satisfactory level of estimation accuracy.

A stronger approach to reducing (or modifying the impact of) combinatorial issues, involves segregating the constrained variables (e.g.  $T_x$  and  $T_y$ ). If the constrained variables are  $T_x$  and  $T_y$  then a single correspondence defines a value for  $T_x$  and another one for  $T_y$ . This *boundedness* of  $T_x$  and  $T_y$  can be represented in a 2D data structure. In the stronger approach to containing (or modifying) combinatorial problems, we propose collapsing this 2D structure into two 1D structures, one for representing the first constrained variable (e.g.  $T_x$ ) and the other for representing the second constrained variable (e.g.  $T_y$ ). Clearly this reflects a loss of voting resolution and thus tends to lead to a reduction in estimation accuracy. However, as will be shown, this reduction is not significant, and the resulting combinatorial containments compensate for it in most cases.

In order to demonstrate the approach, we will make use of the problem of unconstrained estimation of similarity transformations (i.e. estimating  $T_x$ ,  $T_y$ ,  $s$  and  $\theta$ ). The implementation in Chapter 4 keeps  $s$  and  $\theta$  unconstrained, and constrains  $T_x$  and  $T_y$  using the information provided by single correspondences. Because the constrained variables are bound, a single four dimensional voting structure is used. Since the solution proposed here involves *unbinding* the constrained variables, the  $4D$  voting structure collapses into two  $3D$  structures:  $[T_x, s, \theta]$  and  $[T_y, s, \theta]$ .<sup>3</sup>

The above mentioned collapse has repercussions that go beyond the voting structures themselves. It affects the whole architecture, including the feature maps and the correspondence detectors. Recall Equation 4.11 for estimating horizontal translations:  $T_x = x_2 - s(x_1 \cos \theta - y_1 \sin \theta)$ . Notice that the  $y_2$  coordinate ( $y$  coordinate of the image map feature) is not present/required. This means that, for the case of  $T_x$  estimation, the  $2D$  image map can be collapsed into a  $1D$  map representing only the  $x$ -coordinates of features. The collapse is performed by summing, for each  $x$ -coordinate, the activities of all  $y$ -coordinates (e.g. if  $x = 2$  has three active features at  $(x = 2, y = 1)$ ,  $(x = 2, y = 5)$  and  $(x = 2, y = 6)$  then the second node in the  $1D$   $T_x$  structure will be active with a value of 3). A similar reasoning applies to the collapse of the  $2D$  image map into a  $1D$   $T_y$  representation<sup>4</sup>. This summation of activities is important so that it preserves information regarding the number of correspondences that contribute to a particular coordinate of the  $1D$  structure.

The image map collapse and the summation of activities along a particular coordinate, entail an important modification in order for the information to be adequately propagated to voting structures. Correspondence detectors require a crucial “enhancement”: they need to be multipliers rather than simple conjunctions (e.g.  $2 \times 3 = 6$ ). Figure 7.3 summarizes the main architectural modifications proposed so far.

In order to get a better grasp of the advantages/disadvantages of these modifications, it is important to see how they affect the numbers of nodes and connections

---

<sup>3</sup>Architectures that use a  $4D$  voting-structure will from this point onwards be abbreviated to “ $4D$  architectures” and those that use two  $3D$  voting structures will be abbreviated to “ $3D$  architectures”.

<sup>4</sup>Recall that the  $T_y = y_2 - s(x_1 \sin \theta + y_1 \cos \theta)$  expression does not include  $x_2$ .

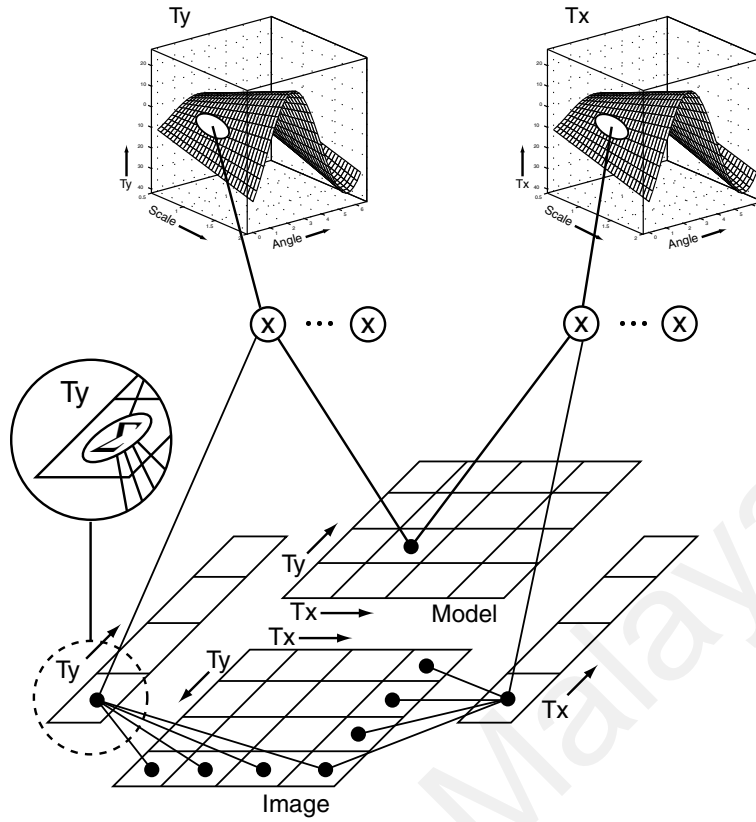


Figure 7.3: Two 3D voting structures.

that are required. The following equations summarize the various cases that stem from the architecture illustrated in Fig. 7.3:

$$\text{Map Nodes} \quad 2t(n + \sqrt{n}) \quad (7.14)$$

$$\text{Model Map Outputs} \quad 2\sqrt{n}r(s)r(\theta) \quad (7.15)$$

$$\text{Image Map Outputs} \quad nr(s)r(\theta) \quad (7.16)$$

$$\text{Correspondence Detectors} \quad 2tn\sqrt{n}r(s)r(\theta) \quad (7.17)$$

$$\text{Vote Nodes} \quad 2r(T)r(s)r(\theta) \quad (7.18)$$

$$\text{Vote Inputs} \quad \frac{tn\sqrt{n}}{r(T)} \quad (7.19)$$

where  $t$  refers to the number of feature-types,  $n$  refers to the number of map features for any one feature-type,  $r(x)$  refers to the resolution of variable  $x$  and  $r(T)$

refers to the resolution of both  $T_x$  and  $T_y$  (i.e.  $r(T) = r(T_x) = r(T_y)$ ).

Starting with the number of map nodes (Equation 7.14) and comparing this to the requirements of a solution that uses a  $4D$  voting structure ( $m = 2tn$ ) one can see that our situation has worsened somewhat<sup>5</sup>:

$$\frac{4D}{3D} = \frac{2tn}{2t(n + \sqrt{n})} = \frac{n}{n + \sqrt{n}} < 1 \quad (7.20)$$

Equation 7.20 shows us that the  $4D$  architecture always requires fewer map nodes than the  $3D$  architecture and that the ratio of nodes is independent of the number of feature types. Since the limit (as  $n \rightarrow \infty$ ) of the sequence defined by equation 7.20 is 1, it is clear that the ratio of nodes becomes increasingly less significant for larger values of  $n$ . However, since practicality demands relatively modest values for  $n$ , the ratio might have to be taken into account when considering which architecture to employ.

In the  $4D$  architecture, the number of outputs per map node is  $m = nr(s)r(\theta)$ . This is identical to the outputs of image map nodes in the  $3D$  architecture. There is however a critical improvement regarding the outputs of model map nodes (Equation 7.15):

$$\frac{4D}{3D} = \frac{nr(s)r(\theta)}{2\sqrt{n}nr(s)r(\theta)} = \frac{n}{2\sqrt{n}} = \frac{\sqrt{n}}{2} \quad (7.21)$$

Since  $4D/3D$  diverges as  $n$  increases, the improvement can be very pronounced, even for modest  $n$ . However, one must admit that overall, this is still not a significant advantage since the architecture does not free the  $1D$  image maps from the requirement of  $nr(s)r(\theta)$  outputs.

Regarding correspondence detectors,  $4D$  architectures require  $tn^2r(s)r(\theta)$  nodes, which if divided by the number required by  $3D$  architectures leads to the following ratio:

$$\frac{4D}{3D} = \frac{tn^2r(s)r(\theta)}{2tn\sqrt{n}r(s)r(\theta)} = \frac{n}{2\sqrt{n}} = \frac{n\sqrt{n}}{2n} = \frac{\sqrt{n}}{2} \quad (7.22)$$

---

<sup>5</sup> $4D/3D$  represents the quotient between the relevant expressions for architectures that use  $4D$  voting structures and architectures that use two collapsed  $3D$  structures respectively.

Equation 7.22 demonstrates one of the main strengths of the  $3D$  architecture being proposed: it leads to significant savings in regards to correspondence detector nodes. The savings become increasingly pronounced as  $n$  increases, seeing that the equation represents a divergent sequence (as  $n \rightarrow \infty$  so does  $4D/3D \rightarrow \infty$ ).

The  $3D$  architecture leads to similar savings in regards to vote-nodes as demonstrated by the ratio

$$\frac{4D}{3D} = \frac{[r(T)]^2 r(s) r(\theta)}{2r(T)r(s)r(\theta)} = \frac{r(T)}{2} \quad (7.23)$$

which as the previous equation represents a divergent sequence, but this time as a function of  $r(T)$ .

The final factor to consider concerns the average number of inputs that vote-nodes receive, where the ratio between  $4D$  and  $3D$  architectures is defined by:

$$\frac{4D}{3D} = \frac{\frac{tn^2}{[r(T)]^2}}{\frac{tn\sqrt{n}}{r(T)}} = \frac{n}{r(T)\sqrt{n}} = \frac{\sqrt{n}}{r(T)} \quad (7.24)$$

If  $r(T)$  is substituted by  $2\sqrt{n}$ , which, as already mentioned, is a natural choice, then Equation 7.24 can be further simplified to

$$\frac{4D}{3D} = \frac{\sqrt{n}}{r(T)} = \frac{\sqrt{n}}{2\sqrt{n}} = \frac{1}{2} \quad (7.25)$$

which indicates that  $3D$  architectures require, on average, twice the number of vote-node inputs compared to  $4D$  architectures.

To summarize, although the  $3D$  architectures proposed here do have some extra-costs, there are two fundamental elements that make them worthwhile in many instances: critical savings in regards to correspondence detectors (see Equation 7.22) and significant savings in regards to vote-nodes (see Equation 7.23). Since these elements and the combinatorial issues that they pose can raise serious questions about feasibility, the savings provided by  $3D$  architectures tend not to be inconsequential.

There is still one aspect of  $3D$  architectures that requires further attention: estimation accuracy. Because these architectures discard some information (i.e. the *boundedness* of particular  $T_x$  and  $T_y$  estimates) and parallel to this, because the

Table 7.1: Comparison of estimation errors between  $4D$  and  $3D$  architectures.

Architecture	$T_x$	$T_y$	$s$	$\theta$
$4D$	0.033	0	0.028	0.027
$3D$	0.067	0	0.034	0.033

discrimination power of voting structures is reduced, it is valid to expect some reduction of estimation accuracy. Is this reduction large enough to justify forsaking  $3D$  architectures? It turns out that it is not, i.e: the reduction of accuracy is not significant. Table 7.1 depicts the mean errors (out of 30 tests) that resulted from applying both  $3D$  and  $4D$  architectures to synthetic patterns (with a PTC  $\approx 0.5$ ) under various random transformations. As one can see, there is a slight (expected) decrease in accuracy for  $3D$  architectures, which is however, for most practical purposes negligible, since the errors are far below the resolution of each transformation (e.g.  $3D_{err}(T_x) \rightarrow 0.067 \ll 1$  pixel).

It is important to note that the principle of collapsing the bounded variables  $T_x$  and  $T_y$  in order to achieve certain savings (e.g. a reduction of the number of correspondence detectors), which was presented here in the context of estimation within the similarity transformation group, is applicable to other transformation groups, e.g: constrained  $(T_x, T_y)$  estimation, constrained  $(s, \theta)$  estimation, and others. In the latter case of  $(s, \theta)$  estimation it is advantageous to convert model/image maps into polar representations (i.e. instead of representing features by their  $2D$  positions they should be represented by the lengths and angles of the vectors formed between the map origins and the features).

## 7.2 Structural Optimization

### 7.2.1 Introduction

In this context, structural optimization refers to the process through which a particular spatial configuration of nodes is found that maximizes/minimizes some structural function of the neural architecture: optimality is defined relative to a specific function. Usually *cost* functions are used, which means that optimization

attempts to find the neural architecture that *minimizes* the function. An example of a cost function might be the distance between related nodes (e.g. the distance between model nodes). Clearly, the type of architecture that minimizes this cost function is one that clusters related nodes as closely as possible. Any number of cost functions can be thought of, each one with its own uses in different contexts. We are going to be dealing with the *total wiring length* (TWL) cost function. Saving wiring length has important practical (artificial/hardware) consequences, e.g: saving space and energy and increasing computational speed. Furthermore, there is strong evidence that biological neural systems use this cost function in many of their architectural designs, and for the same reasons as those stated above. Refer to (Chklovskii et al., 2002) for evidence that biological neural systems use this cost function and see (Allman, 1999) for an interesting discussion regarding the brain's necessity to be energy efficient.

It might be important to stress here that although structural optimization has no direct effect on the functionality of the system (in the sense of estimation accuracy), the converse is not true, i.e: the types of transformations that are being estimated (e.g.  $(T_x, T_y)$ ,  $(s, \theta)$  and others) and their accuracies have a direct effect on the resulting optimal architectures, as will be seen.

## 7.2.2 Assumptions and Simplifications

In order to implement realistic optimization processes (in terms of convergence time) certain simplifications are required. Recall that the architectures involve map nodes (model and image), vote nodes, correspondence detectors, and the connections between these elements. Four main simplifications were used in our optimization experiments: 1) nodes were modeled as points, 2) connections were defined as immaterial (i.e. they could cut through nodes and other connections), 3) connections always took the shortest path (straight lines connecting nodes) and 4) correspondence detectors were collapsed into their corresponding vote nodes. Apart from simplifying the optimization process, these abstractions can be further justified by arguing that they are not as unrealistic as they might first appear. Regarding the

fourth abstraction, for example, if one recalls that correspondence detectors are simple conjunctions of two inputs, and that they are thus likely to be small components compared to vote nodes, and that resources are saved by keeping them close to their corresponding vote nodes, then one can conclude that this is not too different from collapsing them into their corresponding vote nodes. The narrowness and flexibility of biological connections (i.e. neurites) might in some ways equate to the “immaterial” and “straight-line” simplifications used here. And finally, representing nodes as points can easily be given an extra degree of realism by simply adding a minimum inter-node distance constraint.

### 7.2.3 Optimization Methods

The problem of finding the configuration of nodes that minimizes TWL for a particular architecture is not an easy problem. To get a feel for the difficulty, consider doing an “exhaustive search” through the space of all possible node configurations. Even a simple problem consisting of 5 model nodes, 5 image nodes and 9 vote nodes in a space of 25 cells<sup>6</sup>, and assuming that it takes one second to evaluate each configuration, will take more than 30,000 times the age of the Universe (according to recent estimates). Furthermore, the complexity of the problem increases exponentially (rather than polynomially) with  $n$  (where  $n$  is the number of nodes in the architecture). Therefore, it is fundamental that we find a useful optimization process, one that is fast and that can deliver useful answers (good local minima).

For initial experiments we devised a hybrid evolutionary algorithm which apart from the usual mutation and cross-over operators also employed what we call “subset optimization”. In subset optimization random subsets of nodes are selected and optimized by a fast exhaustive search. Algorithm 6 summarizes the approach.

Although the above approach does succeed in finding useful architectures convergence can be slow if the problems are too large and complex. Because of this, we experimented with a simpler more geometric solution: an elastic network (refer to Algo. 7). Refer to (Durbin & Willshaw, 1987) and (Goodhill & Willshaw, 1990)

---

<sup>6</sup>Each node can be placed in any unoccupied cell. The cells are organized in a  $5 \times 5$  array.



---

**Algorithm 6** Hybrid Evolutionary Optimization

---

```
procedure LAMARWIN(maxIterat)
  Initialize Population
  iterat ← 0
  while iterat ≤ maxIterat do
    Generate Offspring                                ▷ Mutate and Cross-Over
    Optimize Parents and Children                    ▷ Subset Optimization
    Select the x% best for the Next Generation
    iterat ← iterat + 1
  end while
end procedure
```

---

for other examples of elastic networks applied to combinatorial optimization problems. In our solution, nodes which are interconnected attract each other along their connections. Furthermore, in order to avoid nodes collapsing into each other, and to spread out the layouts (indirectly implementing the solidity/impenetrability of nodes up to a degree), a general repulsion force is applied to all nodes (each node repulses all others) which is inversely proportional to the square of the distance of the nodes being considered (refer to Fruchterman & Reingold 1991 for an example from the literature on graph layout optimization). It is important to point out that although this solution is not explicitly minimizing TWL it is doing so indirectly by attracting interconnected nodes: if two nodes are connected, they will be attracted to each other, bringing them closer together and consequently decreasing the length of the connection between them. Further evidence for the functional equivalence (or similarity) of these approaches comes from observing their outcomes when applied to the same problem (see Fig. 7.4).

Consider the problem of estimating shifts along a single dimension (e.g.  $T_x$ ). In this case we have one-dimensional model and image maps of  $n$  nodes, and a one dimensional vote representation of  $2n - 1$  nodes ( $[-(n - 1), +(n - 1)]$ ). Model and image nodes are connected to vote nodes in a way that reflects shift estimation: e.g. model node 2 and image node 5 are connected to vote node +3 because  $5 - 2 = 3$  represents a right shift of three positions. The left hand side of Fig. 7.4 shows an optimal architecture found by the Hybrid Evolutionary algorithm while the right hand side shows an optimal architecture found by the elastic net. As one can see both architectures share the same fundamental design principles:

---

**Algorithm 7** Elastic Structural Optimization

---

```
1: procedure ELASTICNET(net,  $\alpha$ ,  $\beta$ )                                ▷ In most cases  $\alpha = \beta = 0.1$ 
2:   while  $\neg stable$  do                                           ▷ Loop while the elastic net is unstable
3:      $a \leftarrow compAttract(net, \alpha)$                           ▷ Compute attraction forces
4:      $r \leftarrow compRepulse(net, \beta)$                             ▷ Compute repulsion forces
5:      $net \leftarrow updateNet(net, a, r)$                             ▷ Apply the forces
6:   end while
7: end procedure

8: procedure COMPATTRACT(net,  $\alpha$ )
9:   for all  $n \in net$  do                                           ▷ Scan all nodes
10:     $n.force \leftarrow [0\ 0]$                                        ▷ Initialize force
11:     $[n_x\ n_y] \leftarrow getPos(n)$                                   ▷ Node position
12:    for all  $c \in n$  do                                           ▷ Scan the node's connections
13:      $v \leftarrow getVote(net, n, c)$                                 ▷ Get a connected vote-node
14:      $[v_x\ v_y] \leftarrow getPos(v)$                                 ▷ Vote-node position
15:      $f \leftarrow \alpha [(v_x - n_x)\ (v_y - n_y)]$                 ▷ Compute a force
16:      $n.force \leftarrow n.force + f$                                   ▷ Add force
17:   end for
18: end for
19: end procedure

20: procedure COMPREPULSE(net,  $\beta$ )
21:    $nodes \leftarrow getNodes(net)$ 
22:    $nodes.repulse \leftarrow 0$                                        ▷ Initialize repulsive forces
23:    $numNodes \leftarrow countNodes(nodes)$ 
24:   for  $n_a \leftarrow 1, numNodes - 1$  do                            ▷ Scan nodes
25:     for  $n_b \leftarrow n_a, numNodes$  do                            ▷ Scan nodes
26:       $[x_1\ y_1] \leftarrow getPos(nodes(n_a))$                        ▷ First node position
27:       $[x_2\ y_2] \leftarrow getPos(nodes(n_b))$                        ▷ Second node position
28:       $f \leftarrow [(x_1 - x_2)\ (y_1 - y_2)]$                          ▷ Preliminary force
29:       $d \leftarrow \sqrt{f(1)^2 + f(2)^2}$                              ▷ Inter-node distance
30:       $force \leftarrow \beta (f/d^2)$                                   ▷ Repulsive force
31:       $nodes(n_a).repulse = nodes(n_a).repulse + force$ 
32:       $nodes(n_b).repulse = nodes(n_b).repulse - force$ 
33:     end for
34:   end for
35: end procedure
```

---

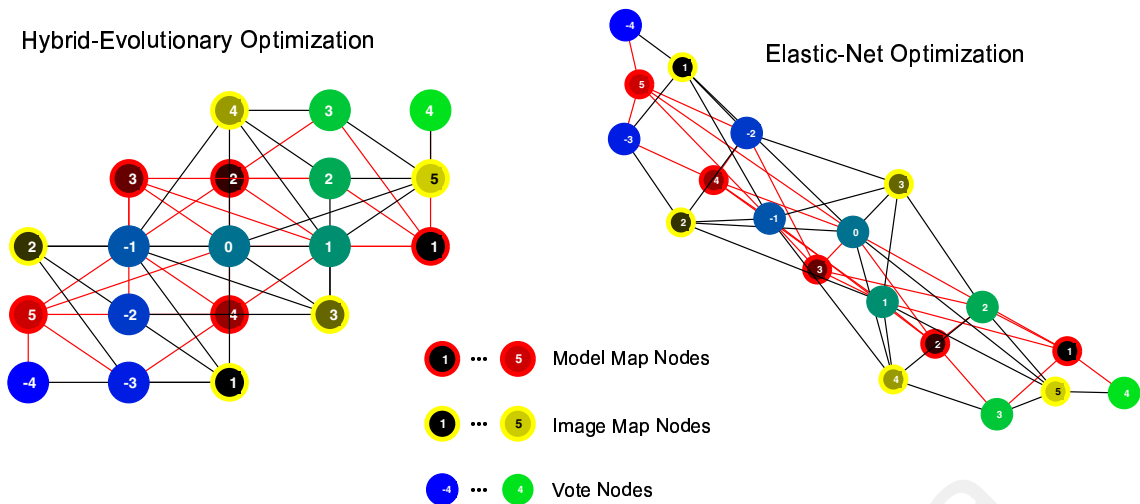


Figure 7.4: The results of two structural optimization methods.

1. Vote, model and image nodes are organized topographically.<sup>7</sup>
2. The zero-shift vote node is at the center of the architecture.
3. Vote nodes are sandwiched in between map nodes.
4. Model and image nodes wind around the central vote nodes in a helicoidal fashion.
5. Model and image nodes run in opposite directions.

Because the elastic network approach implicitly implements TWL minimization, and finds architectures with the same design principles as the Hybrid Evolutionary algorithm (see Fig. 7.4), and considering that it is simpler and thus faster to run, we chose to apply it to the remaining cases. Refer to Fig. 7.5 for a sequence of steps in the elastic optimization of a one-dimensional shift estimator.

## 7.2.4 Examples

### $(T_x, T_y)$ with a Single Feature Type

This case takes the problem depicted in Fig. 7.4 one step further by including another dimension into the estimation, i.e: estimate  $T_x$  and  $T_y$ . Figure 7.6 depicts the optimal architectures that were found.

<sup>7</sup>Note that this topography emerges spontaneously from the pattern of connections between map and vote nodes, seeing that the architectures are randomly initialized before the optimization processes.

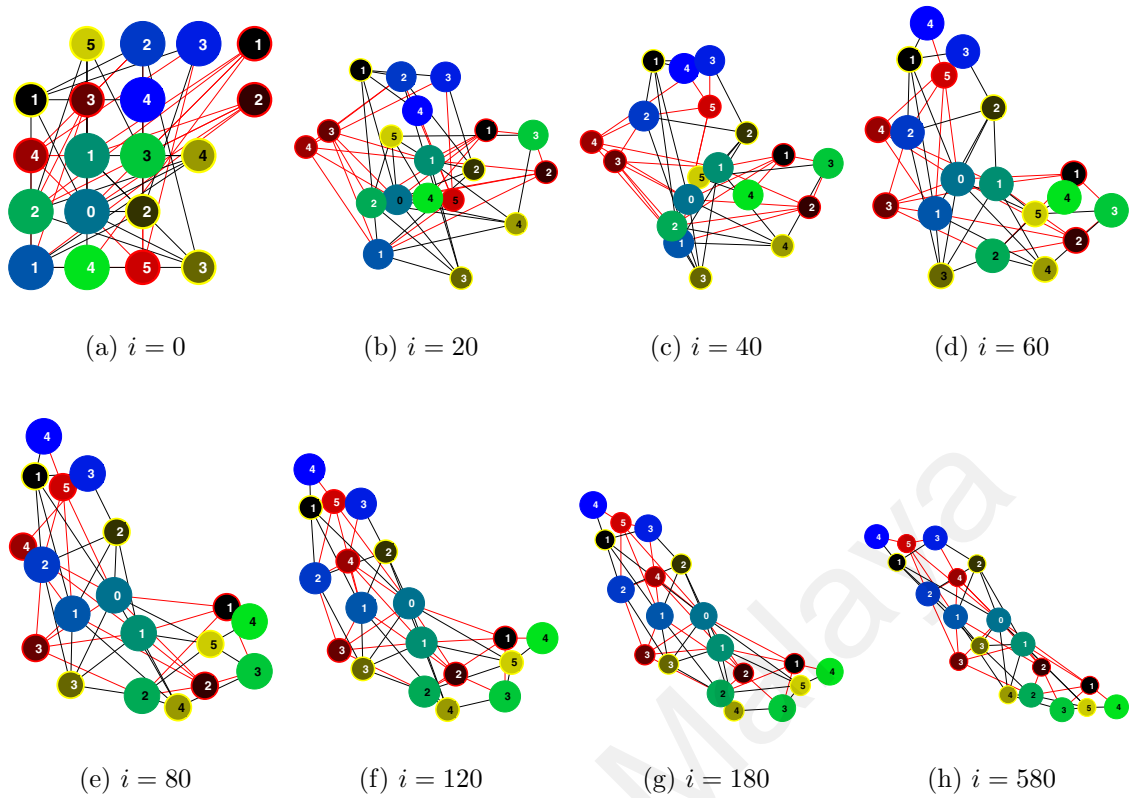


Figure 7.5: A sequence of elastic network steps.

The top-left architecture depicts an optimized architecture for  $2D$  space, while the top-right architecture depicts an optimized architecture for  $3D$  space. The bottom-left architecture is the result of a side-view of the same architecture which is viewed from above in the top-right illustration. Maps consist of  $3 \times 3 = 9$  arrays of nodes. The nodes are numbered from 1 to 9, counting from left to right and bottom to top, e.g. node 2 corresponds to  $(x = 2, y = 1)$  while node 7 corresponds to  $(x = 1, y = 3)$ . Model nodes are identified by their red boundaries while image nodes are identified by their yellow boundaries. Vote nodes, on the other hand, are drawn with black boundaries. Vote nodes represent all possibilities comprehended between  $(T_x = -2, T_y = -2)$  and  $(T_x = +2, T_y = +2)$  inclusive. Vote nodes are numbered from 1 to 25, counted, like map nodes, from left to right and bottom to top, e.g: vote node 2 represents the vote  $(T_x = -1, T_y = -2)$  while vote node 24 represents the vote  $(T_x = +1, T_y = +2)$ .

Several design principles emerged for the optimization case in  $2D$  space (top-left architecture of Fig. 7.6):

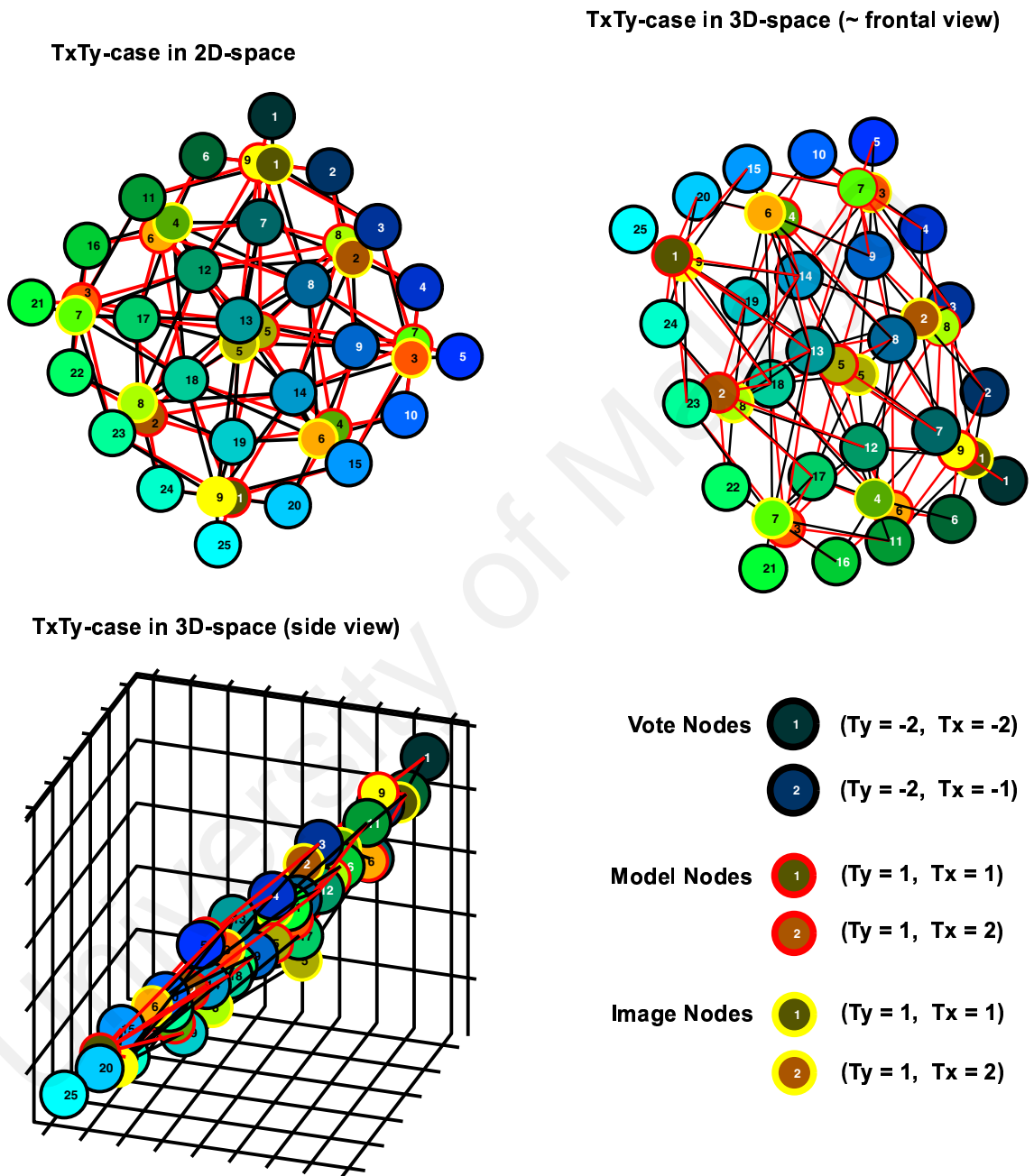


Figure 7.6: The  $(T_x, T_y)$  case with a single feature type.

1. Vote, model and image nodes are organized topographically in two dimensions.
2. Image nodes run in the same direction as vote nodes.
3. Model nodes run in the opposite direction of vote and image nodes.
4. Model and image nodes representing opposite positions, are clustered in pairs.
5. Model/image pairs are evenly distributed across the  $2D$  topography of vote nodes.
6. Vote map origins are centered in the optimized architectures.

When the same type of architecture is optimized in  $3D$  space (top-right and bottom-left architectures of Fig. 7.6), the same design principles emerge, with an additional subtle detail: there is some meandering of map nodes along the vote-node plane, which is somewhat similar to the helicoidal meandering of the simpler “one-dimensional shift estimation” problem depicted in Fig. 7.4.

### $(T_x, T_y)$ with Multiple Feature Types

This architecture is essentially equivalent to the previous one except for the addition of multiple feature types for each map position. Figure 7.7 depicts on the top-left an optimization of the architecture in  $2D$  space, on the bottom-left a top-view of an optimization in  $3D$  space and on the bottom-right a side-view of the same  $3D$  architecture.

The numbering conventions of the map and vote nodes are exactly the same as those employed in the previous case. The main difference pertains to the boundary colors of different feature-types. Feature-types are distinguished by the brightness of their boundaries.

The design principles that emerged for the single feature-type case can also be found in this case, with the additional design principles that feature-types for the same position (and same type of map: model or image) cluster together and that, for the  $3D$  case, map nodes make more use of vertical space (relative to the vote node plane) as can be seen in the side-view on the bottom-right of Fig. 7.7. This

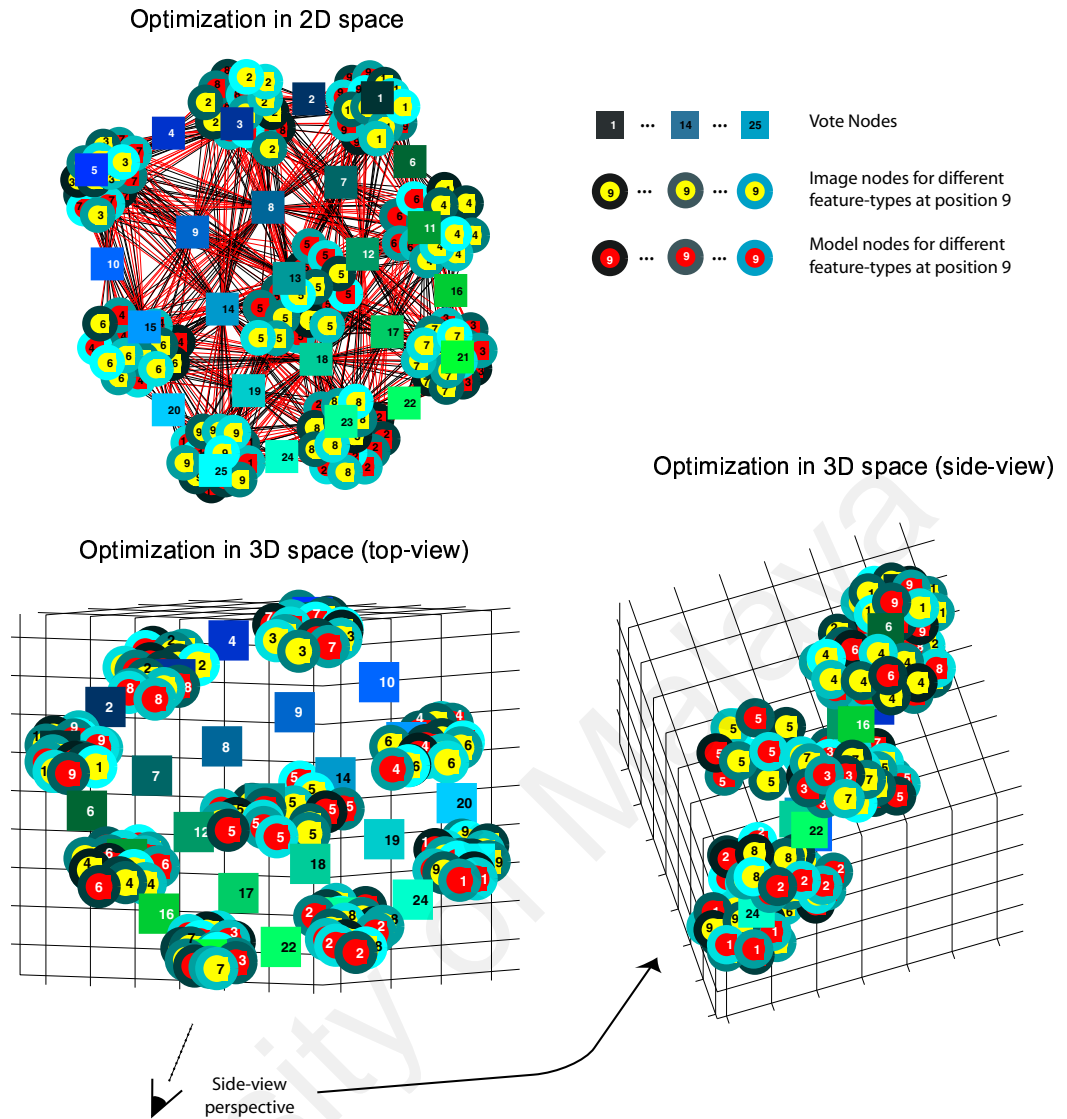


Figure 7.7: The  $(T_x, T_y)$  case with ten distinct feature types.

additional use of vertical space exhibits interesting symmetries: notice how different feature-types for the same position (and type of map) can be found (symmetrically) on both sides of the vote node plane.

### Scaling and Rotation with a Single Feature Type

The current problem illustrates very clearly what is meant by the functionality (e.g. type of transformation being estimated) of the network affecting the appearance of the optimal architecture. The transformation being estimated determines the pattern of connections between map and vote nodes and thus affects what architecture minimizes TWL. The left-hand side of Fig. 7.8 depicts an optimized scale-estimation architecture, while the right-hand side illustrates an optimized rotation-

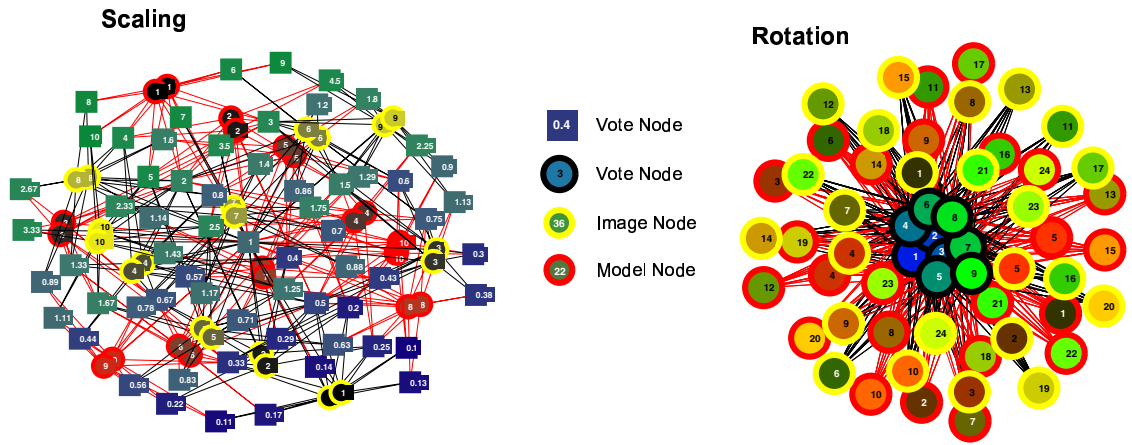


Figure 7.8: Elastic optimization for both the scale and rotation cases.

estimation architecture.

The boundary-color conventions for map-nodes are equivalent to the previous cases. However, the maps used in the scale estimation architecture are one dimensional and so the numbering representation is somewhat different. Maps are assumed to have an origin and to extend in two opposite directions for 10 nodes. Both directions are functionally equivalent and are thus not expected to have a great impact on the resulting architecture. Likewise, the numbering scheme for the rotational architecture is somewhat different: there are 24 map nodes representing positions  $(x = -2, y = -2)$  to  $(x = +2, y = +2)$  (from left to right and bottom to top), ignoring the origin (i.e.  $(x = 0, y = 0)$ ). In the scale-estimation architecture vote-nodes are represented by squares whose colors vary from blue (down-scaling) to green (up-scaling). In the rotation-estimation architecture vote-nodes are represented by blue/green circles with black boundaries which vary from  $-\pi$  radians (vote number 1) to  $+\pi$  radians (vote number 9) in  $\pi/4$  radian increments.

For the scale-estimation architecture the following design principles are apparent:

1. Global topography of vote nodes: general transition from blue (scale-down) nodes on one side to green nodes on the other (scale-up). Notice however, that there are minor local exceptions/variations.
2. Symmetry of scaling-up and scaling-down nodes, e.g: the second largest scaling-up node (i.e. 9) at the top right corner of the architecture is symmetrical to the second smallest scaling-down node (i.e. 0.11) at the bottom left



corner.

3. Global topography of map nodes with some local exceptions/distortions.
4. Model and image nodes run in opposite directions and meander symmetrically.
5. Each model/image node is paired with its positional equivalent (at the opposite map side).
6. Vote and map nodes are evenly interleaved.

The design principles that emerge from optimizing the rotation-estimation architecture are fundamentally different from the previous case, again highlighting the fact that functionality and thus the pattern of connections between map and vote nodes are critical to the resulting optimal architectures. The following design principles are apparent:

1. Vote nodes are clustered in the center. This is the opposite of the even interleaving of the scaled-estimation architecture.
2. Vote nodes are somewhat topographically organized: the left half (see Fig. 7.8) of the vote-node cluster contains nodes representing angles smaller than  $\pi$  radians, while the right half contains nodes representing angles larger than  $\pi$  radians.
3. Map nodes are spherically and evenly distributed around vote nodes.

### $(T_x, T_y, \theta)$ with Multiple Feature Types

The last case illustrates how some of the design principles that emerge in simpler architectures are general and powerful enough to apply to more complex cases (including unconstrained estimation). This case pertains to the estimation of rigid transformations (translation and rotation) using single correspondences. See Fig. 7.9 for an illustration of the optimal three-dimensional configuration for a particular instance of the problem: the left-hand side depicts a top-view of the architecture while the right-hand side illustrates a side-view.

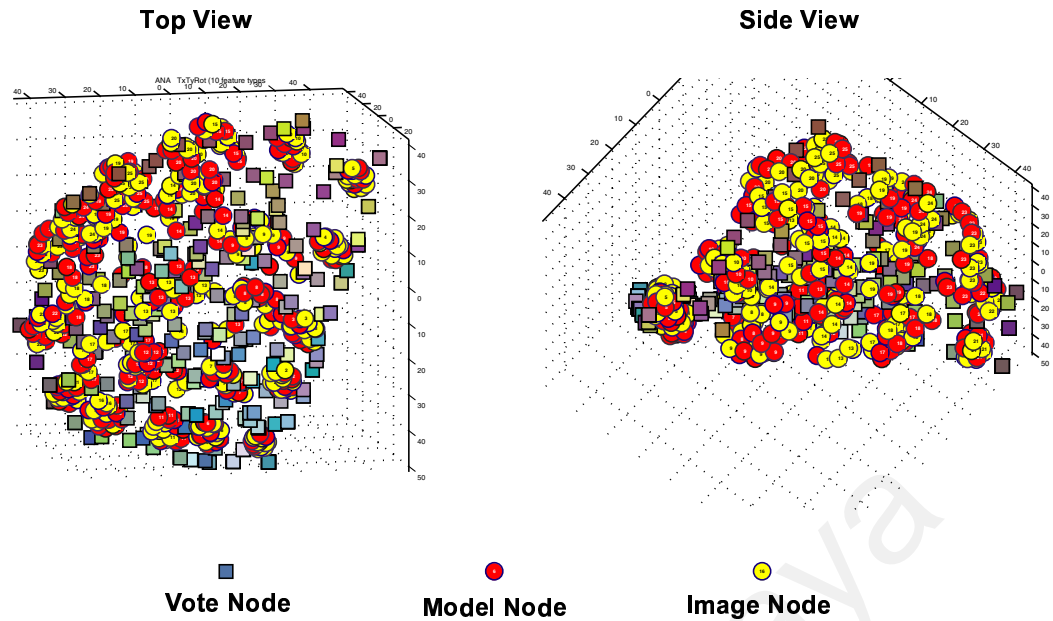


Figure 7.9: The  $(T_x, T_y, \theta)$  case with ten feature types, optimized in 3D space.

As before, model nodes are red, image nodes are yellow, feature types are represented by edges with varying brightness levels, and the numbers on map nodes represent their  $2D$  positions. More importantly vote nodes, which now represent a complex three-dimensional structure  $(T_x, T_y, \theta)$ , are represented by squares whose color reflects their position in the  $(T_x, T_y, \theta)$  voting structure: a larger degree of red in the color composition represents a larger  $\theta$ , while more green represents a larger  $T_x$  and more blue corresponds to a larger  $T_y$ .

Although the complexity of the architecture makes it harder for the identification of design principles, some crucial ones are still easily observed, e.g:

1. Both vote and map nodes have spontaneously emerged topographical organizations.
2. Although one might expect vote nodes to be organized in a cuboid structure, in fact, they seem to be forming a moderately flat pyramidal structure.
3. Model and image nodes representing the same position cluster together, as do the various feature-types for the same position.
4. Interesting uses of 3D space seem to be taking place: some helicoidal meandering and other structures.

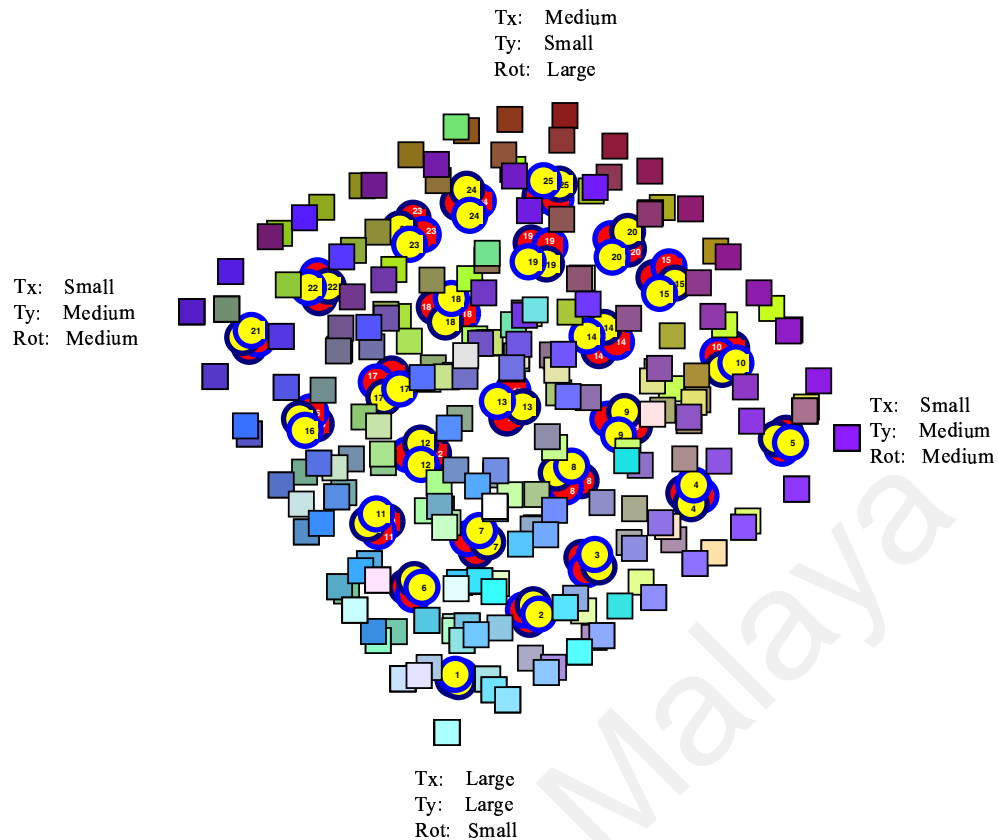


Figure 7.10: The  $(T_x, T_y, \theta)$  case with two feature types, optimized in 2D space.

In order to visualize some of the principles more clearly, the same  $(T_x, T_y, \theta)$  architecture (except for the number of feature types) was optimized in two-dimensional space: see Fig. 7.10. Notice how extreme map positions (e.g. 1 and 25) coincide with extreme vote positions (e.g. large  $\theta$ ). Notice also how  $T_y$  and  $\theta$  increase in opposite directions:  $T_y$  increases from top to bottom while  $\theta$  increases from bottom to top. The configuration of  $T_x$  positions is somewhat more elusive: there seems to be a vague Gaussian variation from left to right. This is likely to be a consequence of the fact that the optimization process is aligning a 3D map to a 2D map.

### 7.2.5 Conclusions

Probably one of the greatest lessons from the above optimizations, is that optimal architectures (at least relative to TWL) can be counter-intuitive and unexpected. Several of these unexpected principles keep recurring in different architectures, e.g: model and image nodes meandering in opposite directions. The fact that many of the identified principles keep recurring, is an indication that they are quite general,

and most probably apply to more complex/realistic cases. They provide us with suggestions for hardware implementations and hypotheses for biological<sup>8</sup> investigations. The following list condenses the main lessons provided by the above optimization experiments:

1. Topographical organizations (whether of map or vote nodes) are not only natural (a reflection of earlier inputs), they are also efficient in terms of TWL, i.e: they emerge (from initial random configurations) as a consequence of the patterns of connections between map and vote nodes.
2. Vote nodes tend to be sandwiched in between model and image nodes.
3. An even distribution of vote and map nodes seems to be the favored configuration. Although rotation-estimation prefers vote-nodes in the center (see the right-hand side of Fig. 7.8) when it is combined with translation estimation (i.e. the rigid transformation case), the favored architecture reverts more to an even distribution (see Fig. 7.9).
4. Model and image nodes tend to run in opposite directions.
5. Model and image nodes tend to meander somewhat symmetrically to each other (see the triple helix in Fig. 7.4).
6. Feature types for the same position (and type of map) tend to cluster in the same region. The clustering seems to favor a spherical rather than a columnar (or other) structure.

In the first half of this chapter we investigated various combinatorial issues regarding neural architectures that embody correspondence distributions for pose estimation, e.g: number of correspondence detectors. In the second half we structurally optimized the architectures according to the total wiring minimization goal and

---

<sup>8</sup>Regarding biological hypothesizing, although TWL minimization seems to be a strong force determining neural architectures, one must not forget that other important constraints/goals might also be playing a part. Developmental constraints are an important example. Even if there is an obvious architecture that greatly reduces TWL, a developmental mechanism that can put neurons in that configuration might not exist.

extracted several general design principles. Referring to the problem framework delineated in Fig. 2.1 one can say that the current chapter belongs to the implementational level of abstraction. In the following chapter we will attempt to form bridges between the architectures in the current and previous chapters on one side, and biological findings on the other.

University of Malaya

# Chapter 8

## Biological Neural Architectures

This chapter is concerned with further advancing the argument in favor of the biological plausibility of the artificial neural architectures discussed in the previous chapter. First, certain elements of biological nervous systems which provide evidence or implementational support for the ANAs will be presented. Following this, important biological constraints which affect the plausibility of different biological neural architectures (BNAs) will be discussed. Finally, the information gathered from the “support” and “constraints” sections will be combined in order to hypothesize biologically plausible neural architectures for pose estimation.

### 8.1 Biological Support

#### 8.1.1 Topographic Maps

Topographic maps (e.g. Gierer & Muller 1995) in a neural context define architectures where neighboring nodes represent neighboring regions of some aspect of the external world. Examples of aspects range from muscular (relevant to motor representations) to visual, auditory or other types of stimuli. Within visual stimuli, sub-aspects might for example include: feature position, colour, edge orientation, motion direction, depth and others. Using feature position as an illustration, a topographic map in this context implies that neighboring nodes in the map represent neighboring feature positions as represented in the retina. The nomenclature for

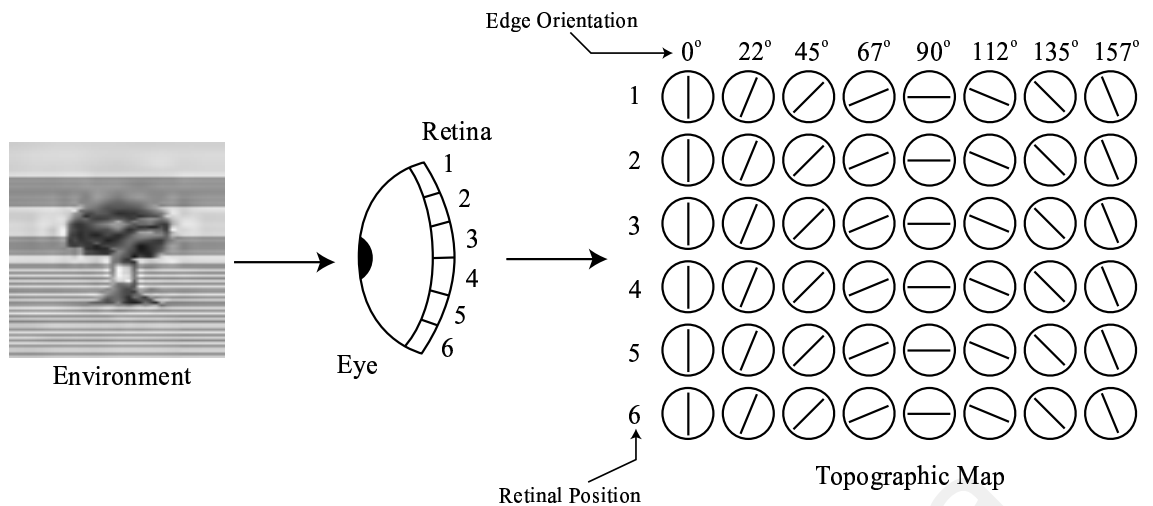


Figure 8.1: The preservation of position and orientation relationships.

topographic maps is further refined according to what neighborhood relationships are being preserved. If a map preserves retinal position relationships, then it is referred to as a retinotopic map. Other examples are: tonotopic maps (preserve sound frequency relationships), somatotopic maps (preserve positional relationships of the body surface) and others (see Gazzaniga et al. 2002). Refer to Fig. 8.1 for an illustration of a neural map that preserves positional and, more abstractly, orientational relationships.

This subsection consists essentially of two arguments, one weak and the other strong. The weak argument states that since the structurally optimized architectures of Chapter 7 consistently exhibit topographic organizations, then if biological neural systems also exhibit these organizations then one should be more confident that the pose estimation architectures are biologically plausible. The strong argument states that if the *types* of topographic organizations found in the structurally optimized architectures of Chapter 7 are also found in biological neural systems, then one should be more confident that not only they are biologically plausible, but that they actually exist in biological systems. The weak argument argues for *they can exist*<sup>1</sup> while the strong argument argues for *they do exist*.

Considering the weak argument first. It turns out that topographic maps are ubiquitous in biological systems (see Allman 1999). From reptiles, to birds, to

<sup>1</sup>Even if they are not found in any known biological system, they *can* exist in some hypothetical or unknown biological system.

mammals, and others, topographic maps are repeatedly found, whether in visual, auditory, olfactory, somatosensory, motor or other areas. One particular retinotopic map that is so ancient that it is shared by all vertebrates lies in the optic tectum. In fish, amphibians and reptiles the optic tectum is the main visual processing center and thus is involved in a broad range of functions (including the integration of visual, auditory and somatosensory information represented in the tectum), while in primates, it is involved in the specialized process of shifting gaze towards interesting stimuli (see Allman 1999). Some topographic maps are evolutionarily more recent, such as those found in the neocortex of mammals. The neocortex of primates, for example, is replete with topographic maps. The visual cortex, for example, consists of many distinct areas, which are anatomically, chemically, physiologically, or otherwise, defined, e.g: V1, V2, MT and others (see Zeki 1993)<sup>2</sup>. Each one of these areas consists of multiple topographic maps. Area V1 for example can be seen to contain topographic maps for color, edge orientation, spatial frequency, disparity and motion direction. Thus, evidence seems to abound that satisfies at least the weak argument.

Now for the strong argument. Interestingly, there are various aspects of the structurally optimized architectures in Chapter 7 which do seem to appear in biological systems:

1. Mirror symmetric topographic maps. Refer back to Figures 7.4, 7.6, 7.7 and 7.8 (in Chapter 7), where structural optimization led to, among other things, interesting mirror symmetries between model, image and vote nodes. Similar types of symmetry have been found in the sensory neocortexes of many different species (refer to Schulz & Reggia 2005)<sup>3</sup>.

---

<sup>2</sup>V1, also known as the primary visual cortex, or the striate cortex, is the first cortical area processing visual information. Neurons in V1 have been found that are tuned to edge orientation, spatial frequency, depth, colour and motion direction (refer to Olshausen & Field 2005 for a very interesting review of the current state of knowledge regarding V1 and what remains to be discovered). Area V2 is V1's major projection area. The tuning properties of V2 neurons are similar to those of V1 (refer to Levitt et al. 1994 for an overview of the functional properties of V2 neurons in the macaque) except for a greater degree of invariance (e.g. larger receptive fields), complexity and abstraction (e.g. illusory contours). Area MT, or V5, is well known for its neurons specialized for motion analysis (e.g. Mikami et al. 1986).

<sup>3</sup>One caveat to this piece of evidence however is that the mirror symmetric maps discussed in (Schulz & Reggia, 2005) are adjacent to each other, rather than overlapping, as what emerged from our structural optimization experiments



2. Variables tangential to the cortical surface. One interesting feature of the structurally optimized architectures is that they do not explore three-dimensional space as much as one might expect, i.e: variables (e.g. model positions, image positions and vote indeces) tend to spread out two-dimensionally. The same seems to occur in the neocortex, where the tuning properties of neurons tend to vary tangentially to the cortical surface whilst remaining significantly invariable in the dimension perpendicular to the surface (see Mountcastle 1997). In fact, the neocortex seems to have an abundance of vertical structures where neurons are characterized by their tuning similarities, i.e: minicolumns, columns and macrocolumns (see Buxhoeveden & Casanova 2002).
3. Complex feature overlapping. Seeing that the neocortex seems to represent variables tangentially to the cortical surface (only two dimensions are available), how can three or more variables be represented? To use an example (although the whole picture has not emerged yet), it is already clear that area V1 exhibits interesting patterns of semi-overlapping orientation<sup>4</sup>, ocular dominance, spatial frequency and disparity features (see Mountcastle 1997). Note that the phenomenon of feature<sup>5</sup> overlapping emerged from the structural optimization experiments of the previous chapter.
4. Clusters of feature types at the same position. Structurally optimizing architectures where each map position is represented by multiple feature types reveals a design principle stating that multiple features cluster around their corresponding positions, rather than disperse. This principle is reflected in cortical maps, where for example, there is evidence for negative correlations between a retinal position gradient and the gradients of other feature representations such as orientation preference (see Yu et al. 2005).
5. Inter-map connections. It is clear that the neural pose estimation approach advocated in this thesis depends on the relationships/connections between two maps. Inter-map connections are also found in the neocortex, e.g: reciprocal

---

<sup>4</sup>Orientation variations have been shown to be linear and non-linear. Examples of non-linear variations are: singularities, fractures and saddles.

<sup>5</sup>Model feature positions, image feature positions and transformation votes.

connections between V1 and V2 exist that even seem to play a part in cortical folding (see Allman 1999).

Although the case for the strong argument is not as heavy as that for the weak argument, it is clear that there is considerable support and that it should be worthwhile to investigate other aspects of biological neural systems.

### 8.1.2 Features

Local features are indispensable elements of SCAPE (see Chapter 4). Without them, there can be no correspondences, and thus no information on which to base pose estimates. Therefore, it is crucial that biological neural systems show evidence of local features, for without this evidence, the case of SCAPE's biological plausibility is quite weak. It turns out that the visual cortices of many vertebrates are rich with local features. The fact that individual neurons respond to particular aspects of different stimuli (e.g. edge orientation and patch colour) attests to their *featuralness* (and discrimination power), while the fact that they are responsive to stimuli within a limited (spatially constrained) receptive field attests to their *locality*. Furthermore, and to great advantage, several known local features in biological systems exhibit invariance to affine transformations (e.g. colour, corners and junctions). Although other feature-types exist in biological systems, we will here briefly describe edge orientation, colour, corners/junctions and depth.

#### Orientation

Single neurons tuned<sup>6</sup> to different edge-orientations were originally discovered by Hubel and Wiesel in 1958 (refer for example to Hubel & Wiesel 1968). Since then, the literature on edge-tuned neurons, their properties, the neural processes underlying them, their functions, their interactions with other features and their topographic organizations, has greatly expanded (refer, for example, to the introductory review in Troyer et al. 1998), and so, one can conclude that there is no lack of evidence

---

<sup>6</sup>A neuron "tuned" to aspect  $X$  of the environment means that, in most cases, when  $X$  is placed in the neuron's receptive field, the neuron fires with a significantly higher rate than when all other types of stimuli are presented.

for this type of local feature. An interesting property of this type of cell, which provides some geometric invariance, shows that halving a line within a relevant neuron's receptive field halves the intensity of the neuron's response (see Zeki 1993).

### **Colour**

Many neurons in the macaque visual cortex (e.g. area V1) have small receptive fields which respond to light of a particular wavelength (see Zeki 1993). Furthermore, in area V4, neurons have been found which are sensitive to the actual colour of objects, discounting the effect of ambient illumination and other variations and thus demonstrating the property of colour constancy (see Rolls & Deco 2001). Note that colour is invariant to geometric transformations such as rotation and scaling.

### **Corners and Junctions**

Corners and junctions are local features with a significant amount of robustness to affine transformations, and discrimination power. Regarding the latter, Biederman in (Biederman, 1985) has shown how human object recognition is impaired by removing corners with high curvature, but not so by removing corners with low curvature. Cells in the monkey visual cortex have been found which are sensitive to corners and ends of lines (see for example Hubel & Wiesel 1968). Area V4 has recently been shown to contain neurons that are tuned to curves and angles (see Pasupathy & Connor 2001). Refer to (Hansen & Neumann, 2004) for an interesting proposal for the neural mechanisms that might be underlying the neural detection of junctions.

### **Depth**

Depth might also be considered a useful local feature. Since as far back as 1968, cells tuned to binocular disparity have been known to exist (see Barlow et al. 1967). Neurons in other areas, which are also tuned to binocular disparity have been found in V3, MT and MST (see Rolls & Deco 2001).

### 8.1.3 Neuronal Variety

The argument underlying this subsection is quite simple and conservative but not without its uses. As Chapter 6 showed, neural implementations of SCAPE require several neural components. Future extensions (or hypotheses) might even require other unforeseen components. Therefore it is essential that biological systems exhibit a rich variety of neural components if they are to be capable of accommodating SCAPE. A variety of neural components in this context, implies a variety of cortical<sup>7</sup> neurons, since different types of neurons have different computational properties.

It turns out that there is an incredible variety of cortical neuron types. So much so that there is still no definitive taxonomy of cortical neurons (see Hevner et al. 2003), in contrast to retinal or spinal-cord neurons. But there are various dimensions (not necessarily independent) along which cortical neurons can be classified.

One obvious dimension along which one can classify cortical neurons refers to whether they are excitatory or inhibitory. When an action potential arrives at an axonal bouton of an excitatory neuron, this increases the probability of the post-synaptic neuron firing. Conversely, when an action potential arrives at an axonal bouton of an inhibitory neuron, this decreases the probability of the post-synaptic neuron firing. Although there are exceptions<sup>8</sup>, most neurons in biological systems are either excitatory or inhibitory. The existence of excitatory neurons is clearly relevant to computations behind vote accumulation and inhibitory neurons are important for some types vote sharpening networks.

Another dimension based on the projection properties of neurons creates the following categorization: 1) projection neurons and 2) interneurons. Projection neurons have long axons to other cortical or subcortical areas and tend to be spiny and excitatory, whereas interneurons have short axons that synapse with nearby neurons in local circuits and tend to be smooth and inhibitory. Interestingly, there is a significantly larger morphological variety of inhibitory interneurons as compared to excitatory neurons in general.

---

<sup>7</sup>We are assuming here that the neocortex is one of the most plausible locations for SCAPE.

<sup>8</sup>During early development, for example, some terminals in auditory synapses can release both excitatory and inhibitory neurotransmitters simultaneously (Guosong Liu, personal communication, November 5, 2004).

Another dimension along which cortical neurons can be classified is based on the presence/absence of spines on dendrites. Spines are small processes that protrude from dendrites with a thin neck and somewhat bulbous head and upon which at least one synapse is formed (e.g. see Coss & Perkel 1985, Yuste et al. 1999 and Kandel et al. 2000). This property allows neurons to be classified into two main categories: 1) those whose dendrites exhibit spines (spiny neurons) and 2) those whose dendrites are devoid of spines (smooth neurons). A much smaller sub-class of cortical neurons has also been identified which might be referred to as “sparsely spiny” (see Shepherd 2003). Examples of spiny neurons are pyramidal and spiny stellate cells and examples of smooth neurons are basket and chandelier cells. Examples within each category (spiny or smooth) can be further distinguished according to other morphological properties. Basket cells, for example, get their name from the fact that multiple basket-cell axons form basket-like structures around the soma of target cells. On the other hand, chandelier cells, get their name from the fact that their axonal boutons form vertical candle-like structures which on a whole, make the axonic arborization resemble a chandelier.

The morphological variations mentioned above, only begin to scratch the surface of what biological systems actual exhibit. This flexibility is likely to be very useful for SCAPE, when one considers the various (sometimes counter-intuitive) architectures that are suggested by structural optimization (refer to Chapter 7 for examples).

Neurons can also be classified according to their electrophysiological properties. Different types of neurons can be distinguished from each other based on their firing patterns. Some excitatory neurons, for example, exhibit regular spiking while others exhibit mainly intrinsic bursting and others mostly manifest a chattering pattern (see Contreras 2004 for more details).

As one can conclude from this very brief foray into neuronal variety, the number of types that can be defined along neurochemical, morphological, electrophysiological and other dimensions, is very significant, and thus one should be more confident that the neural components required by SCAPE do indeed exist in biological systems.

### 8.1.4 Dendritic Conjunctions

SCAPE is an approach to pose estimation that depends on the analysis of feature correspondences, therefore correspondence detectors are indispensable to any architecture that aims to implement SCAPE. As Chapter 6 showed, neural correspondence detectors essentially compute a logical-AND between two inputs. Neurons have significantly more synapses than what is required by a simple conjunction of two inputs, therefore it would be extremely wasteful and unrealistic to use neurons as correspondence detectors. Do biological systems have smaller/cheaper neural components that can do the job? Fortunately, as hinted by previous modeling research (see Poirazi et al. 2003), and as indicated by recent empirical research (see Polsky et al. 2004), smaller neural components for computing the logical-ANDs of correspondence detectors do indeed exist.

In (Polsky et al., 2004), the authors showed that dendritic branches are capable of complex non-linear computations, which can implement a logical-AND between two inputs. More specifically, the experiments were conducted on layer-5 pyramidal cells in the rat cortex and demonstrated non-linear integration of inputs in basal and apical-oblique dendritic branches. If two excitatory post-synaptic potentials (EPSPs) were applied to the same dendritic branch, close enough to each other (spatially and temporally), and if their sum exceeded some threshold, then the output was superlinear (i.e. stronger than the linear sum). In the same context, if the sum of EPSPs was below the threshold then the output was linear. Furthermore, if the sum greatly exceeded the threshold the output was sublinear (i.e. weaker than the linear sum). This combination of linear, superlinear and sublinear integrations, essentially draws out the familiar sigmoid function, which to our advantage can be used to compute a logical-AND. It should be added, that the experiments reported in (Polsky et al., 2004) also found that if, instead of placing the EPSPs on the same dendritic branch, they were placed on separate branches, the resulting integration was mostly linear. In conclusion, dendritic branches, function as somewhat independent computational units with sigmoid activations functions.

This is extremely advantageous from our point of view because it means that

correspondence detectors can be implemented at the level of dendritic branches, thus saving a great deal of neuronal resources. Placing correspondence detectors at the level of dendritic branches also makes questions regarding the unsupervised learning of SCAPE's connection patterns more approachable since it opens the door to considerations regarding axonal and dendritic developmental/learning dynamics. Furthermore, the fact that dendritic branches can compute conjunctions, and that the results of these conjunctions can still be summed/integrated by the same neuron, means that SCAPE's two main computations can be implemented by a single neuron, i.e: correspondence detection and vote summation (see Fig. 8.4).

In the following subsections, we will make use of some of the insights gained from SCAPE's biological support and some of the design principles that emerged from our structural optimization experiments to propose biologically plausible architectures for SCAPE, both for the constrained (C-SCAPE) and unconstrained cases (U-SCAPE). The question of where exactly biological SCAPE implementations might be found (e.g. in what cortical layer(s) and/or in what visual area(s)) is not addressed, since it is felt that the general architecture (essentially a non-linear transformation/integration of two maps into a third map using conjunctions) is simple and general enough to be implementable in different regions (e.g. posterior parietal cortex, superior colliculus, cerebellum, etc).

### 8.1.5 Example Constrained Architecture

Because of the amount of wiring that SCAPE architectures require, and the visualization problems that this entails, we have chosen to illustrate a  $(T_x, T_y)$  estimator using maps that contain  $2 \times 2 = 4$  nodes (see Fig. 8.2).

Figure 8.2 consists of model nodes (in red), image nodes (in yellow), vote nodes (in blue) and the connections between them: axons from model and image nodes synapse onto the dendrites of vote nodes. Each map neuron represents a different map-coordinate and each vote node represents a different translation-estimate.

Several design principles that minimize the total-wiring-length (TWL) are evident in Fig. 8.2: 1) map and vote nodes are laid out in a topographical organization,

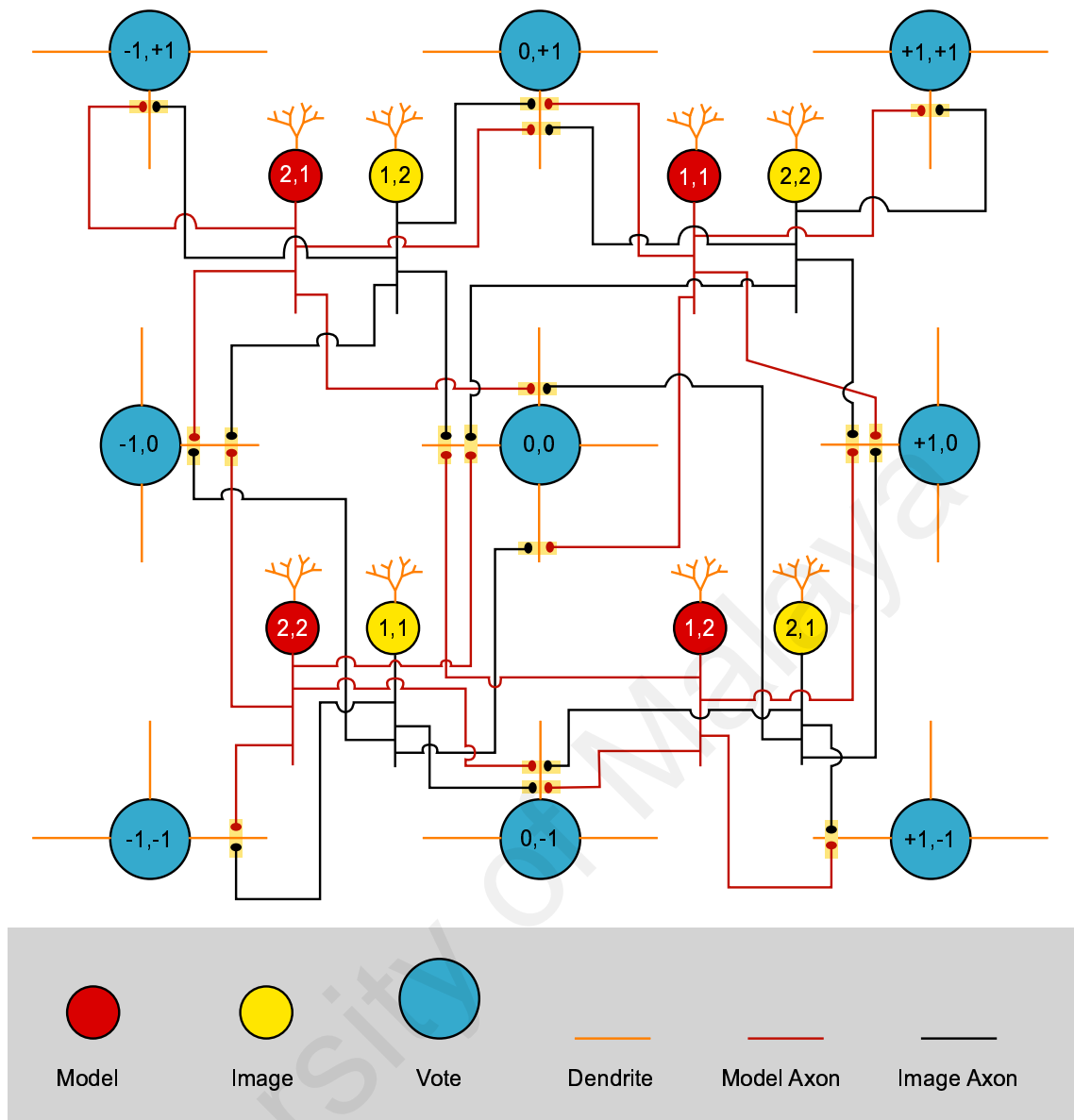


Figure 8.2: Biological neural architecture for a simple  $(T_x, T_y)$  estimator.

2) map and vote nodes are evenly spaced out in an interleaving fashion, 3) image nodes run in parallel to vote nodes and 4) model nodes run in the opposite direction to image and vote nodes. Refer to Fig. 7.6 in Chapter 7 for the relevant structural optimization experiment.

The following biological facts were incorporated into the diagram in Fig. 8.2: 1) all neurons are excitatory, 2) vote node dendrites have a stellate structure (e.g. spiny stellated cells) and 3) correspondence detectors are implemented as conjunctions at the dendrites of vote cells (the relevant inputs must be close enough in order to implement the logical-AND computational unit).



### 8.1.6 Example Unconstrained Architecture

The unconstrained case creates an additional problem regarding the implementation of voting-manifolds. How should voting-manifolds be implemented? One possibility, of course, is through the use of extra dendritic-conjunctions, but as Chapter 7 demonstrated, a U-SCAPE solution based on correspondence detectors with single-outputs is usually too expensive. Another solution is by incorporating intermediary nodes as correspondence detectors and which can project particular manifolds through their axonic arborizations. Ideally, we would like to find neurons with a single dendrite and a large number of axonal boutons<sup>9</sup>. Do such neurons exist in biological systems? Biological nervous systems exhibit such tremendous morphological (neuronal, dendritic and axonal), electrochemical and dynamic variety, that one might imagine that almost any type of neuron that one can conceive of is plausible. However, plausibility does not prove existence, therefore we have to look at what real nervous systems exhibit. Several neuron-types seem to apply as reasonable candidates for implementing correspondence-detection and vote-manifold projection: 1) LGN single-input cells (see Mastronarde 1987), 2) cortical bipolar neurons, 3) unipolar brush neurons (see Kalinichenko & Okhotin 2005) and 4) amacrine related cells.

Figure 8.3 contains a simple diagram of a hypothetical biological U-SCAPE architecture. The axons of model and image nodes synapse (as conjunctions which represent particular correspondences) onto manifold-neurons, each one of which sends its axons to multiple vote-nodes. One problem with this solution is that the burden of representing correspondences falls on individual neurons, rather than particular locations of some dendritic tree (as for the C-SAPE case). If each map consists of  $10^4$  nodes, then  $10^8$  manifold-neurons are required, which for example approximately encompasses the whole of human area V1. One solution is to use smaller maps with more abstract features: maps with  $10^2$  positions lead to the requirement of  $10^4$  manifold-neurons which is more acceptable (i.e. 0.01% of V1). Other solu-

---

<sup>9</sup>Although adendritic neurons are also good candidates, if one assumes that a dendritic conjunction can be implemented on the cell body.

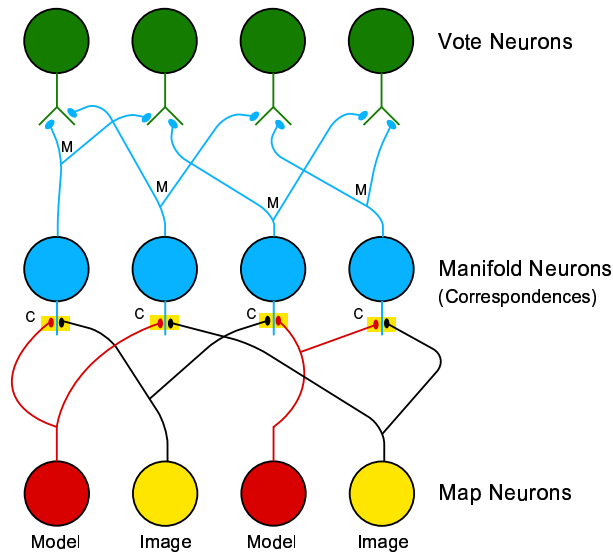


Figure 8.3: A SCAPE BNA for a simplified unconstrained estimation case.

Table 8.1: Various human cortical constraints.

Constraint	Value	Reference
Num. Cortical Neurons	$\sim 2.1 \times 10^{10}$	(De Haan & Johnson, 2003)
Num. V1 Neurons	$\sim 10^8$	(Wandell, 1995)
Num. Synapses per Neuron	$\sim 10^4$	(Johansson & Lansner, 2004)
Syn. per L2/3 Pyramidal Neuron	$\sim 6 \cdot 10^3$	(Binzegger et al., 2004)
Syn. per Purkinje Neuron	$> 2 \times 10^5$	(Gazzaniga et al., 2002)
Bout. per L2/3 Pyramidal Neuron	$\sim 6 \cdot 10^3$	(Binzegger et al., 2004)
Neuronal Density	$\sim (5 \times 10^4)/mm^3$	(Shepherd, 2003)
Dendritic/Axonal Density	3/5 of Total Volume	(Chklovskii et al., 2002)
Synaptic Density	$\sim (1.6 \times 10^9)/mm^3$	(DeFelipe et al., 1999)

tions involve clustering similar manifolds into the same manifold-neuron<sup>10</sup>, which however, has the disadvantage of some loss of accuracy.

## 8.2 Biological Constraints

Although it is important to consider temporal constraints (e.g. firing patterns, coding strategies and reaction times), we will here focus primarily on morphological and spatial constraints. Table 8.1 lists down a subset of relevant constraints for biologically plausible models of SCAPE. Note that the “Syn.” abbreviation stands for “number of synapses” while the “Bout.” abbreviation stands for “number of axonal boutons”.

<sup>10</sup>The manifold-neuron, in this case, should contain a dendritic tree where several correspondences are represented, and should project a superposition of the corresponding manifolds.

In this brief discussion, we will focus mainly on whether biological systems (in this case the human nervous system) have sufficient neurons and synapses for implementing SCAPE. Regarding neuron numbers, we will look at area V1 (although we could have used other regions), and in particular what percentage of V1 neuronal resources SCAPE requires. Regarding synapses, we will use the average number listed in Table 8.1, i.e:  $10^4$ .

We will exemplify the constrained case first, by computing the requirements of a translation estimator (i.e.  $T_x, T_y$ ). Assuming the number of nodes in each map is  $10^4$  (resolution  $100 \times 100$ ), and the number of vote nodes is  $4 \cdot 10^4$  (an overestimate based on the estimation range  $[-100... + 100]$ ) then the total number of nodes required is  $2 \cdot 10^4 + 4 \cdot 10^4 = 6 \cdot 10^4$ . From Table 8.1 we see that the number of neurons in V1 is approximately  $10^8$ , therefore the percentage of V1 resources required by our translation estimator is:

$$P = \frac{6 \cdot 10^4}{10^8} 10^2 = 0.06\% \quad (8.1)$$

which does not seem over-demanding.

Regarding the number of synapses, since the average number of synapses ( $S$ ) per vote node is equal to  $2C/V$  where  $C$  is the number of correspondences and  $V$  is the number of vote nodes, then:

$$S = \frac{2 \cdot 10^8}{4 \cdot 10^4} = 5 \cdot 10^3 \quad (8.2)$$

which is within the acceptable average bound of  $10^4$  synapses per neuron, if one assumes that dendritic conjunctions can be established anywhere along the dendritic tree. If on the other hand, we assume that dendritic conjunctions can occur only at terminal dendritic branches, as depicted in Fig. 8.4, and if we assume that each dendritic segment contains on average two synapses and that each segment bifurcates, then the number of available correspondence detectors is just sufficient for the number of required synapses (i.e.  $5 \cdot 10^3$ ). Based on the above assumptions (i.e. bifurcation and two synapses per segment) note that the total number of synapses is  $\sum_{i=1}^x 2^i$  where  $x$  refers to the number of “dendritic layers” ( $n$  bifurcations lead

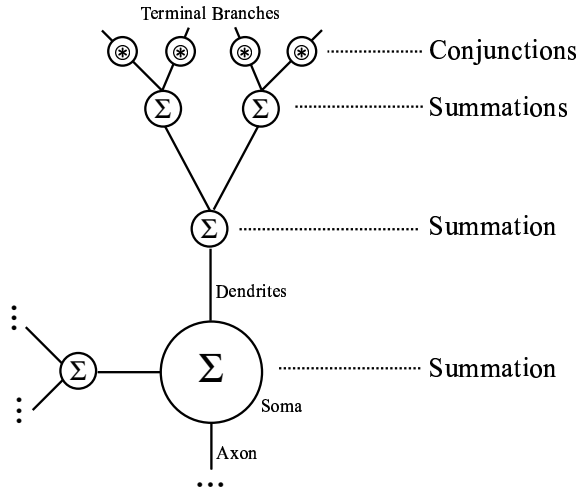


Figure 8.4: Dendrites with terminal conjunctions and inner summations.

to  $n + 1$  layers). Note also that the number of synapses at each layer is equal to  $p + 2$  where  $p$  represents the total number of synapses in all lower layers. Thus, the number of synapses at the terminal dendritic branches of any such neuron is approximately half of the total number of synapses. Since,  $(10^4)/2 = 5 \cdot 10^3$ , and this is the required number of synapses from Equation 8.2, the solution is still feasible within the constraints.

If we consider next a U-SCAPE architecture, then we most probably have to increase the abstraction of the features and reduce the map resolution to, for example,  $10^2$  nodes. If we consider the case of estimating similarity transformations (i.e.  $T_x$ ,  $T_y$ ,  $s$  and  $\theta$ ) and assume that  $r(s) = 10$  and  $r(\theta) = 32$  (where  $r$  as before stands for resolution) then the total number of nodes required is  $m + f + v = 2 \cdot 10^2 + 10^4 + 32 \cdot 10 \cdot 4 \cdot 10^2 = 2 \cdot 10^2 + 13.8 \cdot 10^4$  where  $m$  represents the number of map nodes,  $f$  represents the number of manifold nodes and  $v$  refers to the number of vote nodes. The percentage of V1 resources required is thus

$$P = \frac{2 \cdot 10^2 + 13.8 \cdot 10^4}{10^8} 10^2 \approx 0.14\% \quad (8.3)$$

which is still within acceptable bounds. Furthermore the average number of synapses per vote neuron (computed as the fraction between the total number of votes and the total number of vote nodes) is

$$S = \frac{32 \cdot 10 \cdot 10^4}{32 \cdot 10 \cdot 4 \cdot 10^2} = \frac{10^2}{4} = 25 \quad (8.4)$$

which is well below upper bounds.

In conclusion, it seems that the demands imposed by average C-SCAPE or U-SCAPE architectures, do not exceed typical empirical bounds, thus making the case for their plausible existence in biological systems stronger.

In the present chapter we discussed several lines of biological support for SCAPE architectures, namely: topographic maps, local features, neuronal variety and dendritic conjunctions. Biologically plausible architectures for constrained and unconstrained SCAPE were provided. Additionally, SCAPE was shown to satisfy some relevant biological constraints. In the following chapter developmental algorithms capable of generating SCAPE architectures will be investigated.

University of Malaya

# Chapter 9

## Development of Connection-Patterns

### 9.1 Introduction

Now that we have discussed an algorithmic approach to pose estimation, have analyzed its performance, have found artificial neural architectures that can implement it, have analyzed some of the combinatorial and spatial issues that these architectures raise and finally have found evidence for its plausible existence in biological systems, we are left with the fundamental question: how can biological systems develop such architectures? Even if a biological neural system has the machinery for implementing a neural architecture that embodies a particular algorithm, there is still the question of whether such an architecture can develop. If it is impossible to find plausible mechanisms through which such architectures can develop, then one has to conclude that, after all, and in spite of the theoretical possibility, the algorithm in question is unlikely to be implemented in biological systems. If on the other hand, plausible developmental mechanisms can be found, then this increases one's confidence in the biological plausibility of the approach. Furthermore, if this is the case, a new door is opened for a dialogue between artificial (e.g. Machine Learning) and biological (e.g. Developmental Cognitive Neuroscience) domains.

Computational systems such as brains undergo structural/functional changes, in response to the environment, and in order to increase an organism's chances

of survival. These changes can happen within different timescales, e.g: thousands or millions of years (evolution), months or years (development), seconds or minutes (learning) and milliseconds (adaptation). Although development, learning and adaptation are distinct processes<sup>1</sup>, some structural changes might be difficult to classify, lying somewhere between two categories, or might involve mechanisms from two or more processes. The formational processes discussed in this chapter fall more easily into the category of development, although they may borrow some machinery from learning. For this reason, the terms development and learning will be used interchangeably.

How difficult is the problem of developing/learning a SCAPE architecture? From an artificial perspective, and assuming that we maximally simplify the problem, the learning of connection patterns between map nodes and vote nodes is not difficult. If one assumes that each transformation encountered in a training set is unique, and that all possible correspondences are detected for each transformation, then learning can be as simple as assigning each new set of detected correspondences to a new vote node. The problem in the artificial domain becomes more complex when one considers images that are sparsely populated with salient features and/or that contain many repeated features (leading to a large proportion of false correspondences) and one starts to consider issues such as efficient coding (e.g. undercomplete vote representations<sup>2</sup> with broadly tuned vote nodes) and feedback from other processing areas (e.g. motor and somatosensory feedback). In the biological domain the problem becomes even more difficult, and thus all the more interesting, e.g: 1) the axonal terminations of model nodes have to pair up with the axonal boutons of image nodes, whose precise origin (i.e. map coordinates) they are ignorant of, 2) vote nodes can accept only a limited number of synapses (e.g.  $\approx 10,000$ ) and are thus unlikely to start off with all required synaptic-pairs. In biological systems, there is no global intelligence that can guide which axonal bouton another one should be paired with,

---

<sup>1</sup>Each one of the processes has its own set of mechanisms (e.g. development is more genetically determined although in many cases it is stimuli dependent).

<sup>2</sup>Undercomplete vote representation: when the number of vote nodes is smaller than the number of transformations to be represented. Complete vote representation: when the number of vote nodes is equal to the number of transformations to be represented. Overcomplete vote representation: when the number of vote nodes is larger than the number of transformations to be represented.

contrary to the artificial domain. In biological systems, development/learning must progress, in most cases, through local “blind” mechanisms.

As already hinted at, the job of learning connection patterns between map and vote nodes can be facilitated by various feedback connections. For example, if a vote-node “tells” an organism that food can be found to its left, and then if subsequent to a motor compensation the organism gets contradictory somatosensory feedback (the hand “tells” the organism that there is no food), then the vote node can be signaled to unlearn the connections that led to the erroneous judgment<sup>3</sup>. This type of supervised learning unfortunately exhibits a circularity which we wish to avoid: it depends on the motor areas being able to compensate for the transformation which the system is trying to learn in the first place. How do the motor areas learn the compensatory transformations? In order to avoid any form of circularity, we chose an unsupervised approach which does not depend on any feedback signals. For this reason the approach might be called: unsupervised clustering of correspondences.

## 9.2 Main Elements

For our initial experiments regarding the modeling of SCAPE development, we called upon three main elements, partly chosen for their compatibility with crucial biological constraints and partly chosen for their effectiveness regarding learning transformations: 1) neural traces, 2) random axonal-growth and synapsing and 3) synaptic-pair fixation levels. Models of biological systems often attract unresolvable debates, mainly regarding what levels of abstraction should be represented (e.g. proteins, membrane potentials, firing rates, and so on) and which ones need not be represented. Therefore, and following our understanding of the advice of Churchland and Sejnowski in (Churchland & Sejnowski, 1992), we have followed the rule of thumb whereby the model is made as simple as possible without jeopardizing performance, and rich enough to support our hypotheses.

---

<sup>3</sup>Note that the problem could lie elsewhere, e.g: the connections between vote-nodes and motor processing areas, the connections between motor processing areas and motor-effectors, the connections between tactile sensors and somatosensory areas, and so on.



### 9.2.1 Neural Trace

It is a well known fact that neural activity affects the strengths of synapses. This can be observed in the context of learning, and more specifically, for example, in a Hebbian synapse where correlated activity between pre and post-synaptic neurons, increases the relevant connection strengths (see Haykin 1999). A neural trace is essentially the prolonging of this effect throughout time. A modified Hebbian rule for example (see Földiák 1991) might state that changes in synaptic strength are proportional to pre-synaptic activity and a temporal average of post-synaptic activity. As demonstrated in (Földiák, 1991), neural traces are very useful for capturing invariant properties of the environment. Interestingly, and maybe unsurprisingly, the same mechanism that throws away transformation information for the implementation of invariant feature-detectors, is useful for the development of architectures that can detect/estimate those same transformations. The idea is based on the notion that transformations are somewhat stable through time. If transformation  $X$  occurs between time  $t_1$  and time  $t_4$ , and node  $n_1$  is activated at  $t_1$ , then propagating some of  $n_1$ 's activity to the following time-step (neural trace), makes it more likely that  $n_1$  will be activated again for the same transformation  $X$ , thus accelerating the learning of “ $n_1$  represents transformation  $X$ ”.

### 9.2.2 Random Rewiring

SCAPE depends in a fundamental way on the detection of correspondences. As was discussed in Chapter 8, a plausible implementation of correspondence detectors in biological systems relies on pairs of synapses at dendritic branches whose non-linear interactions compute logical-AND operations. But how are the synapses formed? How does each axonal bouton “know” which other bouton it should pair with? It is difficult to imagine how the complex connection patterns demanded by the estimation of transformations can be completely determined by the genetic code<sup>4</sup>. We hypothesize here that most of the “guidance” is provided by the environ-

---

<sup>4</sup>Although recent evidence of experience-independent dendritic development in motion sensitive neurons in the *Drosophila* visual system - see (Scott et al., 2003) - might be seen to argue against this point.

ment: activity-dependent (more specifically, experience-dependent) development. We hypothesize further that environmental stimuli provide indirect guidance by influencing the structural stability of synaptic-pairs which are continuously generated/eliminated through a process of random rewiring. In other words, if rewiring is here assumed to be implemented by axonal growth/retraction<sup>5</sup>, then the molding of the correct connection patterns is established by axons connecting to random<sup>6</sup> neurons and dendritic branches (see Chklovskii et al. 2004 and Poirazi & Mel 2001), and the environment providing feedback on the relative validity of such connections. Note that biological neurons have an added disadvantage in that they place limits on the number of synapses that they can receive. Thus, if the number of possible correspondences is larger than this limit then it is impossible to have a situation whereby all correspondence detectors are available at any time for each node, thus precluding a type of development which simply places every possible synaptic-pair on each node and develops by changing the weights of each pair.

### 9.2.3 Fixation Levels

A fundamental ingredient to the process of random rewiring as already hinted at is the notion of the stability of synaptic pairs. Stable synaptic pairs are less likely to be eliminated. Through the guidance of environmental stimuli, synaptic pairs, which form randomly, can increase or decrease in terms of stability. We have termed the measure for synaptic pair stability “fixation level”. No synaptic pair can be assumed to be right or wrong from one environmental instance. Put crudely, fixation levels provide synaptic-pairs with an opportunity to prove themselves. In other words, fixation levels provide the medium in which learning takes place, whereby an organism can adapt to the statistics of the transformations it encounters. The most basic rule that determines how fixation levels change echoes the Hebbian synapse: 1) if a neuron is active and one of its synaptic-pairs is active<sup>7</sup> then the fixation level of

---

<sup>5</sup>It can also be implemented by structural modifications at the level of dendritic trees and/or dendritic spines.

<sup>6</sup>Random within a relatively local set of neurons.

<sup>7</sup>An active synaptic-pair denotes the detection of a correspondence.

the latter increases, 2) if a neuron is active and one of its synaptic-pairs is inactive<sup>8</sup> then the fixation level decreases. Two differences relative to a conventional Hebbian synapse must however be pointed out: 1) two synapses rather than one are involved in the process<sup>9</sup> and 2) the changes caused by the learning rule do not affect the efficacy (see Hebb 1949) of the presynaptic neuron causing the postsynaptic neuron to fire (as in the notion of connection weights) but rather, affect the probability of the synaptic-pair being eliminated<sup>10</sup>. Although the mechanisms underlying synaptogenesis and synapse elimination are still undergoing intense research and debate, several non-mutually exclusive candidates for how fixation levels might be implemented in biological systems include: 1) relative concentrations of AMPA receptors (see Lynch & Baudry 1984), 2) postsynaptic density protein 95 at glutamatergic synapses (see Cohen-Cory 2002) and 3) the gradual removal or addition of presynaptic structures such as vesicles and associated proteins (see Hopf et al. 2002).

### 9.3 Algorithm

In order to test our ideas on how SCAPE connection-patterns might develop, using the elements discussed in the previous section, we designed an algorithm consisting of two main parts. The first part models organism/environment interactions, whereby the data for learning (or the experience-dependent side) is obtained. The second part models the developmental process itself. Both aspects occur simultaneously, as would be expected in a realistic scenario.

Seeing that the development of a SCAPE architecture (SCAPED) is likely to require large amounts of data, we decided to place the problem of pose estimation in the context of motion analysis. The pose estimation problem is embodied in the

---

<sup>8</sup>An inactive synaptic-pair denotes that the correspondence that it represents has not been detected.

<sup>9</sup>The biological existence of this process is open to discussion and provides an interesting hypothesis to test.

<sup>10</sup>In reality, questions on whether “firing weight” and “structural stability” elements are mediated by overlapping mechanisms, leading to functional overlap (e.g. a synapse with more stability might be more effective in causing postsynaptic neurons to fire and a synapse with a small weight might increase the chances of it being eliminated) are still completely open to debate. Answers to these questions are likely to have a great impact on Neural Computation and Computational Neuroscience and related fields and subfields.

problem of estimating the transformations that relate image data from two time-steps, namely the current and previous time-steps, where the image data is obtained from a small window placed on a larger test image. Three situations are possible: 1) a dynamic window on a static image, 2) a static window on a dynamic image (video) or 3) a dynamic window on a dynamic image. All three situations are useful since transformations are relative. However, we chose to explore the first case (i.e. a dynamic window on a static image) since this provided us with the possibility of controlling which transformations occur. Furthermore, it is easier to control the statistical properties of static images, thus allowing a deeper understanding of the limits of the developmental processes under investigation. Various parameters were chosen to control the generation of synthetic images, the most important of which determine the sparsity of features and the probability of true correspondences. Algorithm 8 summarises the environment/organism model for the  $(T_x, T_y)$  case and its relationship to SCAPE development.

---

**Algorithm 8** SCAPED Environment

---

```

1: procedure SCAPED( $p$ )                                ▷  $p$  encapsulates several parameters
2:    $net \leftarrow \text{initSCAPENET}$                         ▷ Network Initialization
3:    $img \leftarrow \text{initImage}$                           ▷ Image Initialization
4:    $[win_Y, win_X] \leftarrow \text{initWindow}$                 ▷ Initial Window Coordinates
5:    $IV_{old} \leftarrow \text{getData}(img, win_Y, win_X, p.winRad)$   ▷ Old Input-Vision
6:    $[\Delta_y, \Delta_x] \leftarrow \text{getRandSpeeds}$         ▷ Initial Random Window Speeds
7:    $iteration \leftarrow 0$                               ▷ Iteration Counter
8:    $stillDeveloping \leftarrow true$                     ▷ Boolean for Development Loop
9:   while  $stillDeveloping$  do
10:     $win_Y \leftarrow win_Y + \Delta_y$                  ▷ Update the Window's Y Position
11:     $win_X \leftarrow win_X + \Delta_x$                  ▷ Update the Window's X Position
12:     $IV_{new} \leftarrow \text{getData}(img, win_Y, win_X, p.winRad)$   ▷ New Input-Vision
13:     $net \leftarrow \text{updateFixations}(net, IV_{old}, IV_{new}, p)$ 
14:     $IV_{old} \leftarrow IV_{new}$ 
15:    if  $rem(iteration, p.iterSpeedChange) \equiv 0$  then  ▷ Change speed?
16:       $[\Delta_y, \Delta_x] \leftarrow \text{getRandSpeeds}$     ▷ If remainder zero: new speeds
17:    end if
18:    if  $rem(iteration, p.iterRewire) \equiv 0$  then      ▷ Rewire?
19:       $net \leftarrow \text{rewire}(net, p)$                  ▷ Substitute weak synaptic-pairs
20:    end if
21:     $iteration \leftarrow iteration + 1$                 ▷ Next Iteration
22:     $stillDeveloping \leftarrow \text{checkStopConditions}$     ▷ Stop developing?
23:  end while
24: end procedure

```

---

One relevant parameter in Algo. 8 is *iterSpeedChange*. This parameter de-

termines how frequently the window speed (intensity and direction) changes. The main loop essentially revolves around updating the window position, remembering the current and previous time-step input-visions and updating fixation levels based on those input-visions and the existing synaptic-pairs. Every *iterRewire* iteration, the network fixation levels will be checked: the synaptic pairs whose fixation levels are smaller or equal to some minimum level are eliminated and replaced with new random synaptic-pairs. This is accomplished via the *rewire* function<sup>11</sup>. Note that certain elements have been omitted from Algo. 8 for economy of space, e.g: window boundary checks and boundary-to-origin saccades.

As has already been mentioned, fixation levels provide the main “space for learning”. Therefore the *updateFixations* function is our main concern here (see Algo. 9). It contains the rules through which fixation levels change thus indirectly determining which synaptic pairs are preserved and which eventually get eliminated. Before continuing it might be useful to briefly mention the contents of the SCAPENET data-structure (in Algo. 8 SCAPENET is abbreviated to *net*). The SCAPENET data-structure essentially consists of a fixed number of vote-nodes, each one of which contains a fixed number of synaptic-pairs, each one of which in turn, consists of a cyclop, an attractor, an activation (1/0) and a fixation level.<sup>12</sup> Each cyclop and/or attractor is defined by a pair of  $(y, x)$  coordinates that refer to an input-vision feature position. Initialization of the SCAPENET data-structure involves generating random synaptic-pairs and setting all activation and fixation levels to zero.

The first step undertaken by *updateFixations* (Algo. 9) is to compute the activations of synaptic-pairs and thus of their respective vote-nodes. This is accomplished by *compActivations* in Algo. 10. Essentially, *compActivations* is doing correspondence detection and accumulation. If both features represented by a synaptic-pair

---

<sup>11</sup>In the current version there is an important assumption: new random synaptic-pairs always contain one random axonal-bouton from the model map and one random axonal-bouton from the image map. This assumption seems to violate the locality (non-global guidance) principle of rewiring. However, one might counter-argue that simple chemical markers can distinguish between model and image axons, and thus mediate a process which disallows model-model or image-image synaptic-pairs. If this chemical-marker hypothesis is unfounded, it should not be difficult to extend the algorithm to include computational mechanisms that can filter out model-model and image-image synaptic-pairs.

<sup>12</sup>Future extensions should exploit the added flexibility (with arguably no loss of realism) entailed by dynamic numbers of nodes and maybe even synaptic-pairs.

---

**Algorithm 9** SCAPED Update Fixations

---

```
1: procedure UPDATEFIXATIONS(net,  $IV_{old}$ ,  $IV_{new}$ ,  $p$ )
2:   net  $\leftarrow$  compActivations(net,  $IV_{old}$ ,  $IV_{new}$ ,  $p$ )  $\triangleright$  Synpair and vote activations
3:   Votes  $\leftarrow$  getVotes(net)
4:   [winner, winActiv]  $\leftarrow$  getWinner(Votes)  $\triangleright$  Strongest node and activation
5:   net  $\leftarrow$  resetLosers(net, winner)  $\triangleright$  Activity of losers  $\leftarrow$  0
6:   net  $\leftarrow$  makeTraces(net, winner,  $p$ .maxTrace)  $\triangleright$  Traces for next iter
7:   for all  $v \in$  Votes do  $\triangleright$  Scan votes
8:     for all  $s \in$  getSynpairs( $v$ ) do  $\triangleright$  Scan synaptic pairs
9:       if  $v \neq$  winner AND  $s$ .activ  $\equiv$  0 AND  $s$ .fixat  $\leq$  0 then  $\triangleright$  R1
10:         $s$ .fixat =  $s$ .fixat - 0.5
11:       else if  $v \neq$  winner AND  $s$ .activ  $\equiv$  1 then  $\triangleright$  R2
12:         if  $s$ .fixat  $\leq$  1 then
13:            $s$ .fixat  $\leftarrow$   $s$ .fixat - 1
14:         else
15:            $s$ .fixat  $\leftarrow$   $s$ .fixat -  $\alpha/s$ .fixat
16:         end if
17:       else if  $v \equiv$  winner AND  $s$ .activ  $\equiv$  1 then  $\triangleright$  R3
18:          $s$ .fixat  $\leftarrow$   $s$ .fixat + 1
19:       else if  $v \equiv$  winner AND  $s$ .activ  $\equiv$  0 then  $\triangleright$  R4
20:         if  $s$ .fixat  $\leq$  1 then
21:            $s$ .fixat  $\leftarrow$   $s$ .fixat - 1
22:         else
23:            $decr = (\alpha/s$ .fixat) +  $\beta$ (winActiv/numSynpairs)
24:            $s$ .fixat  $\leftarrow$   $s$ .fixat -  $decr$ 
25:         end if
26:       end if
27:       if  $s$ .fixat <  $p$ .minFix then  $\triangleright$  Fixation boundaries
28:          $s$ .fixat =  $p$ .minFix
29:       else if  $s$ .fixat >  $p$ .maxFix then
30:          $s$ .fixat =  $p$ .maxFix
31:       end if
32:     end for
33:   end for
34: end procedure
```

---

(a correspondence detector) are matched (see line 9 in Algo. 10), this leads to the synaptic-pair being active, and to the total activity of its vote-node being incremented.

Subsequent to the computation of activations, Algo. 9 finds the node with the largest activation. The existence of winner and loser nodes is essential for the dynamics of fixation changes, as will be seen shortly. After finding the winning node, the algorithm proceeds to the computation of neural traces, which will be used in the subsequent iteration. We chose a very simplified/abstract implementation of a neural trace: all non-winner nodes are given a trace of zero, while the winning

---

**Algorithm 10** SCAPED Compute Activations

---

```
1: procedure COMPACTIVATIONS( $net, IV_{old}, IV_{new}, p$ )
2:    $Votes \leftarrow getVotes(net)$ 
3:   for all  $v \in Votes$  do ▷ Scan vote nodes
4:      $v.activ \leftarrow v.trace$  ▷ Copy trace from previous iteration
5:     for all  $s \in getSynPairs(v)$  do ▷ Scan synaptic pairs
6:        $[C, A] \leftarrow eachSynapse(s)$  ▷ Get cyclop (C) and attractor (A)
7:        $F_C \leftarrow IV_{old}(C_y, C_x)$  ▷ Cyclop feature
8:        $F_A \leftarrow IV_{new}(A_y, A_x)$  ▷ Attractor feature
9:       if  $F_C \equiv F_A$  then
10:         $s.activ \leftarrow 1$  ▷ Correspondence detected  $\rightarrow$  synpair active
11:         $v.activ \leftarrow v.activ + 1$  ▷ Increment vote-node activity
12:       else
13:         $s.activ \leftarrow 0$  ▷ Corresp.  $\neg$ detected  $\rightarrow$  synpair inactive
14:       end if
15:     end for
16:   end for
17: end procedure
```

---

node's trace consists of its activity level (a *maxTrace* parameter bounds the trace). After this, the algorithm scans all vote nodes and their respective synaptic-pairs in order to update fixation levels.

Four main rules (*R1*, *R2*, *R3* and *R4*) determine how fixation levels are modified. The first two rules (*R1* and *R2*) are concerned with synaptic-pairs in non-winner nodes while the last two rules (*R3* and *R4*) are concerned with synaptic-pairs in the winner node. Following is a brief list of the four rules and their functions:

1. *R1*. If a synaptic-pair in a non-winner node is inactive (i.e. it does not detect a correspondence) and its fixation level is zero or negative (i.e.  $fixat \leq 0$ ) then decrement the synaptic-pair's fixation by 0.5. Note that inactive synaptic-pairs in non-winner nodes with positive fixation levels are unmodified.
2. *R2*. If a synaptic-pair in a non-winner node is active (i.e. it detects a correspondence) then two outcomes are possible depending on the current fixation-level. If the fixation is smaller or equal to one then decrement it by one. Otherwise, if the fixation is larger than one, make the decrement inversely proportional to the fixation level, i.e:  $decr = \alpha / fixation$ . This inverse proportionality provides positive fixation-levels with added stability without sacrificing flexibility (e.g. ability to re-learn).

3. *R3*. If a synaptic-pair in the winning node is active then increment its fixation level.
4. *R4*. If a synaptic-pair in the winning node is inactive then two outcomes are possible. If the fixation level is smaller or equal to one, decrement the fixation by one. Otherwise, decrement the fixation by a quantity that is inversely proportional to the fixation level and directly proportional to the node's activity.

As already mentioned, the general form of the above rules is quite Hebbian in nature and so should not raise many eyebrows. Arguably the least intuitive rule is the second part of *R4* which states that the decrement should be  $decr = (\alpha/s.fixat) + \beta(winActiv/numSynpairs)$ . The first half of the term, as was already mentioned in *R2*, provides positive fixations with stability without sacrificing flexibility. Thanks to the second half of the term the stronger the activity of a winning node, the more strongly the fixation levels of inactive synaptic-pairs are decremented. The biological existence of such a rule is open to question and thus serves as an interesting hypothesis to test. It is useful, for example, in cases where two or more transformations compete for the same node, i.e: it makes it easier for one transformation to “kick out” the other. As will be seen, this is not always desirable, e.g: some efficient-coding schemes, such as broadly tuned nodes, require the co-existence of multiple transformations in the same node. The notion of transformations competing for the same vote node will become clearer in subsequent subsections after discussing several examples.

At this point it might be useful to summarise three important simplifying assumptions of our model: 1) eliminated synaptic-pairs are substituted without any delay and through a random selection based on a uniform probability distribution, 2) the neural trace is highly simplified and 3) vote nodes generate a single winner which is computed within one iteration. The fact that the learning rules depend on finding the node with the largest activity is somewhat of an exception to the locality of the model (i.e. synaptic-pairs are generated, modified and eliminated through local interactions). However, it should be possible to implement this non-local aspect through a simple competitive process, which can be implemented in networks such



as the ones discussed in Chapter 6 in the context of vote sharpening. This consistency adds to the plausibility of the algorithm, i.e: the same competitive network used for developing a SCAPE network can subsequently be used for vote-sharpening during pose estimation.

Figure 9.1 demonstrates a sequence of SCAPE developmental stages based on Algos. 8, 9 and 10. The state of the network is depicted at three iterations (i.e. 6, 787 and 5241). The network consists of nine vote nodes each one of which contains 27 synaptic-pairs. Each synaptic-pair detects a particular correspondence, which in turn estimates one particular transformation (seeing that we are dealing with the constrained  $T_x T_y$  case). In Fig. 9.1 vote nodes are represented by subplots whose axes are numbered from 0 to 4 and synaptic-pairs are represented by lines. The origins of lines are represented by blue circles and the targets are represented by green squares. The coordinates of the origin for a particular line are derived from its cyclop (the synapse in the synaptic-pair which originates from the previous time-step map). The coordinates of the target for a particular line are derived from its attractor. If for example, a cyclop is at coordinate  $[y = 0, x = -1]$  and an attractor is at coordinate  $[y = 0, x = 0]$  this supports the transformation  $[T_y = 0, T_x = +1]$ , which is eventually represented by vote node  $V3$  (top-right sub-plot of the bottom set of nine sub-plots).

As we can see from Fig. 9.1, “vote node transformations” (short for “the transformations represented by the various synaptic-pairs in a vote node”) are randomly organized at the early stages of development (i.e. iteration 6). As development progresses (e.g. iteration 787), each vote-node starts to specialize in particular transformations, although many incorrect/random synaptic-pairs still abound. At the last stage of development, after the network has converged, each vote-node represents a single transformation, and all<sup>13</sup> transformations are represented by a distinct node.

Figure 9.2 illustrates SCAPE development from the perspective of fixation-level dynamics. Fixation states are depicted at six different iterations. Each iteration is represented by a rectangular matrix where each row represents a different vote-

---

<sup>13</sup>In this case, the environment presented the network with 9 different transformations  $T_x$  consisting of different combinations of  $T_x$  and  $T_y$  both ranging between  $-1$  and  $+1$ .

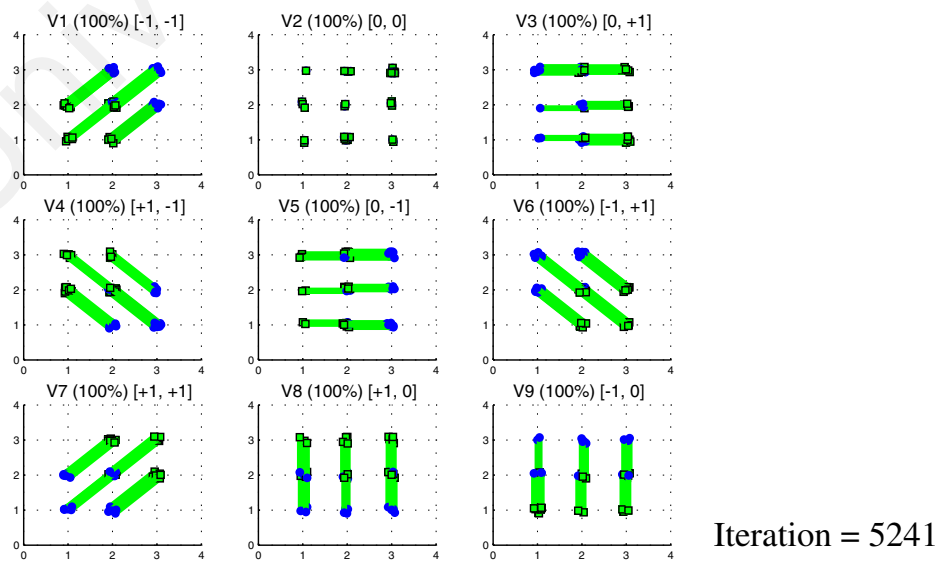
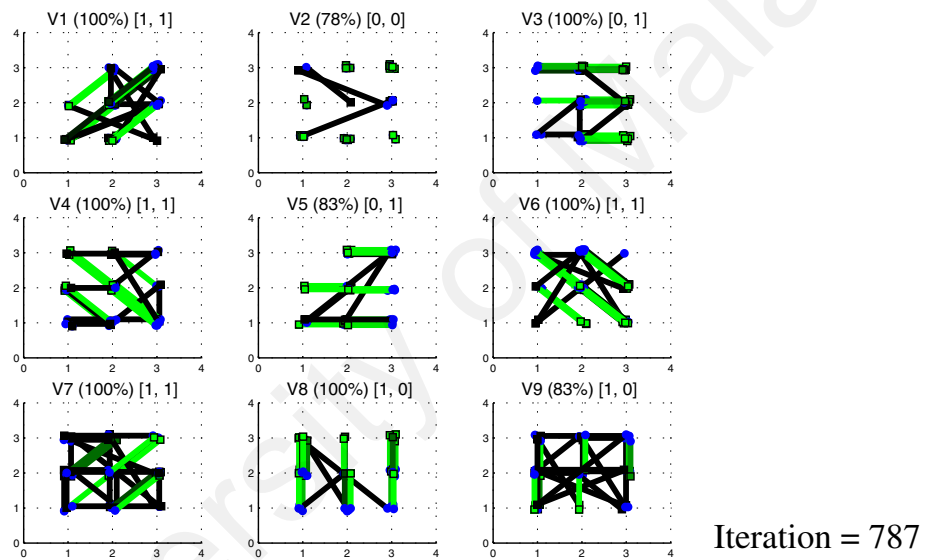
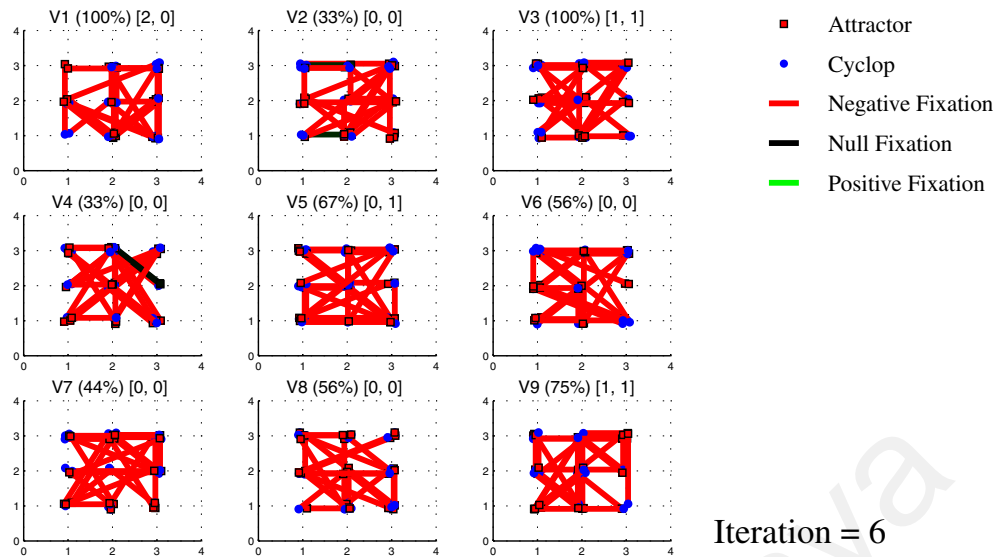


Figure 9.1: Three stages of SCAPE development.

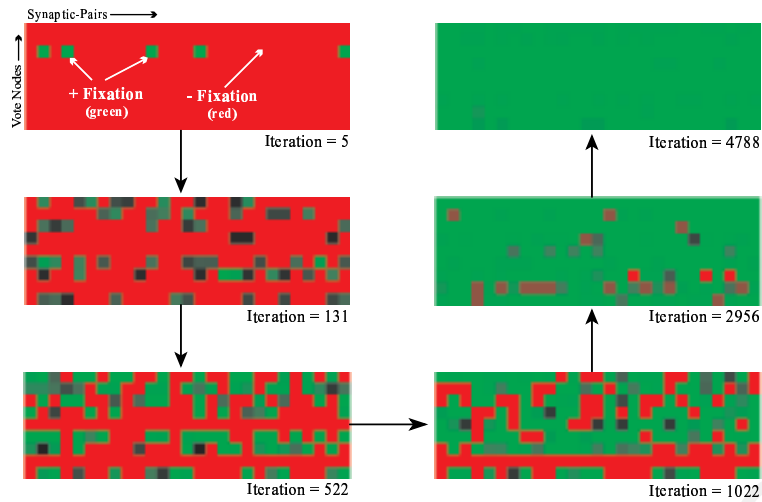


Figure 9.2: Changes in fixation levels.

node and each cell (or square) represents a different synaptic-pair. The fixation-level of a synaptic-pair is represented by colour: redness is proportional to the magnitude of negative fixations while greenness is proportional to the magnitude of positive fixations. Figure 9.2 shows how the totality of fixation-levels gradually change from red to green and how, due to probabilistic reasons that will become clearer in subsequent subsections, this progression slows down significantly towards the later stages of development.

## 9.4 Experiments

In order to get a better feel for the performance strengths and limitations of the ideas introduced above, this section discusses the effects of several parameters. Unfortunately the number of free parameters is not small enough to enable a complete study of all parameters and their combinations, hence it is necessary to focus on the most critical ones individually. To further facilitate the investigation, a small-scale problem was used, i.e: window motion was restricted to nine possible translations consisting of  $T_x$  and  $T_y$  combinations, each within the range  $[-1, +1]$ .

Table 9.1 summarises the main parameters used in Algo. 9 with their standard settings. The *Speed Change Interval* parameter as already mentioned determines the number of iterations that need to pass before the window's speed changes. The *MaxTxSpeed* and *MaxTySpeed* parameters determine the maximum magnitudes of

Table 9.1: Standard SCAPED parameters.

Parameter	Setting
Speed Change Interval	3
MaxTxSpeed	1
MaxTySpeed	1
Number of Votes (V)	9
Max Fixation	20
Min Fixation	-1
Elimination Fixation	-1
Elimination Interval	10
$\alpha$	0.3
$\beta$	1
Correspondences per Node (CpN)	$3 \cdot (C/V)$
Max Trace	$CpN/10$
Window Radius	1

the window's speed for both the  $x$  and  $y$  axes. The *Number of Votes* parameter determines that number of votes comprising the network. The *Max and Min Fixation* parameters determine the maximum and minimum fixation levels synaptic-pairs are allowed to express. If a synaptic-pair possesses a fixation level which is equal or inferior to the level defined by the *Elimination Fixation* parameter it is eligible for elimination and substitution by a new random synaptic-pair. The *Elimination Interval* parameter determines the frequency with which synaptic-pairs go through rewiring (i.e. elimination and substitution of weak synaptic-pairs). The  $\alpha$  parameter can be found in rules R2 and R4 of Algo. 9 and the  $\beta$  parameter can be found in rule R4 also of Algo. 9. The *Correspondences per Node* parameter determines the number of synaptic-pairs that each vote node can contain. The *Max Trace* parameter defines an upper bound on the neural trace. The *Window Radius* parameter determines the window's dimensions, e.g: a radius of two, defines a side of five and a total of 25 inputs.

#### 9.4.1 Time vs. Fixation

This first experiment gives us a feel for how development converges by observing fixation percentage relative to the progression of time (measured in iterations): see Fig. 9.3. Fixation percentage refers to the percentage of synaptic pairs whose

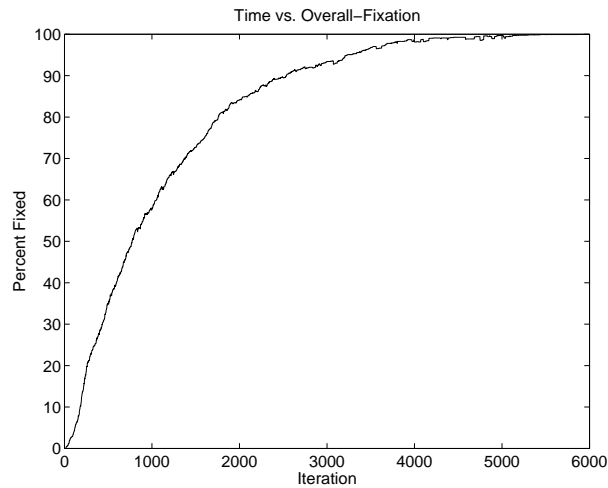


Figure 9.3: Time versus overall fixation level.

fixation levels are larger than or equal to half of the maximum fixation. Figure 9.3 was obtained by using the standard parameter settings in Table 9.1. The “percent fixed” value for each iteration was averaged from five separate runs. The shape of the curve is related to the gradually decreasing probability of getting the correct synaptic-pair at the right time.

### 9.4.2 Feature Sparsity

Earlier we mentioned that one of the advantages of using synthetic images for the environment was that we could control certain statistical characteristics about them. One such characteristic is feature sparsity. The feature sparsity of an image defines the proportion of space that is occupied by features. Maximum sparsity characterizes an image without features, while minimum sparsity characterizes an image where every location that can be sensed contains a feature. The simulations behind Figs. 9.1, 9.2 and 9.3 all used images with a large proportion of features (i.e. low sparsity). The parameter used for controlling feature sparsity is “feature percentage”. If an image is square and its side is  $n$ , it contains  $n^2$  pixels. If the “feature percentage” parameter is set at 50% then the number of features to be randomly placed around the image is  $0.5 \times n^2$ . Those pixels that do not receive a feature are considered transparent.

Feature sparsity is an extremely important variable which greatly affects the difficulty with which SCAPE networks can develop. The smaller the proportion of

features in an image (i.e. the greater the sparsity) the harder it is for SCAPE to develop. There are various reasons for this increased difficulty: 1) the probability of a correspondence being detected decreases<sup>14</sup>, 2) the probability of the formation of “vote unions” increases and 3) the probability of the formation of “greedy amalgams” increases. The term “vote unions” refers to the case when two or more vote-nodes represent the same transformation. The term “greedy amalgams” refers to the case when the same vote-node represents two or more transformations. Although vote unions and greedy amalgams can be useful in certain circumstances, here they are disadvantageous because, in the case of unions, some transformations might not have a vote node to represent them (due to “node stealing” by unions), while in the greedy amalgam case, the two or more transformations which co-exist in the same node are ambiguously represented. Furthermore, in the case when the number of vote nodes is fixed and equal to the number of transformations in the environment, the occurrence of greedy amalgams usually leads to another phenomenon which we denote as “deprived votes”. The fixation levels in deprived votes resist being increased mainly because greedy amalgams steal their transformations and thus activity levels.

Why does feature sparsity tend to lead to the formation of vote unions? A simple example might help explain this phenomenon. Consider a vote node  $v_a$  which has just increased the fixation level of synaptic-pair  $s_{a1}$  as a consequence of the occurrence of transformation  $t_1$ . Consider next, two scenarios, one where there is no feature sparsity and one where there is considerable feature sparsity. In the first scenario, if at a later stage, transformation  $t_1$  occurs again, then because the environment is saturated with features, the correspondence represented by  $s_{a1}$  is likely to occur again, and thus  $v_a$  is likely to be the node with the highest activation (i.e. is more likely to be the winning vote node) and any new synaptic-pairs representing transformation  $t_1$  in  $v_a$  will have their fixations increased. In the second scenario, because the environment is sparsely populated with features, then when transformation  $t_1$  occurs again at a later stage, the probability that the correspondence

---

<sup>14</sup>In the non-sparse case: the right synaptic pair at the right time is required. In the sparse case: the right feature and the right synaptic pair are required at the right time.

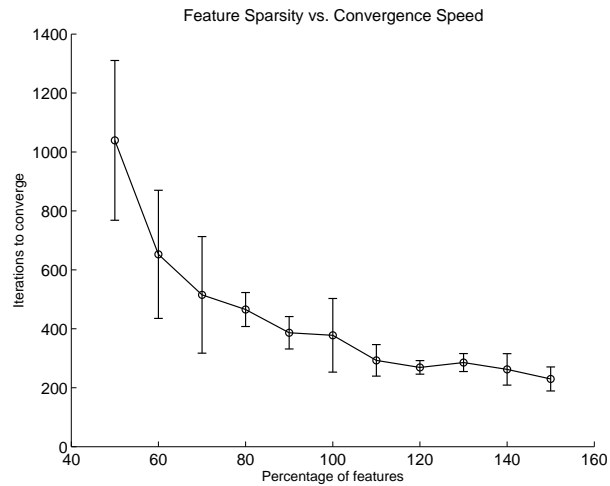


Figure 9.4: The effect of feature sparsity on convergence time.

represented by  $s_{a1}$  will occur and thus the probability that  $s_{a1}$  will fire is much lower, therefore greatly increasing the probability of other vote nodes becoming winners, and thus stealing activation, and finally forming unions.

Why does feature sparsity tend to lead to the formation of greedy amalgams? Probably the main reason for this lies in the fact that feature sparsity leads to low average activations which in turn makes it harder for the second part of rule R4 to kick out invading transformations. This can probably be compensated by extending the network with a sparsity analyzer which sends a measure of the environment's feature sparsity back to the main network.

Figure 9.4 illustrates the effect of sparsity on the speed of SCAPE development. The speed of development is here measured as the number of iterations the network takes before 15% of its synaptic-pairs are fixed, where a synaptic-pair is defined as fixed if its *fixation*  $\geq 10$ . Feature percentages (that determine sparsity) vary along the x-axis, in increments of 10, starting at 50 and ending at 150. At each feature percentage, 5 simulations were run so that the means and standard deviations of the number of iterations could be computed. Unsurprisingly, as feature percentage increases (i.e. sparsity decreases) the mean and standard deviation of the number of iterations required for "convergence" greatly decrease.

The crucial question is: can sparsity still lead to proper development? The notion of proper development can be measured in three non mutually-exclusive ways. One measure is the percentage of synaptic-pairs which are fixed. Obviously, if a network

is incapable of developing beyond the point where it has a 60% level of fixation one can safely conclude that its development is unsuccessful. However, the contrary is not necessarily true, i.e: a network where all synaptic-pairs have reached near-maximum levels of fixation is not necessarily correct. In order for it to be correct it needs to be capable of estimating the transformations that it experienced during development. Therefore the two remaining measures are “transformation coverage” (or T-Coverage) and “correspondence coverage” (or C-coverage). If the environment manifests 10 different transformations, and vote nodes represent only 4 of these transformations, then the network’s T-Coverage is 40%. Correspondence coverage, on the other hand, is concerned with the innards of each vote-node. If vote-node  $v_1$  represents transformation  $t_1$  and transformation  $t_1$  has a maximum of 10 different correspondences which represent it, and the synaptic-pairs in  $v_1$  only represent 8 of these correspondences, then  $v_1$ ’s C-Coverage is 80%. The most successful network, in this context, is therefore one that exhibits 100% fixation, 100% T-Coverage and where all nodes exhibit 100% C-Coverage.

Figure 9.5 shows how SCAPE networks can still develop in spite of sparsity, albeit with some deterioration in terms of C-Coverage<sup>15</sup>. Standard parameter settings (see Table 9.1) were chosen for running the simulations. The graph on the left of Fig. 9.5 depicts the case where the feature percentage parameter was set to 150% (i.e. low sparsity) while the graph on the right depicts the case where the parameter was set to 50% (i.e. high sparsity). Each figure averages the curves obtained from 20 separate simulations. The fixation goal for the graph on the left was set at 95%, while the fixation goal for the graph on the right was set to 75% due to the slow convergence caused by feature sparsity. Surprisingly, T-coverage is still quite strong for the sparse case. Future extensions should look into ways of improving and accelerating C-Coverage for the sparse case.

What happens if we increase sparsity further, e.g: a feature percentage of 20%? One might be tempted to think that increasing the speed-change interval might help by allowing the neural trace more time to attract synaptic-pairs to the same vote-

---

<sup>15</sup>Note that the C-Coverage curves in Figs. 9.5(a) and 9.5(b) represent the averages of all vote-node C-Coverages for each case.



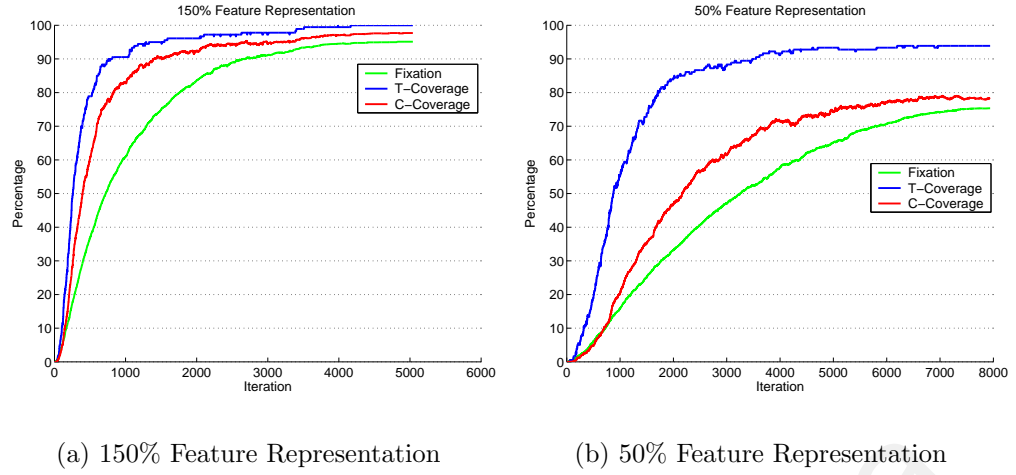


Figure 9.5: The effect of feature sparsity on development.

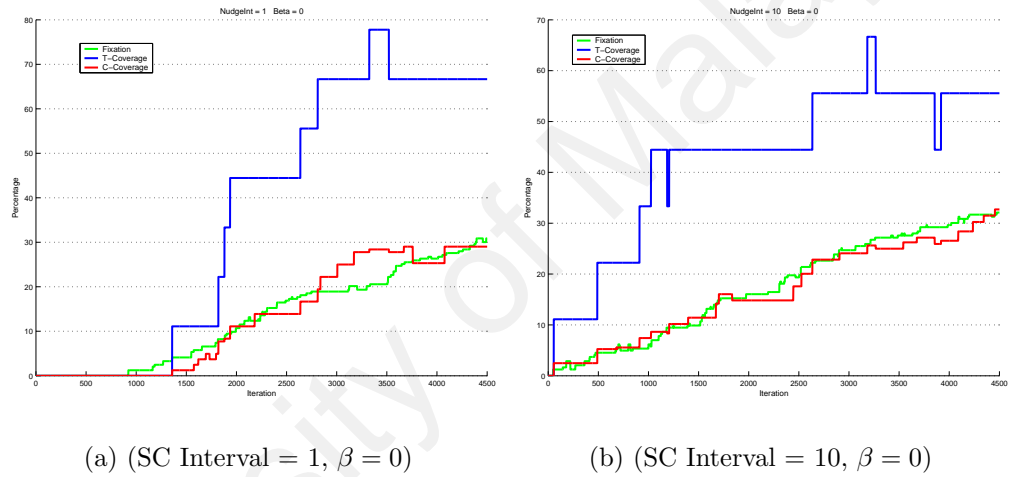


Figure 9.6: Larger speed-change intervals do not help the sparse features case.

node. Unfortunately, as Fig. 9.6 suggests, this expectation seems to be unfounded. Standard parameter settings were used in both simulations (run up to iteration 4500) with the exception of  $\beta = 0$  and the fact that on the left-hand side the speed-change interval was 1 while on the right it was 10. As the figure indicates, increasing the speed-change interval does not seem to lead to any significant change regarding performance curves.

The difficulties faced by the development of SCAPE networks in environments with sparse features is not really a disappointing result, if one remembers that natural images are actually replete with features of different hues and levels of saturation and brightness. Therefore one might say that sparse environments are unrealistic and are only of theoretical interest. Furthermore it is a well known fact

that many cognitive systems in biological organisms can not develop to their full potential in impoverished environments. Although the link between impoverished and sparse environments is to some degree indirect, it is strong enough to further support the conclusion that sparse environments are somewhat unrealistic.

### 9.4.3 False Correspondences

Another important characteristic of an image which can have a great impact on the development of SCAPE networks is the repeatability of features. The problem with repeating features<sup>16</sup> is that they produce false correspondences and the problem with false correspondences is that they mislead the developmental process by allowing false synaptic-pairs to be learnt along with true synaptic-pairs, thus leading to greedy amalgams.

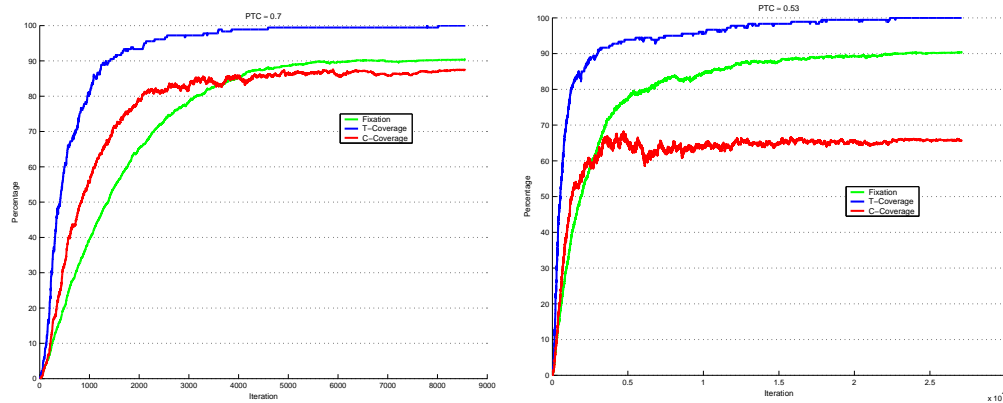
The synthetic image generator uses the “percent repeating” parameter to control the proportion of repeating features in generated images. If feature-percentage is 50% and if percent-repeating is 20% and if the total number of pixels in the image is  $p$ , then the number of features is  $0.5 \cdot p$ , the total number of repeating features is  $0.2(0.5 \cdot p)$  and the total number of non-repeating features is  $0.8(0.5 \cdot p)$ . The repeating features (with a single brightness value) are randomly layed out as with the non-repeating features.

When the window radius is one (i.e. a window with 9 inputs), feature percentage is 150% and the percent repeating parameter is set to 20% the average probability of a true correspondence (PTC) is approximately 0.7. If on the other, for the same window radius and feature sparsity, the percent repeating parameter is set to 30% then the average PTC is approximately 0.53.

Figure 9.7 shows how a relatively large proportion of false correspondences can still lead to acceptable development. The fixation, T-Coverage and C-Coverage curves in both figures represent averages from 20 simulation runs. Standard parameter settings (Table 9.1) were used with the exception of the speed-change interval which was set to 5,  $\beta$  which was set to 2 and, of course, the percent-repeating param-

---

<sup>16</sup>An image consisting of a single hue, saturation and brightness, has the largest number of repeating features amongst images with the same dimensions.



(a)  $PTC \approx 0.7$

(b)  $PTC \approx 0.53$

Figure 9.7: The effect of different probabilities of true correspondences.

eter. In both cases, the simulations were programmed to stop at a fixation percentage of 90%. Unsurprisingly, the left-hand case (i.e. larger PTC) converged faster than the right-hand case. In both cases, T-Coverage reached the 100% level. The main victim of low PTCs, as in the sparsity experiments, seems to be C-Coverage as illustrated in the right-hand side graph.

One interesting problem that is evident in the low PTC case is that if vote-nodes develop at different rates, the faster nodes, eventually start to steal activity from the slower, eventually impeding further progress from the latter and thus ultimately leading to the formation of deprived nodes. Faster nodes are capable of stealing activity mainly because of false correspondences. A transformation might be new (i.e. it might not yet be represented by any vote-node) thus increasing the probability of a new vote node being recruited, but if many nodes are already developed, then the false correspondences that come with the new transformation (due to repeating features), will tend to activate one of the older/faster nodes.

Again, as in the case of sparsity, the issue of false correspondences is not as critical as one might initially suspect, since natural images in most circumstances, specially when analyzed at local scales, tend to have quite high levels of PTC.

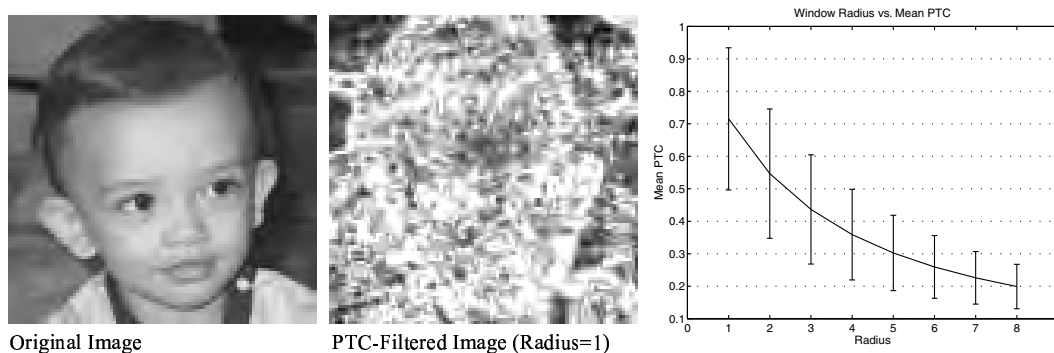


Figure 9.8: A natural image and its PTC-filtered version.

#### 9.4.4 Natural Images

In the previous two sections it was argued that SCAPE development in sparse and/or low-PTC environments was more of a theoretical question and that in practice, natural images were densely packed with features that rarely repeated at local scales. In the current subsection this argument will be verified.

The middle image of Fig. 9.8 depicts a PTC-filtered version of the image on the left. A PTC-filtered image is obtained essentially by moving a window over a source image, and for each location computing the PTC of the image-patch relative to itself. A window of radius one was used in Fig. 9.8 where the maximum PTC obtained was 1 (represented as white) and the minimum was 0.1 (represented as black). Intermediate PTCs are represented by intermediate greyscale values. Upon immediate inspection of the PTC-filtered image one can conclude that on average the PTC level is significantly close to 1. The graph on the right of Fig. 9.8 illustrates the mean PTCs and standard deviations for the same source image but with different window radii. As one can see, the larger the window radius the smaller the mean PTC and standard deviation become.

The crucial question is then: can SCAPE networks develop adequately using natural images such as the one in Fig. 9.8? Figure 9.9 illustrates performance curves averaged from 20 runs, using the standard parameters in Table 9.1 with the exception of  $\beta = 2$ . The simulations were allowed to run up to a fixation percentage of 95%. Not only does the network seem capable of developing properly, this development appears to be better than what results from using non-sparse and large-PTC synthetic images (compare with the left-hand side of Fig. 9.5),

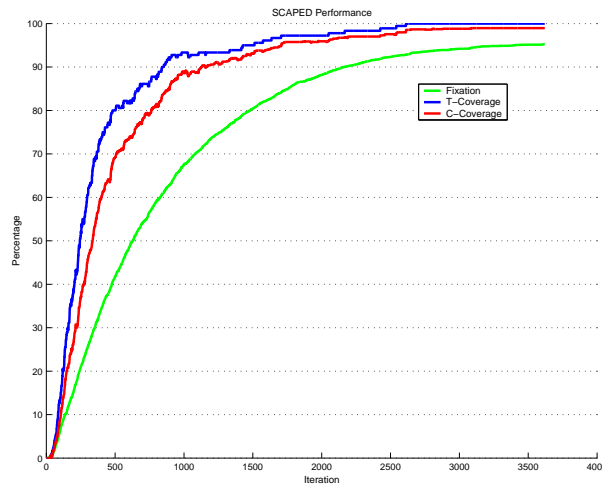


Figure 9.9: SCAPED performance on a natural image.

i.e: convergence is faster, and C-Coverage seems to be closer to 100%. This is most probably due to the fact that synthetic images with 150% features can still have some transparent pixels, and are therefore not completely non-sparse, due to the fact that features are placed randomly and blindly (i.e. without checking the contents of new random positions).

## 9.5 Rigid Transformations

Up to now we have dealt with SCAPE development in the context of translation estimation. As has been mentioned before, the problem of estimating translations from single correspondences is constrained. How does SCAPE development fare when confronted with an unconstrained problem? One significant difference about trying to learn unconstrained transformations is that any one correspondence needs to be represented in more than one vote node (vote manifolds). This creates certain difficulties that were not present in the translation estimation case, and thus calls for some modifications of the basic algorithm. For simplicity sake we chose rigid transformations for the unconstrained problem, i.e: translation and rotation combinations.

### 9.5.1 Algorithm

As already mentioned, unconstrained problems require the learning of manifolds, which in turn calls for some modifications to our main algorithm. Three modifications were applied to Algo. 9:

1. A dynamic-threshold was introduced thus enabling the existence of multiple winners: nodes whose activities are larger than or equal to the threshold are considered winners. The threshold is computed at each iteration by multiplying a threshold factor (e.g. 0.5) by the strongest activity.
2. Rule R2 in Algo. 9 was dropped since it conflicts with the formation of vote manifolds, i.e: it conflicts with the representation of the same correspondence in different nodes. So, in this case, rule R1 of Algo. 9 is applied to non-winners while rules R3 and R4 of Algo. 9 are applied to winners (as computed from the dynamic threshold).
3. The neural trace was eliminated since, one might argue that, the more complex a transformation is the less likely it is to persist in time, thus calling for the need to experiment with extreme conditions, e.g: no neural trace and/or a speed-change interval of 1.

### 9.5.2 Rotation

Before proceeding with rigid transformations per se, it might be useful to verify the behavior of our modified algorithm (from this point onwards referred to as Algo. 9-Dyn) when confronted with the problem of learning rotations (a constrained problem). To simplify matters further we have sampled four rotation angles, i.e:  $0^\circ$ ,  $+90^\circ$ ,  $-90^\circ$  and  $180^\circ$ .

Figure 9.10 illustrates four vote nodes progressing through four developmental stages. Each stage depicts the four vote nodes and the correspondences represented by their synaptic-pairs. Initially correspondences are random, but gradually, and culminating in iteration 4396, correspondences start to cluster in nodes in such a way that each node represents a different rotation. The environment for this experiment

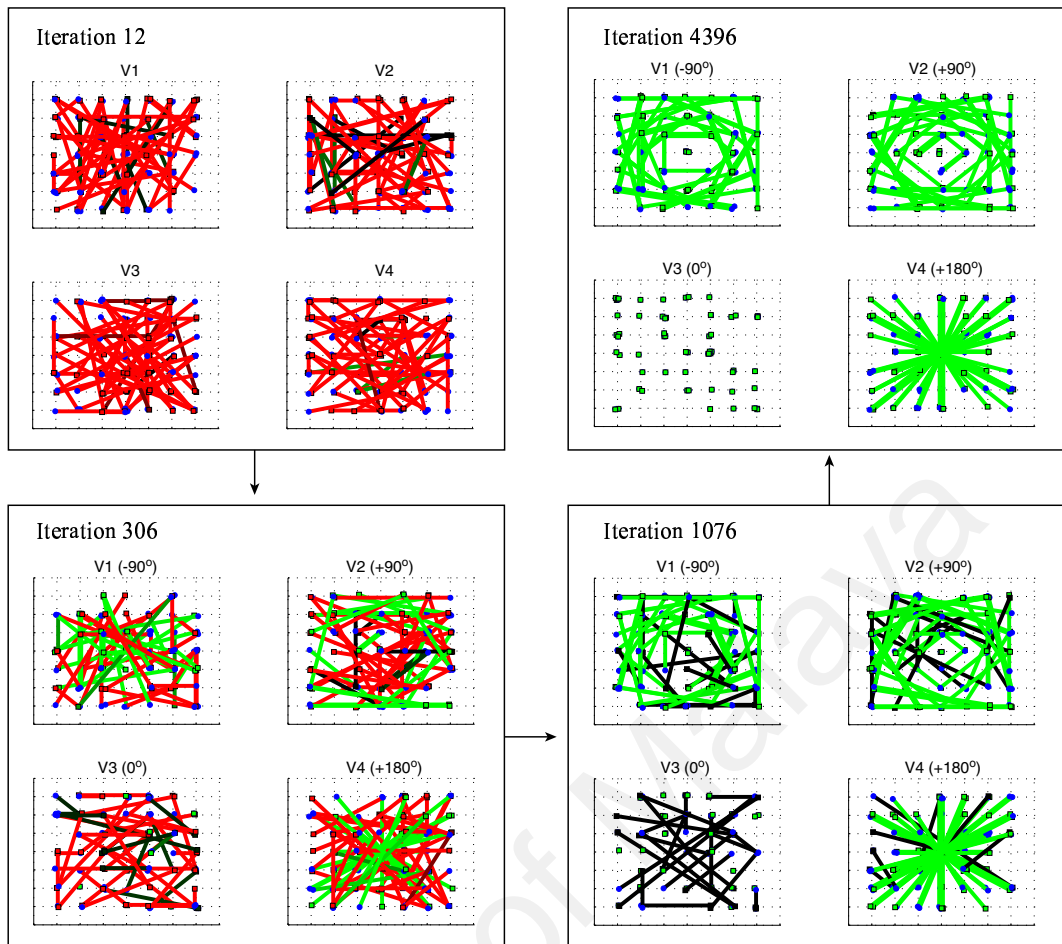


Figure 9.10: The development of rotation estimators.

consisted of the image in Fig. 9.8 and the parameters were set as follows: window radius = 3, speed-change interval = 3, number of votes = 3, maximum fixation = 5, minimum fixation = -1, elimination fixation = -1,  $\alpha = 0.3$ ,  $\beta = 3$ , elimination interval = 10, number of correspondences per node = 60 and threshold-factor = 0.9.

### 9.5.3 Translation and Rotation

To test Algo. 9-Dyn on a simplified unconstrained problem we chose the following transformation sets:  $T_x \in [-1, 0, +1]$ ,  $T_y \in [-1, 0, +1]$  and  $\theta \in [0^\circ, 180^\circ]$ , which total  $3^2 \times 2 = 18$  different transformation combinations. Although we have simplified the problem by greatly sub-sampling the ranges of translations and rotations used, the nature of the problem is the same, i.e: a complex “manifold” has to be learnt.

Figure 9.11 illustrates the 18 vote nodes at two different developmental stages. By iteration 10329 the network has learnt all of the transformation combinations.

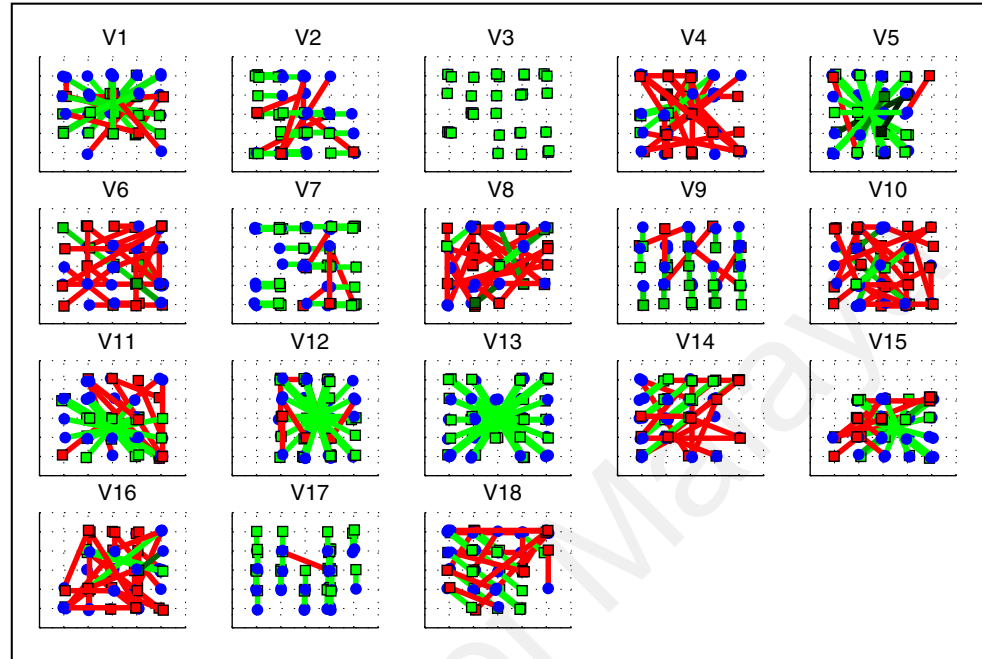
Notice that a title has been given to each vote-node which includes a reference to its transformation in the format  $(T_y, T_x, \theta)$ , where *theta* is given in radians. The environment for this experiment, as before, consisted of the image in Fig. 9.8 and the parameters were set as follows: window radius = 2, speed-change interval = 3, number of votes = 18, maximum fixation = 5, minimum fixation = -1, elimination fixation = -1,  $\alpha = 0.1$ ,  $\beta = 1$ , elimination interval = 10, number of correspondences per node = 30 and threshold-factor = 0.9999.

Although Fig. 9.11 depicts a successful development it is important to point out that Algo. 9-Dyn represents merely the beginning of the effort to find biologically plausible mechanisms for the development of SCAPE networks in the context of unconstrained problems. The main problem encountered by the algorithm, specially when confronted with more complex transformations, consists of differential rates of development. If some nodes develop faster than others, then the former are quite likely to steal activation from the remaining slower nodes (due to vote manifolds) thus impeding further development of the latter and eventually turning them into deprived nodes. One might think that decreasing the threshold-factor would counteract this effect by allowing a broader range of winners: this increases the probability of slow nodes retaining activation and thus of increasing relevant fixation levels. However, this is a double-edged sword since decreasing the threshold-factor increases the probability of the formation of vote unions, which in this context, is undesirable. One might argue that we have proved that it is possible, in principle, to develop SCAPE networks that can solve unconstrained problems using biological constraints. However, obstacles remain (probably mainly due to differential rates of development) that need to be addressed in order to scale the algorithms to larger and more realistic problems.

Preliminary experiments designed to overcome some of the problems posed by unconstrained transformations have shown that one of the main causes behind differential rates of development and incomplete SCAPE development is the existence of repeated synaptic-pairs (i.e. nodes that exhibit multiple copies of the same correspondence detector). Several simulations were run, wherein synaptic-pairs were



Iteration 2859



Iteration 10329

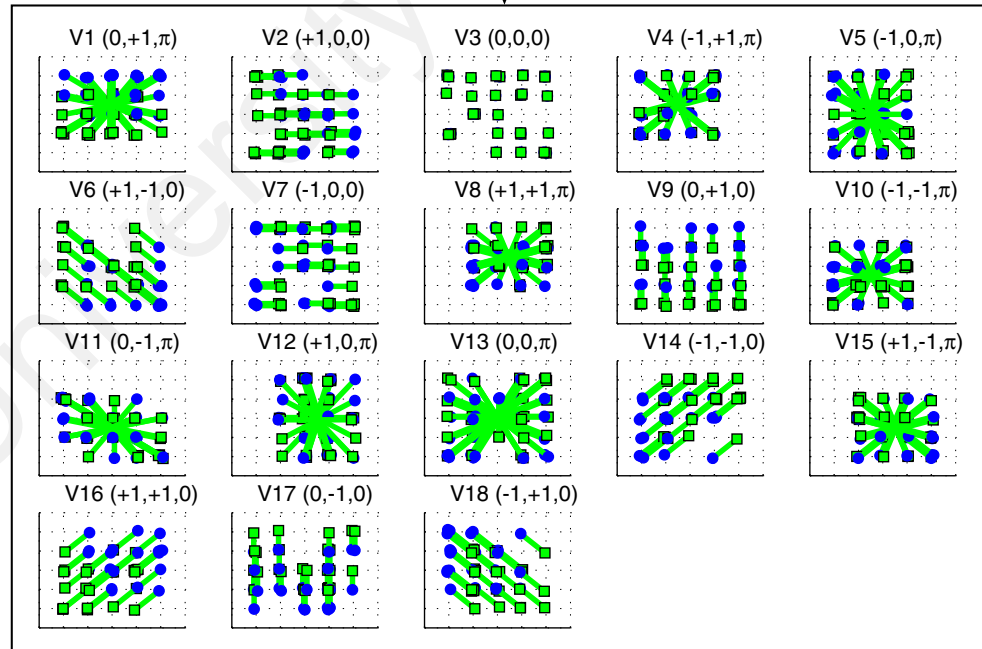


Figure 9.11: The development of estimators for rigid transformations.

forced to be unrepeating in the same node, and which demonstrated significant improvements of SCAPE development in the unconstrained case. Note that in this chapter, we have unfairly assumed a somewhat “dumb” developmental process in the sense that axons grow in random directions, whilst the role of guidance is relegated almost exclusively to visual experience. In reality, the developmental processes that form biological neural systems are extremely complex and rich. For example, there is ample evidence that the positional specificity of axons underlying many topographic maps is a result of rich chemospecific mechanisms where source and target neurons are labeled to a considerable degree (see Brown et al. 2001). One axon guidance mechanism, which is useful in our context, is repulsion. If axons can specifically repulse other axons originating from the same source node (and thus the same feature position), then there is a biological developmental mechanism by which correspondence detectors can be forced to not repeat in the same node. Future work will clarify the computational limits of this advantage and should expand the biological evidence supporting non-repeatability of correspondence detectors in the same node. Recent findings, for example, have shown that activity-dependent competition amongst axons can regulate (and thus indirectly guide) the growth of axons even before the formation of synapses (see Hua et al. 2005).

## 9.6 Probabilistic considerations

Since transformations occur randomly and synaptic pairs are formed randomly, and since SCAPE development depends crucially on the right synaptic-pair existing at the right time, we are dealing with a probabilistic problem. One question one might ask, which provides some understanding of the general shape of the fixation percentage curve (see for example Fig. 9.3) is: given a certain number of synaptic-pairs (regardless of the number of vote nodes) and given a particular range of transformations, what is the probability that at least one of the random synaptic-pairs matches the random transformation? Assuming constrained problems, in the case where there is no sparsity the probability is defined by:

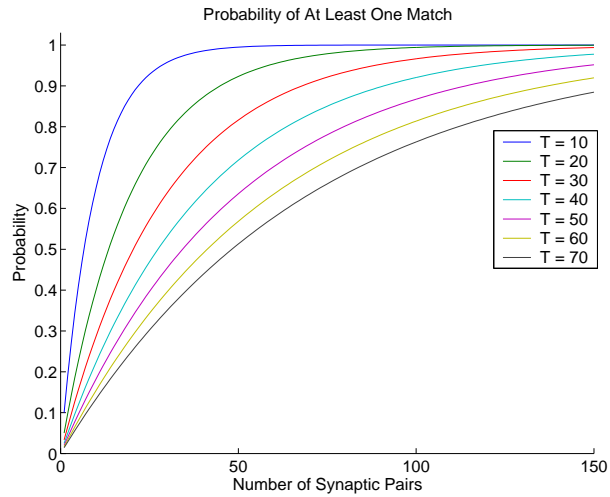


Figure 9.12: At least one match probabilities.

$$P(\text{match}) = 1 - \left(\frac{t-1}{t}\right)^s \quad (9.1)$$

where  $t$  represents the range of transformations that can occur and  $s$  represents the number of synaptic-pairs available (refer to section B.3.1 in Appendix B for a proof/explanation). Figure 9.12 depicts various curves representing different ranges of transformations for numbers of synaptic-pairs varying between 1 and 150.

In the case where there is the added factor of sparse features, the probability is defined by:

$$P(\text{match}) = 1 - \left(\frac{t-f}{t}\right)^s \quad (9.2)$$

where  $t$  and  $s$  are as in Eq. 9.1 and  $f$  refers to the probability of a feature occurring (refer to section B.3.1 in Appendix B for a proof/explanation).

Note that these equations cannot fully explain the dynamics of SCAPE development since there are other factors such as fixation levels, elimination intervals and so on, which play crucial roles. However, the equations do give an idea of how the numbers of synaptic-pairs and transformations affect the speed of convergence. For example, it should not be surprising that the closer a network gets to convergence (larger fixation percentages), the slower it proceeds since it has fewer free synaptic-pairs to work with.

## 9.7 Structurally Optimal Development

Although it has been demonstrated that SCAPE networks can develop obeying core biological constraints, nothing has been said about the structural organization of the resulting networks. We have argued that, proving that SCAPE networks can develop, provides further support for their biological plausibility. We have also argued that since biological architectures tend to be structurally optimal in that they minimize wiring-length, SCAPE networks in biological systems are also likely to be structurally optimized. Therefore, to make the developmental argument that much stronger we should be capable of answering the following question affirmatively: can networks develop not only the functionality of SCAPE but also, and simultaneously, a structurally optimal architecture? This is a big question, one to which a lot of time could be dedicated, therefore, we will aim to tackle it as briefly and essentially as possible, much like a proof of concept.

One of the design principles of structurally optimal SCAPE networks, that emerged again and again in our experiments (see Chapter 7), was that of topographical organization. Therefore, the question of the previous paragraph will be henceforth simplified to: can networks develop SCAPE functionality and simultaneously organize topographically? The answer, as will be shown, is affirmative.

A simple way in which this can be implemented to some degree is by using what might be termed a “trace neighborhood”. In a trace neighborhood, the winning neuron “generates” a neural trace for itself and for its neighboring neurons. This might be achieved for instance by having excitatory connections to neighboring neurons. Apart from the trace neighborhood, we also modified Algos. 9 and 10 to include a dynamic threshold as in Section 9.5. Furthermore, the window motion was changed so that it reflected a gradually varying force, which parallels most types of motions which can be observed in natural conditions. Both the gradual speed changes and the trace neighborhood conspire to making neighboring nodes represent neighboring transformations. Of course this is not a perfect solution, since the neighbor of a neighbor might be twice removed from the the first node, whilst it’s transformation can be one step removed from the first node’s transformation. Nev-

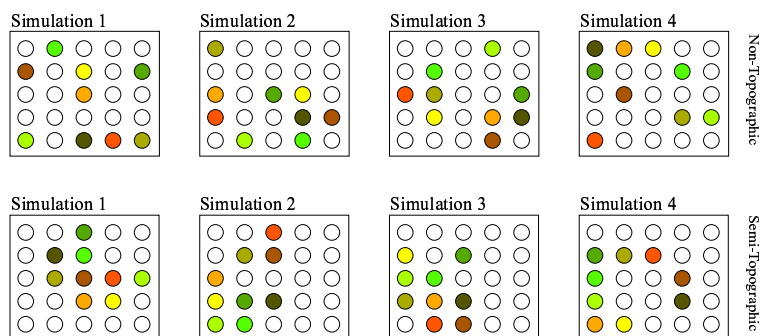


Figure 9.13: The development of SCAPE networks with a degree of topography.

ertheless, a considerable degree of topography emerges, specially when compared to a non-topographical development (without a trace neighborhood). The four network configurations on the top half of Fig. 9.13 represent non-topographical development, whilst the four configurations on the bottom half represent topographical development. The emergent topography is evident in the following facts: 1) functional nodes tend to clump together rather than be scattered (the white nodes are not representing any transformation), 2) there is a vague clustering of nodes based on colour (e.g. greener nodes to one side and redder nodes to the other side), where colour represents a transformation (the greenness of a node is proportional to the non-negativeness of its  $T_x$  representation while the redness of a node is proportional to the non-negativeness of its  $T_y$  representation).

The following parameters were used in the development of the networks in the top-half of Fig. 9.13: window radius = 1, maximum fixation = 10, minimum fixation = -1, elimination fixation = -1, elimination interval = 10, number of votes = 25,  $\alpha = 0.5$ ,  $\beta = 4$ , synaptic-pairs per node = 30, dynamic-threshold factor = 0.999999, maximum trace = 6 and trace neighborhood radius = 0. All parameters for the bottom-half networks were the same except for the trace neighborhood radius, which was set to one.

Clearly, much work remains to be done in order to allow the development of SCAPE networks that fully exploit structural optimization, but hopefully, some useful first steps have been made here.

In this chapter developmental (or formational) algorithms for the generation of SCAPE architectures were investigated. Certain plausible physiological elements

were proposed, e.g: fixation levels. The effects of feature sparsity and false correspondences were investigated through the use of synthetic patterns. Development in the context of real patterns was also demonstrated. Initial experiments attempting to combine developmental and structural optimization goals were conducted. In the following chapter we conclude the thesis by, among other topics, discussing several issues such as direct evidence and generalizations of SCAPE.

University of Malaya

# Chapter 10

## Discussion

*False facts are highly injurious to the progress of science, for they often endure long; but false views, if supported by some evidence, do little harm, for every one takes a salutary pleasure in proving their falseness.*

– Charles Darwin

### 10.1 Summary

This thesis began with the question of pose estimation. More specifically, it chose to investigate how biological neural systems (e.g. the human brain) might solve the problem of pose estimation. In a possibly unconventional way, it chose to do this by investigating not only biological (e.g. Computational Neuroscience) but also artificial (e.g. Computer Vision) and theoretical (e.g. Point Pattern Matching) domains, in search of the most neurally plausible approach, which turned out to be based on correspondences and voting, and gradually building an argument towards the biological domain. The thesis' structure followed an argument of accumulative consistency and functional refinement. Four levels of abstraction were followed in a gradual, stepwise fashion, i.e: computational, algorithmic, implementational and formational.

After fleshing out the correspondence and vote-based approach, certain accuracy issues were investigated, which confirmed the robustness and suitability of the algorithms for pose estimation in real environments. Subsequently, the embodiment of the algorithms in artificial neural architectures was investigated. The straight-

forward translation of the algorithms into neural architectures confirmed our initial prediction of their neural implementability. The resulting architectures were further analyzed and in particular structurally optimal configurations were found for them. The resemblance of some of the resulting architectures to topographic organizations found in biological neural systems, produced further consistency evidence for our main argument. Additional biological parallels were subsequently discussed, apart from recent Neuroscientific findings, which suggest that the fundamental computational element supporting the approach (i.e. conjunctions) is implemented at the level of dendrites, greatly strengthening the biological plausibility of the approach. Although the accuracy, efficiency, structural optimality and implementability of the architectures helped our argument, it was still necessary to ask whether such architectures could actually be “made” by biological systems (i.e. whether neural development could generate the architectures). The final element of our argument investigated this question and found that indeed, experience-dependent development can lead to the emergence of accurate and structurally-efficient pose-estimating architectures based on correspondence-voting.

In so far that multiple separate consistencies increase the probability of the veracity of some hypothesized fact, our main objective was achieved, i.e: suggesting, arguing for and refining a biologically plausible approach for pose estimation.

## 10.2 Strengths and Weaknesses of Correspondence Distributions

Probably one of the main advantages of using correspondence distributions for pose estimation, apart from their accuracy and robustness (e.g. clutter and occlusions), is their computational efficiency in parallel architectures. If one excludes the stages of feature extraction and vote sharpening, the actual estimation process takes a single computational step. The resulting architectures also exhibit the property of graceful degradation.<sup>1</sup> Another great advantage is the simplicity of the approach,

---

<sup>1</sup>If several correspondence detectors are destroyed, SCAPE should still estimate transformations quite accurately, much in the same way as when there are missing features.



which facilitates its analysis and extension. Moreover, and critically to the motivation underlying this thesis, is the neural implementability and biological plausibility of correspondence distributions. Part of the biological plausibility is derived from the fact that the architectures have the ability to autonomously emerge from the interaction between local Hebbian-like mechanisms and dynamic visual information. Another great advantage of the approach pertains to its modularity, e.g: new salient measures and local invariant features can be “plugged in” independently of the subsequent pose estimation step. Lastly, correspondence distributions are not restricted to the problem of pose estimation: the approach is quite general and applicable to other domains, as will be explored in some detail below.

One of the main disadvantages of correspondence distributions, in the context of pose estimation, stems from the usual resolution issues of classical Hough-based methods. Having said this, given a certain target level of accuracy, it is not hard to find a voting resolution which is not too expensive and satisfies the desired accuracy.<sup>2</sup> Probably the main disadvantage of the approach lies in its expensiveness: in serial computers it is temporally expensive, while in parallel computers (e.g. neural networks) it is spatially expensive. We will discuss the issue of expensiveness in greater length in section 10.4.

## 10.3 Direct Evidence

### 10.3.1 Multicolored Labeling

Recent neurohistological methods (see Gan et al. 2000 and Grutzendler et al. 2003) might provide the means to prove/disprove the hypothesis that some biological neural systems use correspondence distributions for solving the pose estimation problem. In these methods, multiply colored dyes<sup>3</sup> are applied to neural tissue via particle-mediated ballistic delivery, fully staining neurons in a Golgi-like manner. The fact that multiple neurons can be simultaneously labeled with different colors,

---

<sup>2</sup>Note that relatively recent extensions to the classical Hough Transform, such as the Hierarchical or Adaptive Hough Transforms, deal with the resolution problem quite effectively.

<sup>3</sup>Lipophilic dyes such as long chain dialkyl carbocyanines, e.g: DiO, DiI, and DiD.

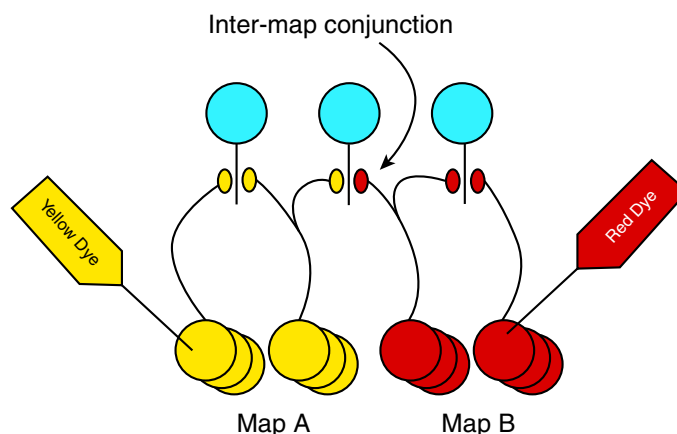


Figure 10.1: Visualizing inter-map conjunctions without positional information.

provides the opportunity to observe different aspects of neural interconnectedness, e.g: if one neuron is stained green and another one is stained red, one might find elsewhere, a green synapse adjacent to a red synapse, which by itself, starts to provide valuable structural and functional information regarding underlying circuits.

If it is possible to identify and target two neuronal maps, and apply differently colored labels to them, then it should be possible to deduce some of the computations taking place between them by observing the colors and placements of synapses at their target locations. If many adjacent synapses exhibit different colors, this strengthens one's confidence that correspondences are being represented. See Fig. 10.1 for a simplified illustration of the method.

Although the setup depicted in Fig. 10.1 is already useful for proving/disproving the fact that two maps interact at a higher computational stage, it is not sufficient to prove/disprove the fact that correspondences are being represented for the sake of computing poses. In order to do this it is necessary to label each map "position" (a neuron or a set of neurons) with a different color and then observe the color patterns of target synaptic-pairs. Figure 10.2 presents a simplified diagram of the method, applied to 1D maps. By observing the color combinations at synaptic-pairs it is possible to deduce what types of correspondences are being represented and thus what poses are being computed, e.g: all synaptic-pairs at the zero translation node exhibit same-color combinations. Note also the simplified depiction of a gene-gun delivering dye coated particles to map nodes.

The setup in Fig. 10.2 is quite attractive, and should not only prove/disprove

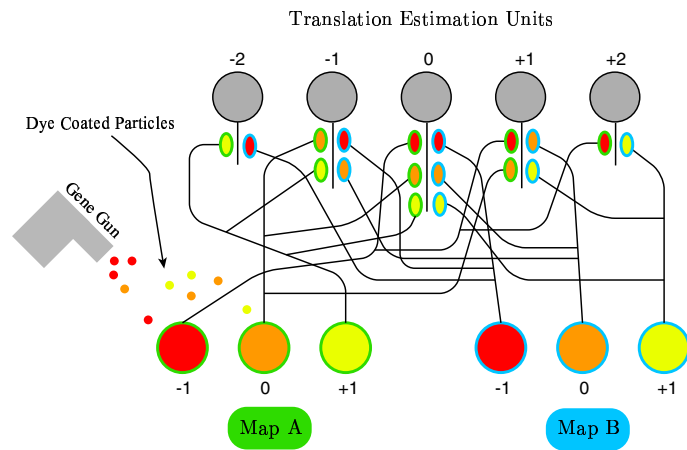


Figure 10.2: Visualizing inter-map conjunctions with positional information.

the use of correspondence distributions by biological neural systems, but should also elucidate other interesting inter-map computations. Unfortunately, there are still some practical obstacles that must be overcome before this is possible: 1) so far only about 7 distinct colors have been applied unambiguously<sup>4</sup> and 2) although labeling of dendritic structures is very good, labeling of axonal arbors at great distances has been less successful. Although, the task of investigating inter-map computations will be greatly facilitated by the time multicolor labeling techniques have reached full maturity, it is already possible to gain useful insights with the current techniques. Multiple slices, stained with different (carefully chosen) patterns of dyes, with subsequent analysis, should provide the required insight<sup>5</sup>.

### 10.3.2 Birds and Humans

Several experiments (see Hollard & Delius 1982 for example) have suggested that humans do not possess full rotational invariance when it comes to object recognition. More specifically, the speed of recognition seems to depend linearly on the amount of rotation applied to a target object. This implies that humans use some form of time-consuming (and iterative) mental-rotation in order to recognize rotated objects. This conflicts with approaches based on correspondence distributions, since their

<sup>4</sup>This is mainly due to the fact that color intensity decreases as the dye spreads away from the point of particle contact.

<sup>5</sup>For example, staining maps in one slice with horizontal dye bands, and maps in another slice with vertical dye bands, should provide some 2D combinatorial insight, in spite of the one-dimensional dye application.

estimates are computed in one-step, regardless of the amount of rotation. Thus, cognitive psychological evidence seems to speak against correspondence distributions for pose estimation in humans. This is not the end of the story however, seeing that many other biological neural systems remain to be investigated. The avian visual system, for example, seems to be a good candidate for having correspondence distributions implementing pose estimation. In (Hollard & Delius, 1982) it was shown that pigeons seem to exhibit pure rotational invariance: their recognition reaction times were independent of the degree of object rotation. Although the problem might be solved via invariant features alone, the fact that the behavior is consistent with the one-step computation of correspondence distributions, should prompt us to look further.

Why should birds and humans solve the rotational problem differently? A great part of a bird's visual experience is based on the horizontal plane (bird's eye view), while humans perceive mostly in the vertical plane. Thus patterns in different orientations are a common feature of avian visual experience, thus placing ecological pressure on the selection of systems which exhibit efficient processing in this regards. The fact that birds need to orient themselves in response to their interpretations of the underlying landscape indicates that the same pressure might have been applied to pose estimation. Interestingly, recent findings (see Kohler et al. 2005) indicate that Rhesus monkeys, whose environment exhibits a combination of vertical and horizontal perspectives (arboreal and terrestrial), can manifest both mental-rotation and rotation-invariance strategies.

## 10.4 Expensiveness

The expensiveness of correspondence distributions might explain why some species (possibly humans) might not use them for computing poses. Special requirements (e.g. rapid flight maneuvers) might justify the costs in the avian case. Significant cost is probably one of the main disadvantages of correspondence distributions, and maybe the reason why they seem to be absent from the neuroscientific literature.

Before proceeding, it might be useful to note that while cost is a difficulty,

it most certainly does not represent an impediment. In the business world, it is a well known adage, that you need to spend money to make money. In nature, many innovations involve significant increases in energy requirements, e.g.: warm-bloodedness, complex brains, and others. While for humans maybe the benefits of correspondence distributions do not outweigh the costs (thus allowing them to survive with an iterative mental-rotation approach), it is possible that for birds, the opposite is true.

For species that require some sort of compromise<sup>6</sup> between cost on one side, and speed and accuracy on the other, various solutions involving correspondence distributions are still possible, some of which were discussed in Chapter 7. For a more extreme solution, and using translation estimation as an example, the number of votes can be decreased so dramatically, that the only information preserved determines directionality alone (i.e. left, right, up, down and stationary). To push things further, the number of correspondence detectors can also be decreased, e.g: only horizontal and vertical correspondences might be represented.<sup>7</sup> See Fig. 10.3 for a diagrammatic representation of this solution: note that only the correspondences of the center source-map node have been depicted for the sake of clarity. Apart from the reduction in the number of vote-nodes (in this solution only 6 nodes are required), there are significant savings in regards to the number of correspondence detectors: instead of the usual  $n^2$  detectors, where  $n$  represents the number of map nodes (for one feature-type), only  $2n\sqrt{n}$  correspondence detectors are required. Although, these measures clearly compromise accuracy, input-clouds are still provided with useful information, which permit them to gradually approach and eventually latch onto objects significantly faster than a random guesser or an exhaustive search.

## 10.5 Artificial and Biological Synergy

Although this thesis emanates from an interdisciplinary effort, lying probably somewhere in between Neural Computation (more artificial) and Computational

---

<sup>6</sup>In general, as the cost decreases, so does the accuracy. The speed decreases as a consequence of accuracy loss, since several iterations are required to compensate for estimation errors.

<sup>7</sup>Salient features with no discriminatory labels should be more useful in this context.

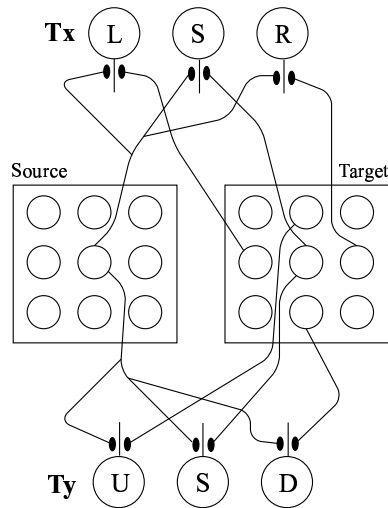


Figure 10.3: A very cheap translation estimator.

Neuroscience (more biological), there has been a distinct directionality in the argument and the motivation. The argument, motivation and end-results have been more biological, whereas the foundation has included artificial elements. In order to complete the cycle, it is necessary to point the arrow back to the artificial domain, i.e: what can the biological findings contribute towards artificial pose estimators?<sup>8</sup> The following list condenses some of the main elements of retribution to the artificial domain:

1. Economical architectural designs. One possible contribution to the artificial domain comes directly from the structural optimization experiments in Chapter 7. By trying to predict the configurations of SCAPE architectures in biological systems, several design principles emerged, which can be applied to the design of artificial circuits, seeing that they (the principles) lead to more economical configurations.
2. Neurocomputing hardware. Another possible contribution to the artificial domain comes from the observation that the evolution of technology (i.e. hardware) follows perceived needs. Hopefully, this thesis will have shown the advantages of a fully parallel pose estimating architecture (i.e. maximal speed), and will usher the development of novel neural computers that may implement correspondence distributions.

<sup>8</sup>Note that chapters 5 and 6 provided some direct contributions to the artificial domain in the form of novel accuracy analyses and artificial neural architectures.

3. Learning, development and manufacture. Other contributions come from the experiments on developmental algorithms in Chapter 9. The development of pose-estimators is useful in the artificial domain for various reasons: 1) more efficient codes/representations can be learnt, 2) new undefined transformations can be learnt prior to analysis and 3) estimators can adapt to new environments exhibiting new transformation statistics. Furthermore, and taking a speculative/futuristic stance, the experiments might impinge directly on neural nanotechnology (see Silva 2006 for an interesting review on the applications of nanotechnology to neuroscience), or more specifically on experience dependent self-assembly of nano-components.
4. Cognitive Neuroscience. One contribution which stems directly from Cognitive Neuroscience, concerns the use of parallel processing streams in primate visual systems (see for example Van Essen & Anderson 1990). Two such streams which are relevant to the current thesis, and which provide useful insights for artificial systems, are the dorsal and ventral streams. Both streams originate in the primary visual cortex (area V1), while the dorsal stream terminates in the inferior temporal cortex (area IT) and the ventral stream terminates in the posterior parietal cortex (area PP) (see Ungerleider & Mishkin 1982). The ventral stream is more concerned with invariant object recognition, while information regarding coordinates and transformations seems to be more preserved and computed in the dorsal stream. Both streams are likely to exhibit significant amounts of interaction, e.g: pose estimation allows for normalization which allows for recognition, while invariant recognition allows for relevant patterns to be brought forth for matching/comparison purposes and thus pose estimation. This approach of concurrently and interactively computing invariances and variant properties is likely to be useful in the artificial domain.
5. Visual functions. If the neural correspondence-distribution hypothesis is confirmed, that is, if some biological neural systems do indeed use correspondence distributions for computing poses, then further research into how they use them, might bring valuable suggestions for artificial applications. For exam-

ple, it might be found that correspondence detectors are actually more complex than a simple conjunction between two adjacent excitatory synapses. Some detectors might incorporate inhibitory synapses for implementing vote weights (e.g. the votes of rarer features should be heavier since they generate fewer false correspondences), or other strategies for greater robustness. Since the greatest disadvantage of correspondence distributions lies in their cost, it is possible that evolution has discovered more efficient correspondence-based solutions which nevertheless preserve accuracy and speed: this would be invaluable for artificial applications.

## 10.6 Generalizations

As already mentioned, one of the strengths of correspondence based approaches lies in their generality: correspondence distributions can solve different visual problems (e.g. pose estimation and shape representation) and they can be applied to different modalities (e.g. vision and audition). This generality is subsumed under the abbreviation CDA (correspondence distribution analysis), which distinguishes solutions along a representational dimension (e.g. cardinality and properties of correspondences, number of maps involved, connectivity patterns, and others) and a computational dimension (e.g. vote accumulation, sharpening, and others). Furthermore, computations involving correspondences constitute a particular instance of the more general class of “inter-map computations”, and thus provide a stepping-stone into this wider area, which we believe forms a significant part of the computations carried out by biological neural systems. In the following sub-sections we provide several examples that support the generality of correspondence distributions.

### 10.6.1 Visual Applications

This thesis was dedicated to a single visual problem: pose estimation. Hopefully, it was clearly demonstrated how correspondence distributions can elegantly solve the problem.

Motion analysis is another crucial problem where correspondence distributions



can be useful. One plausible implementation requires almost no modifications relative to the original pose estimation solution. Since a source and a target pattern are required for the detection of correspondences, all that is required in this context, is two patterns from two time-steps: current and previous. The transformation relating patterns from two nearby time steps provides useful motion information, e.g: “something is moving to the right and getting closer”.

Correspondence distributions should also be applicable to stereopsis, or depth perception. Human depth perception involves the integration of multiple cues, e.g: object size, shadows, motion and others. Disparity is another powerful cue, which is directly relevant to our argument. Because our eyes are at a certain distance from each other, objects project onto their retinas with some displacement relative to each other. This displacement is called disparity. One useful aspect of this phenomenon, is that disparity is inversely proportional to the distance of an object. Thus, if the disparity of object A is greater than that of object B, this means object A is nearer. In order to be able to compute disparity however, the brain needs to know which features of one retina match (or correspond) to which features on the other retina. This has been termed the correspondence problem, and is where correspondence distributions should be useful. Figure 10.4 depicts a simplified and plausible solution for the one-dimensional case. For the sake of clarity, only the correspondences of one map node have been represented. Note that the map features can be of any complexity level. The competition amongst depth nodes should be driven by the goal to maximize the number of neighboring positions with the same depth.

Shape representation is another domain where correspondence distributions can be useful. This application has been investigated to some depth in (Osada et al., 2002) for example, and has been referred to as “shape distributions”. In this context, shapes are characterized initially by undiscriminated but salient features. A shape distribution is generated by collecting all the correspondences of a shape relative to itself, and histogramming one or more statistics of the resulting distribution. One simple/effective statistic consists of the length of the correspondences, which if normalized, provides a scale and rotation invariant shape representation, useful for

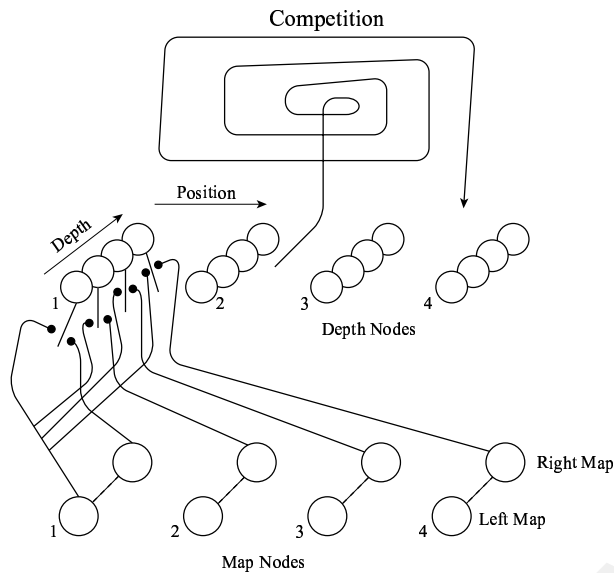


Figure 10.4: A possible correspondence-based architecture for stereopsis.

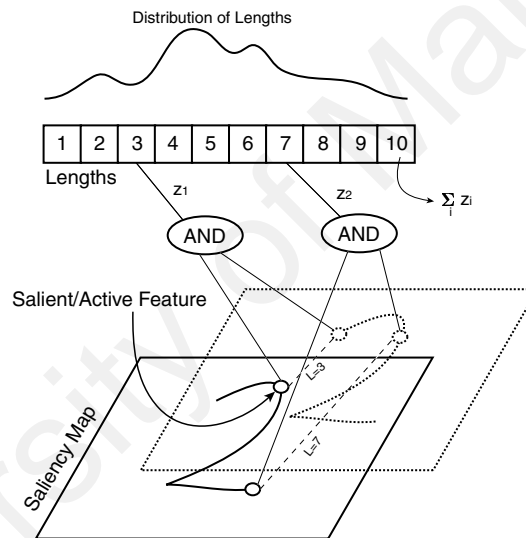


Figure 10.5: Artificial neural architecture for a shape distribution.

classification. Refer to Fig. 10.5 for a simple demonstration of the approach.

Correspondence distributions can also be applied to the computation of centroids and symmetry-axes. Assuming that shapes are defined by a sufficient number of features/points, a simple neural implementation of the approach could involve correspondences<sup>9</sup> and voting: see Fig. 10.6 for two 1D examples. Each pair of active features forms a conjunction at a second map, at a position representing the centroid of that pair. The centroid which receives the most votes represents the actual centroid of the shape and can be isolated by a fast competitive process. Note that, for the sake of clarity, only those conjunctions and connections involved in

<sup>9</sup>Note that we are dealing here with intra-map rather than inter-map correspondences.

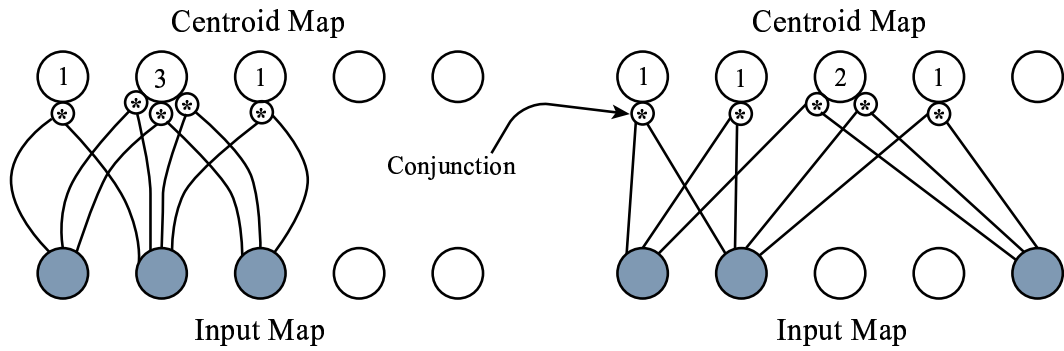


Figure 10.6: A neural architecture for finding the centroid of a shape.

the computation are represented.<sup>10</sup> A correspondence and voting strategy similar to the one used in Fig. 10.6 can be used for the computation of symmetry axes, where instead of each correspondence voting for a centroid it casts a vote in a 2D parametric space representing lines<sup>11</sup> (or symmetry axes).

Other visual functions to which correspondence distributions can be applied include: salience detection, edge detection, corner/junction detection and texture analysis.

## 10.6.2 Modalities

Not only can correspondence distributions be applied to different visual problems, but they should also be capable of solving problems in other sensory modalities, e.g: auditory, somatosensory, olfactory, and others. Consider for example the problem of sound localization. In one simplified solution/architecture (see Jeffress 1948 and Fig. 10.7), axons from both ears project with different lengths to target localization neurons, forming conjunctions. The different axonal lengths, lead to different signal arrival times, which lead to different conjunctions (or localization nodes) firing for different positions of the sound-source (due to different ear arrival times), e.g: sound C in Fig. 10.7 will activate a node towards the left, while sound A will activate a node towards the right. Although this model is most probably over-simplified (see Campbell & King 2004), it demonstrates a plausible underlying

<sup>10</sup>Note that for the case where features are sparse, a recurrent solution, where the results of each iteration are fed back into the centroid network via multiplicative conjunctions, should eventually converge onto the centroid.

<sup>11</sup>Each candidate symmetry axis is perpendicular to the line defined by the pair of points under consideration and intersects this line at the midpoint between the defining points.

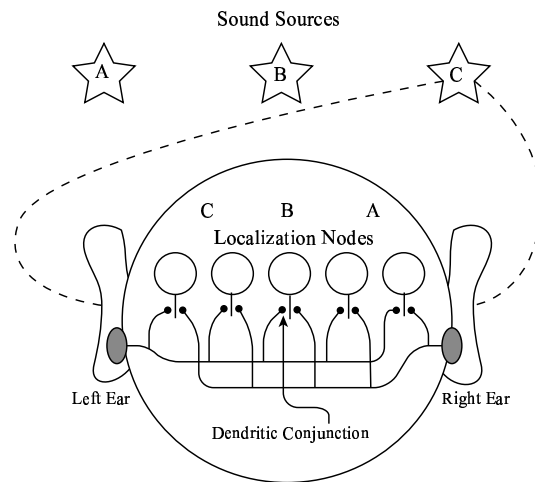


Figure 10.7: A simple architecture for sound localization.

ing mechanism for sound-localization, based on conjunctions, which are much like feature correspondences.

### 10.6.3 Inter-Map Computations

Computations involving correspondence distributions are part of the larger class of inter-map computations. A whole discipline can probably be dedicated to this class of computations. Sub-classes can be differentiated along several dimensions: 1) the number of maps involved, 2) the types of dendritic computations being performed (e.g. conjunctions, disjunctions, negations, and others), 3) connectivity patterns, 4) temporal dynamics, and others.

## 10.7 Future Work

Infinite questions can be asked, hypotheses proposed, experiments conducted, but time is limited, so we will focus here on those directions which we believe are the most fruitful:

1. Interaction between invariant object recognition and pose estimation. For the case when pose estimation is performed in reference to a static object suddenly impinging on the retina, object recognition is hypothesized to be a critical process interacting with the pose estimation process. A simplified

view might break down the process into the following steps: 1) perform invariant object recognition, 2) activate/generate a canonical representation to be compared with an abstract representation of the external object<sup>12</sup>, 3) find correspondences and estimate the pose.<sup>13</sup> Currently, the process through which the stored and the external patterns are activated/generated and “placed side by side” is somewhat of a mystery and deserves serious consideration.

2. Invariant local features. Clearly, the quality of the local invariant feature being used can influence the accuracy of the resulting pose estimation. It should be useful to explore more complex hierarchical feature architectures, and investigate how the different levels may interact with correspondence detection and pose estimation. It should also be worthwhile to investigate more efficient feature coding strategies. For example, how might correspondence detection be implemented among colour features represented by broadly tuned neuron triplets (e.g. red, green and blue wavelength neurons). These considerations prompt the following questions, among others: 1) can biological neural systems compute conjunctions in relation to graded potentials (e.g.  $0.4 \otimes 0.4 \rightarrow 1$  and  $0.4 \otimes 0.7 \rightarrow 0$ ) and 2) can dendritic branches efficiently compute conjunctions with more than two arguments (e.g.  $1 \otimes 1 \otimes 0 \rightarrow 0$ ).
3. Direct evidence. An obvious choice for future research involves the search for direct evidence for/against correspondence distributions (for pose estimation or other functions) in biological neural systems. New neurohistological techniques, such as multicolored labeling, seem to offer the best chances of finding the necessary direct evidence. We suggest that the neurohistological search for correspondence distributions should proceed in three stages, based on the relative locations of the maps involved and presented here in increasing order of difficulty: 1) inter-modal distributions (e.g. between visual and auditory maps), 2) intra-modal but inter-functional distributions (e.g. between visual

---

<sup>12</sup>The process of invariant recognition is likely to produce an abstract/simplified description of the external object, which is compatible with the stored representation. In other words, the recognition process intrinsically simplifies/normalizes the representation of the perceived object so that correspondences can be found.

<sup>13</sup>These steps are likely to be more parallel and interactive than here implied.

form and visual depth maps) and 3) intra-modal and intra-functional distributions (e.g. between different visual form maps). Cognitive Neuroscience experiments and Comparative Neuroscience research should also provide a better understanding of the pose estimation abilities/limitations of different species.

4. Structural optimization. The structural optimization experiments in Chapter 7 all shared the common goal of minimizing the total wiring length. Different goals and new constraints should result in different structures, some of which might concur more with biological observations. Research into new goals and constraints is useful, not only from the perspective of the resulting architectures, but also because biological neural systems might be using other structural optimization goals apart from wiring length minimization. It should also be interesting to investigate a reverse-optimization procedure, by which an architectural configuration is first given (e.g. orientation pinwheels), and then inter-map connection patterns (involving two or more maps) are searched for those that can lead to (by structural optimization) the architecture.
5. Developmental algorithms. Another area which deserves further research involves developmental algorithms which simultaneously satisfy structural optimization goals. Ideally, developmental algorithms should be capable of generating architectures which not only estimate poses accurately, but also use efficient codes and structurally optimal configurations. Regarding pure developmental algorithms, it should also be interesting to experiment with different degrees of experiential input on one side, and genetic influence on the other. In order to more easily experiment with coding-efficiency issues (e.g. populations of broadly-tuned transformation voting nodes), it should be useful to relax the constraint regarding the number of correspondence detectors available (by allowing each neuron to represent every possible correspondence) and adopt some of the learning principles used in artificial self-organizing maps (see Kohonen 1982).

6. Temporal factors. A crucial element, deliberately omitted from this thesis, pertains to dynamics. It should be interesting to investigate how correspondence distribution architectures behave, when embodying temporal factors such as spike timings, refractory periods, differential firing rates, and others.
7. Generalizations. Finally, it should be very advantageous to explore several generalizations of the approach, i.e: the various modalities and applications of correspondence distributions. One such example would be the investigation of the development of “shape” distribution architectures in the auditory domain.

## 10.8 Conclusion

Our original objectives outlined in Chapter 1 can be encapsulated in a single goal: to find a biologically plausible neural solution for pose estimation. Throughout the thesis’ chapters, we have gradually built an argument supporting the fact that correspondence distributions satisfy this goal. We showed that the approach was accurate and robust. Simple neural architectures were demonstrated that could embody the approach. These architectures were shown to satisfy several biological constraints, and were structurally optimized revealing certain design principles commonly observed in biological neural systems. Recent biological findings supporting the computation of conjunctions at the level of dendritic branches lent further support to the argument. Furthermore, it was shown that a developmental model, based on simple local rules and visual experience, was sufficient for generating the architectures in an unsupervised manner. All these threads of consistency were woven, by gradually traversing four levels of abstraction: computational, algorithmic, implementational and formational. Based on these threads, we conclude that, pose estimation neural architectures based on distributions of correspondences are biologically plausible and thus warrant further investigation.

It is our hope that direct evidence of correspondence distributions will eventually be uncovered, confirming the thesis’ hypothesis, and possibly revealing unexpected variations of the approach. We hope that the ubiquitousness and generality (or

applicability) of conjunctions, in the context of inter and intra-map computations, will be confirmed. These discoveries might shed further light on the complexity of computations occurring within dendritic trees and other neural components (e.g. axonic structures). We also hope that by expanding our knowledge on how biological systems solve visual problems such as pose estimation, we will be in a better position to contribute further to the artificial domain thus, for example, allowing better robots and automatic video surveillance systems to be built.

University of Malaya



# Bibliography

- Allman, J. (1999), *Evolving Brains*, W.H. Freeman & Company.
- Anderson, C., Van Essen, D. & Olshausen, B. (2004), *Encyclopedia of Attention*, MIT Press, chapter Directed visual attention and the dynamic control of information flow.
- Arathorn, D. (2002), *Map-Seeking Circuits in Visual Cognition. A Computational Mechanism for Biological and Machine Vision*, Stanford University Press.
- Arathorn, D. (2004), Computation in the higher visual cortices: Map-seeking circuit theory and application to machine vision, *in* 'Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop'.
- Baird, H. (1985), *Model-Based Image Matching Using Location*, MIT Press, Cambridge, MA.
- Ballard, D. (1981), 'Generalizing the hough transform to detect arbitrary shapes', *Pattern Recognition* **13**(2), 111–122.
- Barlow, H., Blakemore, C. & Pettigrew, J. (1967), 'The neural mechanism of binocular depth discrimination', *Journal of Physiology* **193**, 327342.
- Basak, J. & Pal, S. (1999), 'Hough transform network', *Electronics Letters - IEE*.
- Biederman, I. (1985), 'Human image understanding: Recent research and a theory', *Computer Vision, Graphics, Image Proc.* **32**(1), 2973.
- Binzegger, T., Douglas, R. & Martin, K. (2004), 'A quantitative map of the circuit of cat primary visual cortex', *The Journal of Neuroscience* **24**(39), 8441–8453.
- Bishop, C. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press.
- Breuel, T. (1992a), Fast recognition using adaptive subdivisions of transformation space, *in* 'Proceedings. 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.92CH3168-2)', pp. 445–51.
- Breuel, T. (1992b), *Geometric Aspects of Visual Object Recognition*, PhD thesis, Massachusetts Institute of Technology.
- Brown, M., Keynes, R. & Lumsden, A. (2001), *The Developing Brain*, Oxford University Press.
- Brunelli, R. & Mich, O. (1999), On the use of histograms for image retrieval, *in* 'International Conference on Multimedia Computing and Systems'.

- Buxhoeveden, D. & Casanova, M. (2002), 'The minicolumn hypothesis in neuroscience', *Brain* **125**, 935–951.
- Campbell, R. & King, A. (2004), 'Auditory neuroscience: A time for coincidence?', *Current Biology* **14**, 886–888.
- Carpenter, G., Grossberg, S. & Lesher, G. (1998), 'The what-and-where filter: A spatial mapping neural network for object recognition and image understanding', *Computer Vision and Image Understanding* **69**(1), 1–22.
- Cass, T. (1993), Polynomial-time geometric matching for object recognition, PhD thesis, Massachusetts Institute of Technology.
- Chelazzi, L. (1998), 'Serial attention mechanisms in visual search: A critical look at the evidence', *Psychological Research* **62**, 195–219.
- Chelazzi, L., Miller, E., Duncan, J. & Desimone, R. (1993), 'A neural basis for visual search in inferior temporal cortex', *Nature (London)* **363**, 345–347.
- Cheung, K., Yeung, D. & Chin, R. (2002), 'On deformable models for visual pattern recognition', *Pattern Recognition* **35**, 1507–1526.
- Chklovskii, D., Mel, B. & Svoboda, K. (2004), 'Cortical rewiring and information storage', *Nature* **431**, 782–788.
- Chklovskii, D., Schikorski, T. & Stevens, C. (2002), 'Wiring optimization in cortical circuits', *Neuron* **34**, 341–347.
- Churchland, P. & Sejnowski, T. (1992), *The Computational Brain*, MIT Press.
- Cohen-Cory, S. (2002), 'The developing synapse: Construction and modulation of synaptic structures and circuits', *Science* **298**, 770–776.
- Contreras, D. (2004), 'Electrophysiological classes of neocortical neurons', *Neural Networks* **17**, 633–46.
- Coss, R. & Perkel, D. (1985), 'The function of dendritic spines: a review of theoretical issues', *Behavioral and Neural Biology* **44**(2), 151–185.
- De Haan, M. & Johnson, M. (2003), *Encyclopedia of Cognitive Science*, Macmillan Publishers, chapter Neuropsychological development, pp. 347–353.
- DeFelipe, J., Marco, P., Busturia, I. & Merchán-Prez, A. (1999), 'Estimation of the number of synapses in the cerebral cortex: Methodological considerations', *Cerebral Cortex* **9**, 722–732.
- Durbin, R. & Willshaw, D. (1987), 'An analogue approach to the travelling salesman problem using an elastic net method', *Nature* **326**, 689–691.
- Egmont-Peterson, M., de Ridder, D. & Handels, H. (2002), 'Image processing with neural networks – a review', *Pattern Recognition* **35**(10), 2279–2301.
- Espinosa, C. & Perkowski, M. (1991), Hierarchical hough transform based on pyramidal architecture, in 'Northcon '91'.

- Fischler, A. & Bolles, R. (1981), ‘Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography’, *Commun. ACM* **24**(6), 381–395.
- Földiák, P. (1991), ‘Learning invariance from transformation sequences’, *Neural Computation* **3**(2), 194–200.
- Fruchterman, T. & Reingold, E. (1991), ‘Graph drawing by force-directed placement’, *Software – Practice and Experience* **21**(11), 1129–1164.
- Gan, W., Grutzendler, J., Wong, W., Wong, R. & Lichtman, J. (2000), ‘Multi-color “diolistic” labeling of the nervous system using lipophilic dye combinations’, *Neuron* **27**, 219–225.
- Gazzaniga, M., Ivry, R. & Mangun, G. (2002), *Cognitive Neuroscience, Second Edition*, W. W. Norton & Company.
- Gierer, A. & Müller, C. (1995), ‘Development of layers, maps and modules’, *Current Opinion in Neurobiology* **5**, 91–97.
- Giles, C. & Maxwell, T. (1987), ‘Learning, invariance, and generalization in high-order neural networks’, *Applied Optics* **26**(23), 4972–4978.
- Gold, S., Rangarajan, A., Lu, C., Pappu, S. & Mjolsness, E. (1998), ‘New algorithms for 2d and 3d point matching: Pose estimation and correspondence’, *Pattern Recognition* **31**(8), 1019–1031.
- Goodhill, G. & Carreira-Perpinan, M. (2002), *Encyclopedia Of Cognitive Science*, Macmillan, chapter Cortical Columns, pp. 845–851.
- Goodhill, G. & Willshaw, D. (1990), ‘Application of the elastic net algorithm to the formation of ocular dominance’, *Network: Computation in Neural Systems* **1**(1), 41–59.
- Goodhill, G., Finch, S. & Sejnowski, T. (1995), Optimizing cortical mappings, in ‘NIPS’.
- Grutzendler, J., Tsai, J. & Gan, W. (2003), ‘Rapid labeling of neuronal populations by ballistic delivery of fluorescent dyes’, *Methods* **30**(1), 79–85.
- Hansen, T. & Neumann, H. (2004), ‘Neural mechanisms for the robust representation of junctions’, *Neural Computation* **16**(5), 1013 – 1037.
- Hartley, R. & Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, second edn, Cambridge University Press.
- Haykin, S. (1999), *Neural Networks: a Comprehensive Foundation*, Prentice Hall International.
- Hebb, D. (1949), *The organization of behavior: A neuropsychological theory*, New York: Wiley.
- Hecker, Y. & Bolle, R. (1994), ‘On geometric hashing and the generalized hough transform’, *IEEE Transactions on Systems, Man, and Cybernetics* **24**(9), 1328–1338.

- Hevner, R., Daza, R., Rubenstein, J., Stunnenberg, H., Olavarria, J. & Englund, C. (2003), 'Beyond laminar fate: Toward a molecular classification of cortical projection/pyramidal neurons', *Developmental Neuroscience* **25**, 139151.
- Hollard, V. & Delius, J. (1982), 'Rotational invariance in visual pattern recognition by pigeons and humans', *Science* **218**, 804–806.
- Hong, J. & Tan, X. (1988), A new approach to point pattern matching, in 'Proc. 9th International Conference on Pattern Recognition'.
- Hopf, F., Waters, J., Mehta, S. & Smith, S. (2002), 'Stability and plasticity of developing synapses in hippocampal neuronal cultures', *The Journal of Neuroscience* **22**(3), 775781.
- Hough, P. (1959), Machine analysis of bubble chamber pictures, in 'Proc. International Conference on High Energy Accelerators and Instrumentation'.
- Hough, P. (1962), A method and means for recognizing complex patterns, Technical report, US Patent 3,069,654.
- Hua, J., Smear, M., Baier, H. & Smith, S. (2005), 'Regulation of axon growth in vivo by activity-based competition', *Nature* **434**, 1022–1026.
- Huang, J., Chen, C., Wang, W. & Lee, J. (1995), 'A general mean-based iterative winner-take-all neural network', *IEEE Transactions on Neural Networks* **6**(1), 14–24.
- Hubel, D. & Wiesel, T. (1968), 'Receptive fields and functional architecture of monkey striate cortex', *J. Physiol. (Lond.)* **195**, 215–243.
- Illingworth, J. & Kittler, J. (1987), 'The adaptive hough transform', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9**(5), 690 – 698.
- Jain, A., Y., Z. & Dubuisson-Jolly, M. (1998), 'Deformable template models: A review', *Signal Processing* **71**, 109129.
- Jeffress, L. (1948), 'A place theory of sound localization', *J. Comp. Physiol. Psychol.* **41**, 35–39.
- Johansson, C. & Lansner, A. (2004), Towards cortex sized artificial nervous systems, in 'Knowledge-Based Intelligent Information & Engineering Systems'.
- Jurie, F. & Schmid, C. (2004), Scale-invariant shape features for recognition of object categories, in 'International Conference on Computer Vision & Pattern Recognition'.
- Kalinichenko, S. & Okhotin, V. (2005), 'Unipolar brush cells a new type of excitatory interneuron in the cerebellar cortex and cochlear nuclei of the brainstem', *Neuroscience and Behavioral Physiology* **35**(1), 21–36.
- Kandel, E., Schwartz, J. & Jessell, T. (2000), *Principles of Neural Science*, McGraw-Hill Medical.
- Kohler, C., Hoffmann, K., Dehnhardt, G. & Mauck, B. (2005), 'Mental rotation and rotational invariance in the rhesus monkey (macaca mulatta)', *Brain, Behaviour and Evolution* **66**(3), 158–166.

- Kohonen, T. (1982), 'Self-organized formation of topologically correct feature maps', *Biological Cybernetics* **43**, 59–69.
- Koutroumbas, K. (2004), 'Recurrent algorithms for selecting the maximum input', *Neural Processing Letters* **20**(3), 179–197.
- Lamdan, Y. & Wolfson, H. (1988), Geometric hashing: A general and efficient model-based recognition scheme, *in* 'Proc. International Conference of Computer Vision'.
- Levitt, J., Kiper, D. & Movshon, J. (1994), 'Receptive fields and functional architecture of macaque v2', *Journal of Neurophysiology* **71**, 2517–2542.
- Li, H., Lavin, M. & Master, R. (1986), 'Fast hough transform: A hierarchical approach', *Computer Vision, Graphics, and Image Processing* **36**(2-3), 139–161.
- London, M. & Häusser, M. (2005), 'Dendritic computation', *Annu. Rev. Neurosci.* **28**, 503–532.
- Lueschow, A., Miller, E. & Desimone, R. (1994), 'Inferior temporal mechanisms for invariant object recognition', *Cerebral Cortex* **4**, 523–531.
- Lynch, G. & Baudry, M. (1984), 'The biochemistry of memory: A new and specific hypothesis', *Science* **224**, 1057–1063.
- Majani, E., Erlanson, R. & Abu-Mostafa, Y. (1989), *On the K-winners-take-all-network*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Marr, D. (1982), *Vision: A computational investigation into the human representation and processing of visual information*, San Francisco: W.H. Freeman.
- Mastronarde, D. (1987), 'Two classes of single-input x-cells in cat lateral geniculate nucleus. i. receptive-field properties and classification of cells', *Journal of Neurophysiology* **57**(2), 357–380.
- McCulloch, W. & Pitts, W. (1943), 'A logical calculus of ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* **5**, 115–133.
- McInerney, T. & Terzopoulos, D. (1996), 'Deformable models in medical image analysis: A survey', *Medical Image Analysis* **1**(2), 91–108.
- Mikami, A., Newsome, W. & Wurtz, R. (1986), 'Motion selectivity in macaque visual cortex. i. mechanisms of direction and speed selectivity in extrastriate area mt', *Journal of Neurophysiology* **55**, 1308–1327.
- Mikolajczyk, K. & Schmid, C. (2004), 'Scale and affine invariant interest point detectors', *International Journal of Computer Vision* **60**(1), 63–86.
- Miller, K. (1994), 'Models of activity-dependent neural development', *Prog. Brain Res.* **102**, 303–18.
- Mountcastle, V. (1997), 'The columnar organization of the neocortex', *Brain* **120**, 701–722.
- Müller, H. & Krummenacher, J. (2006), 'Visual search and selective attention', *Visual Cognition* **14**(4-8), 389–410.

- Nene, S., Nayar, S. & Murase, H. (1996), Columbia object image library: Coil-100. technical report cucs-006-96, Technical report, Department of Computer Science, Columbia University.
- Olshausen, B. & Field, D. (2005), ‘How close are we to understanding v1?’, *Neural Computation* **17**, 1665–1699.
- Olshausen, B., Anderson, C. & Van Essen, D. (1993), ‘A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information’, *The Journal of Neuroscience* **13**(11), 4700–4719.
- Olson, C. (2001), ‘A general method for geometric feature matching and model extraction’, *International Journal of Computer Vision* **45**(1), 39–53.
- Osada, R., Funkhouser, T., Chazelle, B. & Dobkin, D. (2002), ‘Shape distributions’, *ACM Transactions on Graphics* **21**(4), 807–832.
- Palmer, S. (1999), *Vision Science, Photons to Phenomenology*, MIT Press.
- Pasupathy, A. & Connor, C. (2001), ‘Shape representation in area v4: Position specific tuning for boundary conformation’, *J. Neurophysiol.* **86**(5), 2505–2519.
- Poirazi, P. & Mel, B. (2001), ‘Impact of active dendrites and structural plasticity on the memory capacity of neural tissue’, *Neuron* **29**, 779–796.
- Poirazi, P., Brannon, T. & Mel, B. (2003), ‘Pyramidal neuron as two-layer neural network’, *Neuron* **37**, 989–999.
- Polsky, A., Mel, B. & Schiller, J. (2004), ‘Computational subunits in thin dendrites of pyramidal cells’, *Nature Neuroscience*.
- Rao, R. & Ballard, D. (1997), Localized receptive fields may mediate transformation-invariant recognition in the visual cortex, Technical Report 97.2, National Resource Laboratory for the Study of Brain and Behavior, Department of Computer Science, University of Rochester.
- Rolls, E. & Deco, G. (2001), *Computational Neuroscience of Vision*, Oxford University Press.
- Schmid, C., Dorko, G., Lazebnik, S., Mikolajczyk, K. & Ponce, J. (2005), *Handbook of Pattern Recognition and Computer Vision, 3rd edition*, World Scientific Publishing, chapter Pattern Recognition with Local Invariant Features.
- Schulz, R. & Reggia, J. (2005), ‘Mirror symmetric topographic maps can arise from activity-dependent synaptic changes’, *Neural Computation* **17**, 1059–1083.
- Scott, E., Reuter, J. & Luo, L. (2003), ‘Dendritic development of drosophila high order visual system neurons is independent of sensory experience’, *BMC Neuroscience*.
- Sebe, N., Tian, Q., Loupas, E., Lew, M. & Huang, T. (2003), ‘Evaluation of salient point techniques’, *Image and Vision Computing* **21**(13-14), 1087–1095.
- Shepherd, G. (2003), *The Synaptic Organization of the Brain*, Oxford University Press.

- Silva, G. (2006), 'Neuroscience nanotechnology: Progress, opportunities, and challenges', *Nature Reviews Neuroscience* **7**, 65–74.
- Swiniarski, R. & Skowron, A. (2003), 'Rough set methods in feature selection and recognition', *Pattern Recognition Letters* **24**(6), 833–849.
- Tian, T. & Shah, M. (1997), 'Recovering 3d motion of multiple objects using adaptive hough transform', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(10), 1178–1183.
- Troyer, T., Krukowski, A., Priebe, N. & Miller, K. (1998), 'Contrast-invariant orientation tuning in cat visual cortex: Thalamocortical input tuning and correlation-based intracortical connectivity', *The Journal of Neuroscience* **18**(15), 5908–5927.
- Tuytelaars, T. and Van Gool, L. (1999), Content-based image retrieval based on local affinity invariant regions., in 'Third International Conference on Visual Information Systems', pp. 493–500.
- Ullman, S. (1996), *High-level Vision: Object Recognition and Visual Cognition*, MIT Press.
- Ullman, S. & Koch, C. (1999), *Encyclopedia of Neuroscience (2nd Ed.)*, Elsevier Science, chapter Selective visual attention, pp. 149–151.
- Ungerleider, L. & Haxby, J. (1994), 'What and where in the human brain', *Current Opinion in Neurobiology* **4**, 157–165.
- Ungerleider, L. & Mishkin, M. (1982), *Analysis of visual behavior*, Cambridge, MA: MIT Press, chapter Two cortical visual systems, pp. 549–586.
- Van Essen, D. & Anderson, C. (1990), *An introduction to neural and electronic networks*, Academic Press Professional, Inc., chapter Information processing strategies and pathways in the primate retina and visual cortex, pp. 43–72.
- Wandell, B. (1995), *Foundations of Vision*, Sunderland, Massachusetts: Sinauer Associates, Inc.
- Wong, R. (1999), 'Retinal waves and visual system development', *Annual Review of Neuroscience* **22**, 29–47.
- Wood, J. (1996), 'Invariant pattern recognition: a review', *Pattern Recognition*, **29**(1), 1–17.
- Yena, E. & Smith, A. (2005), 'Image recognition via deformable templates', *Statistical Methodology* **2**, 213–225.
- Yu, H., Farley, B., Jin, D. & Sur, M. (2005), 'The coordinated mapping of visual space and response features in visual cortex', *Neuron* **47**, 267–280.
- Yuste, R., Majewska, A., Cash, S. & Denk, W. (1999), 'Mechanisms of calcium influx into hippocampal spines: Heterogeneity among spines, coincidence detection by nmda receptors, and optical quantal analysis', *The Journal of Neuroscience* **19**(6), 1976–1987.
- Zeki, S. (1993), *A Vision of the Brain*, Blackwell Scientific Publications.