

# Model Learning in Iconic Vision

**Herman Martins Gomes**



Ph.D.  
University of Edinburgh  
2002

# *Abstract*

Generally, object recognition research falls into three main categories: (a) geometric, symbolic or structure based recognition, which is usually associated with CAD-based vision and 3-D object recognition; (b) property, vector or feature based recognition, involving techniques that vary from specific feature vectors, multiple filtering to global descriptors for shape, texture and colour; and (c) iconic or image based recognition, which either complies with the traditional sensor architecture of an uniform array of sampling units, or uses alternative representations. An example is the log-polar image, which is inspired by the human visual system and besides requiring less pixels, has some useful mathematical properties. The context of this thesis is a combination of the above categories in the sense that it investigates the area of iconic based recognition using image features and geometric relationships. It expands an existing vision system that operates by fixating at interesting regions in a scene, extracting a number of raw primal sketch features from a log-polar image and matching new regions to previously seen ones.

Primal sketch features like *edges*, *bars*, *blobs* and *ends* are believed to take part of early visual processes in humans providing cues for an attention mechanism and more compact representations for the image data. In an earlier work, logic operators were defined to extract these features, but the results were not satisfactory. **This thesis initially investigates the question of whether or not primal sketch features could be learned from log-polar images, and gives an affirmative answer.** The feature extraction process was implemented using a neural network which learns examples of features in a window of receptive fields of the log-polar image. An architecture designed to encode the feature's class, position, orientation and contrast has been proposed and tested. Success depended on the incorporation of a function that normalises the feature's orientation and a PCA pre-processing module to produce better separation in the feature space. A strategy that combines synthetic and real features is used for the learning process.

**This thesis also provides an answer to the important, but so far not well explored, question of how to learn relationships from sets of iconic object models obtained from a set of images.** An iconic model is defined as a set of regions, or object instances, that are similar to each other, organised into a geometric model specified by the relative scales, orientations, positions and similarity scores for each pair of image regions. Similarities are measured with a cross-correlation metric and relative scales and orientations are obtained from the best matched translational variants generated in the log-polar space. A solution to the structure learning problem is presented in terms of a graph based representation and algorithm. Vertices represent instances of an image neighbourhood found in the scenes. An edge in the graph represents a relationship between two neighbourhoods. Intra and inter model relationships are inferred by means of the cliques found in the graph, which leads to rigid geometric models inferred from the image evidence.

To my wife Patricia, with profound love.

# *Acknowledgements*

I would like to start thanking my supervisor, Bob Fisher, for his support, patience and wise supervision during the preparation of this thesis. I owe him my profound admiration and respect. I would like also to acknowledge John Hallam for his guidance, collaboration and encouragement during the initial stages of this work.

I am also thankful to my thesis examiners, Chris Williams and James Crowley, for their relevant philosophical questions, technical comments and suggestions for improving this thesis.

Many thanks to all my colleagues in the Vision Lab, specially Craig Robertson and Anthony Ashbrook, with whom I interacted most and had inspiring discussions not only about my work but also about many other interesting research problems, and Helmut Cantzler, for his friendship in helping with the thesis submission.

I would like to acknowledge CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil) for providing the financial support for this thesis and for their continuous contribution to the development of science, technology and higher education in Brazil.

I would like also to acknowledge the efforts of my colleagues back home at DSC/UFCG (Departamento de Sistemas e Computação, Universidade Federal da Paraíba, Brazil), specially Bernardo Lula Junior, who always gave me encouragement to seek this PhD.

My everlasting gratitude to my wife Patricia for her love and support. I would like to conclude by thanking my parents, Cícero and Neuza, and my brothers, who have patiently waited for this work to be concluded, and my parents-in-law, Adelson and Perácia, for all their support.

# *Declaration*

I hereby declare that I composed this thesis entirely myself and that it describes my own research, except where explicitly stated otherwise in the text. Part of the material in Chapters 3, 4 and 5 has already been published in [GFH98], [GF01] and [GF00].

Herman Martins Gomes  
Edinburgh  
May 2nd, 2002

# *Contents*

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Declaration</b>	<b>v</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Algorithms</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Learning, Vision and Techniques . . . . .	2
1.2 What is this Thesis About . . . . .	3
1.3 Overview of the Thesis . . . . .	5
<b>2 Literature Review</b>	<b>8</b>
2.1 Iconic Vision . . . . .	9
2.1.1 Template Matching . . . . .	9
2.1.2 Log-Polar Representation . . . . .	10
2.1.3 Complex-Log Representation . . . . .	11
2.1.4 Log-Log Representation . . . . .	12
2.2 Property Based Approaches . . . . .	13
2.2.1 Multiple Filtering . . . . .	14
2.2.2 Colour Histograming . . . . .	16

2.2.3	Multidimensional Receptive Field Histograms . . . . .	16
2.3	Symbolic or Geometric Approaches . . . . .	17
2.3.1	IMAGINE System . . . . .	17
2.3.2	Projective Invariants . . . . .	19
2.3.3	Comparing Geometric Invariants and Appearance-Based Systems . .	21
2.4	Learning . . . . .	22
2.4.1	Learning Low-Level Features . . . . .	23
2.4.2	Learning Geometric Descriptions . . . . .	23
2.5	Vision Frameworks . . . . .	25
2.5.1	From Primal Sketch to 3-D Models . . . . .	25
2.5.2	Four Frames . . . . .	27
2.6	A Practical Iconic Vision System . . . . .	28
2.6.1	The Initial System . . . . .	28
2.6.2	Some Criticism . . . . .	31
2.6.3	Relation with the Theoretical Systems . . . . .	32
2.6.4	Adding Parallelism . . . . .	32
2.6.5	Improving Parallelism and Adding Full Subcomponent Evidence . . .	33
2.7	Summary . . . . .	33
<b>3</b>	<b>Retina-Like Image Representation</b>	<b>36</b>
3.1	Motivations . . . . .	37
3.2	Structural Design . . . . .	38
3.2.1	Fovea . . . . .	38
3.2.2	Outside the Fovea . . . . .	43
3.2.3	The Entire Retina . . . . .	52
3.2.4	Log-Polar Representation . . . . .	54
3.3	Functionality . . . . .	54
3.3.1	Receptive Field Function . . . . .	55
3.3.2	Providing Support for Estimating the Reflectance Information . . . .	55
3.3.3	Mapping from Cartesian to Retinal Images (Foveation) . . . . .	57
3.3.4	Mapping from Retinal to Cartesian Images (Defoveation) . . . . .	58

3.4	Retina Parameters and Example . . . . .	59
3.5	Summary . . . . .	60
<b>4</b>	<b>Primal Sketch Feature Extraction</b>	<b>64</b>
4.1	Motivations . . . . .	65
4.2	A Previous Approach . . . . .	65
4.3	Proposed Approach . . . . .	66
4.3.1	Feature Detector . . . . .	67
4.3.2	Normalising the Feature Orientation . . . . .	68
4.3.3	Principal Component Analysis (PCA) . . . . .	71
4.3.4	Neural Network Architecture . . . . .	74
4.3.5	Coding the Contrast Information . . . . .	77
4.3.6	Classification Rule . . . . .	79
4.4	Training and Evaluation . . . . .	80
4.4.1	Synthetic Training Data . . . . .	80
4.4.2	Initial Training . . . . .	83
4.4.3	Evaluating the Initially Trained System's Performance . . . . .	84
4.4.4	Adding Features from Real Images . . . . .	90
4.4.5	Experimenting on Synthetic and Real Images . . . . .	91
4.5	Concluding Remarks . . . . .	94
<b>5</b>	<b>Learning Iconic Models</b>	<b>97</b>
5.1	Learning Primitive Object Feature Models . . . . .	98
5.1.1	System's Architecture . . . . .	98
5.1.2	Model Representation . . . . .	100
5.1.3	Similarity Function . . . . .	101
5.1.4	Clustering Algorithm . . . . .	103
5.2	Learning Structured Models via a Representational Graph . . . . .	106
5.2.1	Vertices . . . . .	107
5.2.2	Edges . . . . .	110
5.2.3	Cliques . . . . .	116



5.3	Exploratory Experiments . . . . .	119
5.3.1	Misplacement of the Fixation Points . . . . .	119
5.3.2	Experiments with Synthetic Scenes . . . . .	121
5.4	Structure Learning Case Studies . . . . .	140
5.4.1	Telephone Unit . . . . .	140
5.4.2	PDA and CD Player . . . . .	143
5.4.3	Limitation of the Object Matching . . . . .	147
5.4.4	Doll . . . . .	150
5.5	Summary . . . . .	153
<b>6</b>	<b>Conclusion and Further Work</b>	<b>156</b>
6.1	Summary of the Thesis and Contributions . . . . .	157
6.1.1	Log-Polar Image Representation . . . . .	157
6.1.2	Primal Sketch Feature Extraction . . . . .	157
6.1.3	Model Learning . . . . .	159
6.2	Future Work . . . . .	160
6.2.1	Primal Sketch Feature Extraction . . . . .	160
6.2.2	Model Learning . . . . .	161
	<b>Bibliography</b>	<b>163</b>
	<b>A Primal Sketch Feature Extraction Examples</b>	<b>172</b>
	<b>B Evaluating the Similarity Function</b>	<b>180</b>
	<b>C Graphs from the Combinatorics Experiments</b>	<b>186</b>

# *List of Figures*

2.1	Graphical result of the recognition of two objects (Rothwell et al) . . . . .	20
2.2	Three kinds of relation involving parts of objects as proposed by Winston . .	24
2.3	EXIT difference note by Winston . . . . .	25
2.4	Marr's representational framework. . . . .	26
2.5	Feldman's four frames model. . . . .	27
2.6	Grove and Fisher architecture. . . . .	29
2.7	The 3x3 mask used by the Grove and Fisher system to detect features. . . . .	30
3.1	Retina structure . . . . .	39
3.2	Polar coordinates in the fovea. . . . .	39
3.3	Hexagonal coordinate system in the fovea. . . . .	40
3.4	Accessing receptive fields outside the fovea . . . . .	47
3.5	Angular (concentric) overlap. . . . .	48
3.6	Radial overlap. . . . .	50
3.7	Mapping of a 5 ring retina containing a 3 ring fovea . . . . .	54
3.8	An example of applying the retinal representation to a real image . . . . .	61
4.1	The mask used in Grove and Fisher's system to detect features. . . . .	66
4.2	Overview of the main processes and data of the primal sketch feature extraction system . . . . .	69
4.3	Symmetry operators used to detect the feature orientations. . . . .	72
4.4	Final neural network architecture. . . . .	78
4.5	Target outputs for the testing sets. . . . .	85
4.6	Outputs from the trained edge neural module when applied to the testing sets.	86

4.7	Outputs from the remaining trained neural modules when applied to the testing sets . . . . .	87
4.8	Absolute errors of the symmetry operator for the testing sets of the oriented features. . . . .	88
4.9	Errors type I and II vs. the classification threshold for edges using the proposed approach. . . . .	88
4.10	Errors type I and II vs. the classification threshold for the remaining features using the proposed approach. . . . .	89
4.11	Errors type I and II vs. the classification threshold for edges using the previous approach. . . . .	91
4.12	Errors type I and II vs. the classification threshold for the remaining features using the previous approach. . . . .	92
5.1	System's main modules. . . . .	99
5.2	Processes (rounded boxes) and data (rectangular boxes) involved in building a relational graph from an iconic model database. . . . .	106
5.3	Relations between vertex components to form an edge . . . . .	110
5.4	Four hypothetical scenes used to synthesise an iconic model database. . . . .	122
5.5	Finding an edge threshold with no missing feature. . . . .	134
5.6	Finding an edge threshold with 2 missing features. . . . .	135
5.7	Scene images used for the telephone case study . . . . .	141
5.8	Scene images used for the 2 <sup>nd</sup> case study. . . . .	145
5.9	Images used to test the matching scores of (a) central, $a^{i,1}$ , and (b) peripheral, $a^{i,2}$ , interest points. . . . .	149
5.10	Scenes with the doll object for structure learning. . . . .	151
A.1	<i>Edge</i> classifiers on synthetic images. . . . .	173
A.2	<i>-Bar</i> classifiers on synthetic images. . . . .	174
A.3	<i>+Bar</i> classifiers on synthetic images. . . . .	174
A.4	$\pm$ <i>Blob</i> classifiers, respectively, on synthetic images. . . . .	175
A.5	$\pm$ <i>End</i> classifiers, respectively, on synthetic images. . . . .	175
A.6	Test image 1: final vs. previous approach results. . . . .	176
A.7	Test image 1: results before and after the addition of real features. . . . .	177
A.8	Final results for test images 2 and 3. . . . .	178

A.9	Final results for test images 4 and 5. . . . .	179
B.1	Similarity experiment with <i>Tool1</i> . . . . .	181
B.2	Similarity experiment with <i>Tool2</i> . . . . .	182
B.3	Similarity experiment with <i>Pda1</i> . . . . .	183
B.4	Similarity experiment with <i>Pda2</i> . . . . .	184
B.5	Similarity experiment with <i>Party</i> . . . . .	185
C.1	Average total CPU time spent by the entire structure learning approach versus the number of scenes. . . . .	187
C.2	Average natural logarithm of the total CPU time spent by the entire structure learning approach versus the number of scenes. . . . .	188
C.3	Average number of iterations by the clique finding algorithm versus the number of scenes. . . . .	189
C.4	Average number of vertices versus the number of scenes. . . . .	190
C.5	Average number of edges versus the number of scenes. . . . .	191
C.6	Average number of pruned edges versus the number of scenes. . . . .	192
C.7	Average number of pruned vertices (after edge pruning) versus the number of scenes. . . . .	193
C.8	Average number of (raw) cliques found versus the number of scenes. . . . .	194
C.9	Average number of maximal cliques found versus the number of scenes. . . . .	195
C.10	(a) Fitting an exponential function to the original data; (b) extrapolating this function to 30 images. . . . .	196

# *List of Tables*

3.1	Fovea parameters . . . . .	43
3.2	Out-fovea parameters. . . . .	52
3.3	Unified retinal parameters. . . . .	53
4.1	Pictorial representation of the first 14 <i>eigenvectors</i> and <i>eigenvalues</i> . . . . .	75
4.2	Chosen subsets of principal components from Table 4.1. . . . .	76
4.3	Parameters used in designing feature examples for the training sets. . . . .	82
4.4	Some examples of the training features . . . . .	82
4.5	Training parameters . . . . .	83
4.6	New training patterns manually selected from image 1 . . . . .	94
5.1	Synthetic model database derived from Figure 5.4. . . . .	123
5.2	Cliques found in the first experiment on synthetic scenes. . . . .	123
5.3	Summary of first experiment on synthetic scenes. . . . .	123
5.4	Synthetic model database derived from Figure 5.4 after removing features $a^{1,1}$ , $a^{2,1}$ and $b^{2,1}$ . . . . .	124
5.5	Summary of the second experiment on synthetic scenes. . . . .	124
5.6	Vertices found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex. . . . .	125
5.7	Edges found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex. . . . .	125
5.8	Edges after equivalency pruning in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex. . . . .	126
5.9	Cliques found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex. . . . .	126

5.10	Vertices found in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex. . . . .	127
5.11	Edges found in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex. . . . .	128
5.12	Edges after equivalency pruning in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex. . . . .	128
5.13	Vertices kept after edge pruning in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex. . . . .	129
5.14	Cliques found in the second experiment on synthetic scenes when allowing no more than two wild-card feature per vertex. . . . .	129
5.15	Maximal cliques from the second experiment on synthetic scenes when allowing no more than two wild-card feature per vertex. . . . .	129
5.16	Results of one application of $\mathcal{P}(3,2,30,0.5)$ to Table 5.1. . . . .	131
5.17	Edges created from perturbed model base in Table 5.16 when using a null edge threshold. . . . .	132
5.18	Vertices created from perturbed model base in Table 5.16 when using a null edge threshold. . . . .	132
5.19	Maximal cliques created from model base in Table 5.16 when using a null edge threshold. . . . .	133
5.20	Results of the Algorithm 5.1 on the telephone scenes. . . . .	142
5.21	Maximal cliques found for the telephone scenes when adding no noise to the models. . . . .	143
5.22	Results of perturbing Table 5.20 with noise parameters $\mathcal{P}(3,2,30,0.5)$ . . . . .	144
5.23	Maximal cliques found for the telephone scenes when using a noise modified model database. . . . .	144
5.24	Model database for the 2 <sup>nd</sup> experiment involving a PDA and a CD player. . . . .	146
5.25	Results of perturbing Table 5.24 with noise parameters $\mathcal{P}(3,1,30,0.1)$ . . . . .	147
5.26	Maximal cliques found in the PDA / CD player experiment with features $b^{3,1}$ and $d^{1,1}$ missing. . . . .	147
5.27	Matching scores for (a) central, $a^{i,1}$ , and (b) peripheral, $b^{i,1}$ , interest points. . . . .	148
5.28	Model database from the doll experiment. . . . .	152
5.29	Results of perturbing Table5.28 with noise parameters $\mathcal{P}(3,2,30,0.5)$ . . . . .	152
5.30	Maximal cliques correctly found for the doll experiment with no missing features. . . . .	153

5.31 Maximal cliques correctly found for the doll experiment with missing features  
 $b^{2,1}$  and  $c^{3,1}$  . . . . . 153

## *List of Algorithms*

5.1	Clustering object features. . . . .	104
5.2	Pruning the set of edges $E$ . . . . .	115
5.3	Finding all cliques in a graph $G = (V, E)$ . . . . .	117
5.4	Extracting the maximal cliques. . . . .	118



# *Chapter 1*

## *Introduction*

A scientific model functions both as a concise descriptive device and as a means of predicting what will happen next in a given circumstance ... Visual models play an analogous role. Any reasonably sophisticated visual system is concerned with representing approximations to certain aspects of states of nature. The needs for these representations are all inherently related to being able to form appropriate concise descriptions and predictions about the properties of the domain.

*R. Watt [Wat91]*

### **Contents**

---

<b>1.1 Learning, Vision and Techniques</b> . . . . .	<b>2</b>
<b>1.2 What is this Thesis About</b> . . . . .	<b>3</b>
<b>1.3 Overview of the Thesis</b> . . . . .	<b>5</b>

---

This chapter starts with a brief general discussion about learning and vision, followed by a description of the general problems and classes of techniques normally involved when learning and recognising visual objects. Then it moves to an explanation of the context of the research and presents the scientific questions addressed by the thesis. Finally, an overview of the entire thesis structure is given.

## 1.1 Learning, Vision and Techniques

Learning can be defined as the process of gaining knowledge from the interaction with the environment. It is usually related to the capability of a system to acquire knowledge from examples and to generalise this knowledge in such a way that it can be applied to new and different circumstances. Learning is also associated with the ability to make inferences from incomplete information and to generate new knowledge from an existing one.

Machine Learning is an area that is receiving growing interest by Computer Vision researchers in the recent years. It can offer effective methods to a wide variety of Computer Vision problems ranging from the extraction of low level features to the acquisition and recognition of high level visual (2-D or 3-D) models and behaviours. There exists a fair number of well founded learning technologies available, including Neural Networks, Bayesian and Probabilistic Modelling, Principal Component Analysis, Support Vector Machines among others.

Within the context of visual learning, our ultimate goal is to design a vision system that is capable of automatically learning objects or parts of objects (object features) and their relationships from generic scenes. But this is not an easy task in a completely autonomous context, in which there is no one to define the appropriate training sets with its objects already segmented, normalised and separated into classes. In order to deliver objects in this way, a system has to somehow deduce the object's shape, position, scale and orientation in the scene.

Generally, object recognition research falls into three main categories: (1) geometric, symbolic, or structure based recognition; (2) property, vector or feature based recognition; and (3) iconic (image) based recognition. Most of the research found in the first category is related to 3-D object recognition systems and typically uses either volumetric relationships [Bie86] or surface relationships [Fis89]. Relational matching [Vos92] is a common technique used in this area to do the matching between two relational model descriptions. There is also a large number of well established 2-D based techniques (e.g. CAD based vision), which are mainly related to industrial applications.

The second category presents a wider range of techniques varying from the use of specific feature vectors and multiple filtering to global descriptors for shape, texture and colour. This kind of approach is popular among applications involving image database indexing [BFG96,

PPS96, GJ97]. Swain and Ballard [SB91] proposed matching a colour histogram calculated from a region of the image with a colour histogram from a sample image of the object. Rao and Ballard [RB95] used high dimensional feature vectors obtained from the application of multiple Gaussian derivative filters at a number of orientations and scales. Schiele and Crowley [SC96b, SC96a] presented an approach which can be classified as a combination of multiple filtering (e.g. [RB95]) with histogramming (e.g. [SB91]), in which histograms of local shape properties were calculated as a vector of linear local neighbourhood operators.

Finally, the third category is characterised by the direct use of images. In this case, the most popular technique to recognise objects is template matching [BAR94]. But, when using the traditional sensor architecture (uniform array of sampling units), the number of pixels involved can be too high to allow for a more elaborate computation. An alternative is to use a log-polar representation which is inspired by the human retina [Sch77, Wil83, Zek93] and requires less pixels to represent an image once it is space variant (high resolution in the centre and low resolution in the periphery).

Within this context, an iconic vision system based on primal sketch features extracted from a log-polar representation was developed [GF96]. Afterwards, a full subcomponent evidence mechanism [Mac97, FM98] was added to the system in order to improve the matching process. This is a hybrid technique in which iconic models are represented through the use of geometric relations, which are in turn used during recognition to strengthen the evidence that a particular object model has been found based on other nearby matches. But the training sets to build the models and the model relationships themselves had to be defined manually.

The work presented in this thesis fits in the category of iconic based recognition using geometric relationships and expands the work described in [Jen94, GF96, Mac97, FM98]. It also loosely relates to research in perceptual grouping and image segmentation, which involves finding relationships between pre-determined kinds of low-level features [FF95a].

## 1.2 What is this Thesis About

This thesis presents new approaches for solving two important problems in the area of iconic vision systems. The first problem is how to derive primal sketch descriptions from log-polar images more reliably and with a better level of description. This problem was motivated in

this thesis by the practical issue of producing an image representation that is more suitable for building image-based models and providing interest points for an attention mechanism. To tackle this problem, we developed a neural network architecture that functions by learning and extracting primal sketch features from log-polar images. This approach demonstrated a considerable improvement when compared to a previous one based on hand-crafted local operators.

The second problem is twofold: (i) how to use the above image representation for constructing a database of geometric iconic models from a set of views; and, most importantly, (ii) how to infer rigid body object relations by looking at this database.

To deal with subproblem (i), we designed and implemented a prototype algorithm that, by taking advantage of some properties of the log-polar map, not only clusters image regions that are similar to each other according to a cross-correlation metric, but also registers planar relative scales, orientations and similarity scores between pairs of objects belonging to a same class.

Subproblem (ii) is tackled by constructing a graph in which vertices represent instances of an image neighbourhood and arcs represent a possible link between two adjacent neighbourhoods. Intra and inter model relationships are inferred by means of the cliques (or mutually connected vertices) found in this graph.

In summary, this thesis also addresses the important question of whether or not it is possible to learn rigid geometric models from 2-D image evidence (iconic object models) acquired from a set of views. We found an affirmative answer to this question. We concentrate on the structure learning problem and our main objective is to demonstrate how it is possible to automatically derive geometric model relationships from a set of previously learnt object feature models. We assume that a model consists of 2-D object representations learnt from unsegmented and cluttered scenes by means of an iconic vision system which is inspired by some of the mechanisms found in the mammalian visual system:

**Foveated Vision.** Input data is presented in the form of a set of overlapping receptive fields (resembling the human retina) which produces an image smaller in size but retaining high resolution in the middle. The progressively lower resolution periphery is modelled as a log-polar image, which maps changes in scale and orientation into translations in

the log-polar space [ST92].

**Visual Attention.** Fixating the retina at interesting regions of a scene prevents having to process the entire scene at once. Provided that an appropriate attention mechanism is defined, the fixation points can be seen as places in a scene where object features (or components) are most likely to be found. Thus, it is reasonable to think that visual attention can help the process of figuring out subcomponent relationships. Although there is some evidence that humans can also selectively attend to objects in the visual field independently of the eye's gaze direction [Eri90], this mechanism is not considered in this thesis.

**Primal Sketch.** It is hypothesised that primal sketch features, like *edges*, *bars*, *blobs* and *ends* [Mar82], are used by humans as more compact and intelligible representations for image data and also as cues for an attention mechanism. Input to our structure learning program is a set of raw primal sketch features extracted at attended image locations by a neural network in the log-polar space at a number of orientations and contrasts [GFH98, GF01, GF02]. These features are roughly consistent object features regardless of position, orientation and size. The receptive field computations together with a PCA pre-processing stage give an estimate for the local surface reflectance patterns of objects in the scene, which makes it approximately invariant to local changes in the illumination and scene composition.

### 1.3 Overview of the Thesis

The research developed in this thesis can be divided into four main parts, which are presented below in order of development. Most of the thesis contributions are a direct result of parts 2 and 4:

1. Specification of a retina-like representation, including structural and functional aspects, like geometry, receptive fields, light normalisation, properties etc.
2. Design of a method for extracting raw primal sketch features from this representation using a learning-based approach.

3. Clustering of primitive object models using the new feature extraction and image representation developed in the previous stages. It is assumed that interest points are provided by an existing attention mechanism (not investigated in this thesis). A cross-correlation metric on the feature's space (primal sketch and colour information) and the log-polar properties for scaling and rotation are used to do the matching and to obtain geometric relations between data and models.
4. Inference of structured models from a set of clustered primitive object models using a graph-based representation and algorithms.

A review of the field is presented in Chapter 2, where a number of important works on iconic vision, property and symbolic based approaches are reviewed. We also present some related research on learning. Given the large number of ramifications that the visual learning area has, we focused only on learning low-level features and geometric descriptions from objects and scenes, which are closely related to this thesis.

Also in Chapter 2, we examine two important theories about the human visual system which have inspired some aspects of this research: vision from primal sketch to 3-D models by Marr [Mar82] and the four frames model by Feldman [Fel85]. To conclude Chapter 2, we give a tour on a practical iconic vision system that has been developed in this University [Jen94, GF96, Mac97, FM98]. This system was used as an initial framework for this thesis.

Although the image representation used by the existing system was already log-polar, some problems were found and enhancements needed to be made. Instead of trying to adapt the existing code, we decided to design and implement a completely new retina-like representation. The structural and functional designs together with an example of utilisation are presented in Chapter 3.

Now that we had an image representation adequate to our purposes, we moved on to the problem of extracting primal sketch features from it. In the existing iconic system this was done by means of a set of manually defined logical operators. However the results were unreliable and with poor resolution when measuring the feature's strength based on its contrast. Instead of trying to build a model for the operators, which proved to be a difficult task as seen above, learning the features was a sensible option.

After some unsuccessful attempts with a monolithic neural network architecture, we realised that breaking down the classification task into separate modules would solve the problem. Features were first normalised with respect to orientation, then projected onto a PCA basis and only after that they were fed into a set of neural network modules trained to recognise specific types of features. Experiments were carried out on real and synthetic data to evaluate the feature extractor's performance. All the details about the process of extracting primal sketch features can be found in Chapter 4.

The existing iconic system operates by extracting primal sketch planes at interesting regions in a scene, which are then matched against a set of previously acquired iconic models. These models are organised in a subcomponent hierarchy, which is used to improve matching and attention, however, the objects used to form these models were manually selected from a number of scenes. Primal sketch features and colour information guide the attentional process during the initial system's operation, but later on, this is complemented by the relative position and identity of recognised object's components.

In this thesis, we designed an algorithm to automatically acquire iconic models while attending to a sequence of scenes. The basic idea was to continuously match new regions to previously seen ones while recording the relative scales, orientations and similarity scores for each pair of similar regions. Given that a sufficient number of models has been acquired, we show how to infer rigid body relations by means of a graph based representation and algorithms. This part of the research is detailed in Chapter 5.

We finish this thesis in Chapter 6 by summarising all the research done and contributions, as well as giving a list of possible future research to be carried out on each of the main issues addressed here: primal sketch feature extraction, iconic model learning and structure learning.

## Chapter 2

# Literature Review

Vision is a process that produces from images of the external world a description that is useful to the viewer and not cluttered with irrelevant information.  
*D. Marr* [Mar82]

### Contents

---

<b>2.1 Iconic Vision</b> . . . . .	<b>9</b>
<b>2.2 Property Based Approaches</b> . . . . .	<b>13</b>
<b>2.3 Symbolic or Geometric Approaches</b> . . . . .	<b>17</b>
<b>2.4 Learning</b> . . . . .	<b>22</b>
<b>2.5 Vision Frameworks</b> . . . . .	<b>25</b>
<b>2.6 A Practical Iconic Vision System</b> . . . . .	<b>28</b>
<b>2.7 Summary</b> . . . . .	<b>33</b>

---

The context of this thesis is a combination of image, property and geometric based recognition as it investigates the automatic creation of iconic representations using image features and geometric relationships. This chapter is intended to review the most relevant works in these three categories and sheds light on the learning issues related to this thesis. Two important theories about vision developed by Marr [Mar82] and Feldman [Fel85], which complement each other and provide some background information not only for this thesis but also for many other works in the area, are discussed later on in this chapter. Finally, the existing iconic vision system that this thesis extends is described.



## 2.1 Iconic Vision

The term iconic vision is normally associated with the direct use of pictorial information in a system [Jen94, SE95, Mar96, GF96, Mac97, FM98] and this is the meaning adopted throughout this thesis. Some other intermediate or property based descriptions have been used in the literature under the same generic term, but we preferred to describe those references under a separate section. Some authors consider iconic vision as the use of small visual descriptors derived from the original images [RB95, MKK97, MK97]. Others also related it to *appearance based recognition* [MN95, Rao97, MLP<sup>+</sup>96, EM97, HCK97, MS99, VC99, HVC00], which consists in recognising 3-D objects from multiple 2-D views.

In iconic vision one can either comply with the traditional sensor architecture of a uniform array of sampling units, or use alternative representations, like for example log-polar images, which are inspired by the human visual system and, besides requiring less pixels, have some useful mathematical properties.

### 2.1.1 Template Matching

Template matching is one of the simplest techniques for measuring the similarity between two vectors, which in particular can be image representations. It essentially involves matching an object's image description to a given database of model descriptions. Two of the most widely used methods for template matching are image subtraction and correlation matching. Image subtraction is based only on the absolute difference between image and model (template), but requires total control over image intensity, which makes this method only useful in completely controlled imaging conditions. On the other hand, correlation matching uses the statistical correlation between image and template pixels and is considered to be more robust to noise, and illumination effects. Correlation matching is very sensitive to partial occlusion, scale and orientation of the feature and to its absolute grey level, and does not allow for accurate localisation of the matched features.

A different approach to correlation matching is to expand the scene using a set of orthogonal or non-orthogonal basis functions (e.g. Fourier, cosine, wavelets transforms) that closely resemble the template image, which makes the resulting matching process more robust to occlusions when compared with standard template matching [BAR94]. In another approach,

Edwards and Murase [EM97] took advantage of the global occluding interactions between visual objects to adaptively improve the correlation metric.

### 2.1.2 Log-Polar Representation

Traditionally, the photometric information of images is acquired from sensors that have a uniformly distributed rectangular array of sampling units. As a consequence of the sensor architecture, most machine vision applications tend to use matrices or Cartesian representations to manipulate images.

However this is not the way images are acquired in the mammalian visual system. A biologically inspired approach to iconic vision is to transform the original 2-D image into a retina-like representation and then to use this representation as the main data for the matching process. Typically, the input image is re-sampled through the use of a mask consisting of concentric rings of overlapping circular receptive fields whose centres are geometrically spaced from the centre of the mask. If an image is accessed by using the rings (logarithm of the distance of the rings to the retina centre) and sectors of this type of mask then this is called a log-polar image representation. This essentially simulates the mapping between the retina and neurons in the visual cortex [Sch77, Wil83]. Each receptive field value is computed by the application of a function over a region of the input image. This non-uniform sampling can be realised either by software or by a space variant sensor.

A log-polar representation is very attractive because it transforms both rotation and scaling in the Cartesian domain into translation in the log-polar domain. More details about a particular log-polar representation developed in this thesis can be found in Chapter 3.

Many recent and past interesting works involving the use of log-polar images for feature detection, tracking and recognition can be found in the literature. We selected, and briefly comment on here, the ones most related to our research. A broader discussion on space variant sensing and its applications can be found in [ST92].

In a work more related to sensor design and application, Jurie [Jur99] proposed a log-polar mapping with a parameter that allows choosing the range of the logarithm function to be used, so that a same log-polar image could be tuned to have different pixel sizes and distribution, ensuring a high modularity to the mapping. Log-polar pixels were modelled as a fractional

part of the Cartesian input pixels (sub-pixel precision) instead of using simple integer pixel aggregation. He also presented a face recognition application which used colour histogramming [SB91] for tracking faces and eigenfaces [PPS96] for recognition, both defined at the log-polar level.

Lim et al [LWV97] used a multi-resolution analysis to recognise edge-based line, arc and ellipse features. To detect interest points, they searched for minima and maxima points on edges detected in the log-polar image. Camera tilt and pan (foveation parameters) were calculated directly from the current camera position and from the next interest point location. Micro saccades were then implemented in two steps: a Least Median of Squares (LMS) fit to estimate the current parameters of the model and an optimisation process to minimise the error of the data to model fit. A set of three different parametrised equations were used as models to the line, arc and ellipse features.

The iconic system described in Section 2.6 is a typical system that uses the log-polar approach. Another example is given by Siebert and Eising [SE95] who defined a vision architecture formed by a log-polar representation with a Difference of Gaussians as the receptive field function, and a template matching algorithm operating directly on the log-polar space.

There has been a recent interest in porting traditional well established image analysis techniques into space variant image formats. For instance, Bonmassar and Schwartz [BS99] defined a new linear integral transform, which they called the exponential chirp transform, that gives support to Fourier analysis in the log-polar domain.

Motivated by performance concerns about the software implementation of the log-polar transform on conventional machines, some research has been made on the development of hardware implementations. Van der Spiegel et al [dSKC<sup>+</sup>89] developed the first CCD implementation of a log-polar sensor. A CMOS implementation has also been made by Pardo [Par94].

### 2.1.3 Complex-Log Representation

This is a more general formulation for log-polar images in which the mapping from retinal coordinates to cortical ones is expressed in terms of a complex-log transformation [RS90]:  $w =$

$\log(z + a)$ , with real  $a > 0$ ,  $w$  and  $z$  are the complex number coordinates for the complex-log (cortical) and polar (retinal) representations .

If  $a = 0$  in the previous equation, then this is equivalent to the ‘pure’ log-polar transformation described previously. Rojer and Schwartz [RS90] believe that using  $a > 0$  gives a better fit to biological data at the same time as it removes the singularity of the log mapping at the origin ( $\log(0)$ ). On the other hand, they had to deal with a discontinuity on the vertical meridian line of the retina. Moreover, a side effect of using  $a > 0$  is that in the periphery circles are mapped into vertically curved lines and radial lines are mapped into horizontally curved lines in the complex-log domain, contrasting to perfect straight lines in the ‘pure’ log-polar formulation. Therefore they have to make the approximation that these lines of pixels are straight in order to be able to use a uniform grid to represent complex-log images.

Within the context of the local image structure theory, Fischl et al [FSC97] have derived the most common differential operators and surface characteristics in the complex-log domain, like the Laplacian, the gradient and the divergence, as well as the fundamental forms of the image treated as a surface. However these derivations did not account for the varying support (receptive field size) of each log pixel. In any case, this is an important result, since it can allow some sort of edge (or discontinuity) detection directly in the complex-log domain. Although in our work we are also interested in extracting features like edges, we decided to take a different approach, based on learning, so that the design of an operator becomes more generic, depending only on the selection of a representative set of feature exemplars for training.

#### 2.1.4 Log-Log Representation

Iconic vision approaches based on log-polar representations are not invariant to uneven foreshortening distortion. Therefore, if the problem is to recognise 3-D objects at arbitrary poses from 2-D images, the use of a pure log-polar representation is not convenient.

An alternative approach, proposed by Ben-Arie and Wang [BAW97], samples the frequency domain in a configuration which is logarithmic in two orthogonal axes (log-log) . The sampling process can use the Fourier transform, Gabor transform and also two dimensional Gaussian derivatives. The log-log representation is shown to be invariant to translation, slant and scale. Invariance with respect to tilt and rotation around the optical axis is obtained via a combination

of several log-log configurations calculated from different 2-D views of the object. The model base consists of a hash table containing a set of log-log configurations calculated from each model. The matching mechanism indexes the model base using log-log representations of the test object and then using a voting scheme to decide for the best match.

They have obtained excellent results when applying their approach to test images under variations in translation, slant, scale, tilt, swing, noise, resolution and also images composed by neighbouring objects. However the model base consisted of a small number of objects (26 objects) and only variations calculated on these objects were used as test images. Therefore, some interesting aspects were not tested, like the effect of using a large model base, the use of test objects other than the ones in the model base, illumination variations and partial occlusions.

## 2.2 Property Based Approaches

Another way of addressing the problem of recognising objects from 2-D images is to use property based measurements for shape, colour, texture etc. This kind of approach is popular amongst applications involving image database indexing, which is also known as visual information retrieval [BFG96, GJ97], or content-based image retrieval [MSP96, PPS96].

Messer, Kittler and Kraaijveld [MKK97] have presented a system for searching through an image database. The system's input is an image representing the kind of visual object(s) that the user is interested in. The user also needs to specify heuristically two regions in the image of interest, representing the object itself and the background. Colour and texture features are calculated for each pixel in the object and background regions. All the database images have the same features previously calculated. A neural network is trained to classify between object and background pixels using features extracted from randomly selected pixels of the input image. The search process presents all the database features to the trained neural network. The regions in the images which were judged similar to the reference image are the output of the search process. It is possible to identify two main disadvantages in the above approach. The first one is that all of the images have to be examined for a given query. Moreover, many pixels have to be classified for each of these database images, which implies a computational overhead.

### 2.2.1 Multiple Filtering

Instead of using an image-like representation, many authors have opted to use more compact representations that have been generated through the application of multiple filters over the 2-D original images. Obviously, the previous approach of using a log-polar representation can also be classified as a multiple filtering approach when receptive field functions are interpreted as filters. However, due to the particular sensor architecture, it has been described separately.

Rao and Ballard [RB95] proposed an architecture in which the iconic representations are high dimensional feature vectors obtained from the application of multiple Gaussian derivative filters at a number of orientations and scales. The term iconic was used by them with the meaning of small visual templates that constitute visual memory. The filters were applied to Cartesian coordinate images which were produced by uniform resolution sampling. They also briefly discussed the use of the system with space-variant sensors but this part was included as future research.

The system's architecture has two separate internal representations or memories, one indexed by image coordinates and another indexed by object coordinates, which are used by the two basic visual routines the system implements, object location and object identification respectively.

In the object location routine, a single model vector (from the set describing a particular object) is matched against image features at all possible locations of the visual field. The output of this routine is the location of the object. In the object identification routine, vectors from a single localised point in space are matched against a database of models for different objects. More than one model can share the same index, and then the object's identity is determined by a voting process, with the model obtaining the majority of votes being the winner. In both of the above routines, correlation is used as the similarity metric for the matching process.

For small rotations in the image plane of a given object, invariance is achieved by normalising the Gaussian derivative filters. More drastic rotations are handled by storing feature vectors from different views.

One of the advantages of the Rao and Ballard's system is that their image representations are compact and produced by the computation of simple filters which can reduce the computation

time when compared with the processing of geometric models. However, the use of filtering causes the system to be unable to distinguish between objects with similar frequency responses. Further work in the processes involving dynamic imagery can be found in [Rao97]

The approach adopted by Rao and Ballard, presented above, can be seen as part of a more generic framework for image representation called multi-resolution image and feature analysis. Within this context, Koenderink and van Doorn [KvD92] derived a framework (called *generic neighbourhood operators*) for representing an arbitrary set of local feature extraction operators (or local filters) at multiple scales and orientations.

Unlike traditional image feature extraction operators (like the Canny or Sobel edge detectors) in which there is usually no clear transformation relating different operators, Koenderink and van Doorn's approach allowed linear combinations, concatenations, resolution changes or rotations of operators to be treated in a systematic manner. The fundamental basis for their framework was a scale-space [Lin96] that satisfies the *diffusion equation* [BWBD86]. They concluded that Gaussian derivatives are natural operators to derive from a scale-space representation given the assumption of scale invariance. The theory behind these generic operators assumed a continuous image domain and that the support (or size) of the operators should comprise a relatively large area (or volume).

Another important multi-resolution approach are the multi-scale pyramidal representations. Burt and Adelson [BA83] presented the concept of Gaussian and Laplacian pyramids. A Gaussian pyramid is essentially a hierarchy of low-pass filtered versions of an input image, each sampled at successively sparser densities. Highly decorrelated images may be obtained by subtracting the original image from the results of applying the low-pass filters to it, which yields to an encoding scheme where only the decorrelated and filtered images need to be stored (thus requiring fewer pixels to represent the input image since the decorrelated image has smaller variance). The low-pass filtering is done with a Gaussian filter kernel convolved with the input image. A Laplacian pyramid is a similar structure with the difference that the pyramid levels here are different bands of the image frequencies. This is equivalent to take the difference of consecutive levels in the Gaussian pyramid. They have shown that this structure is not only useful for image compression and progressive transmission but also for feature extraction.

### 2.2.2 Colour Histogramming

Swain and Ballard [SB91] presented an object recognition technique based on the matching of a colour histogram calculated from a region of the image with a colour histogram from a sample image of the object. The use of colour histograms has been shown to be robust to changes in the object's scale, orientation, viewing position and partial occlusion. However, the original technique has a high sensitivity to the lighting conditions (colour and intensity of the light source) and the colour of the object itself. This limitation might be overcome by pre-processing the images with a colour constancy algorithm, but usually the 3-D geometry of the scene has to be known (see i.e. [FG96]). In practise, colour constancy is only possible under restrictions and limitations on the type of illumination and the surfaces that will be encountered.

An intermediate approach is the use of measures which are less sensitive to illumination changes. Funt and Finlayson [FF95b] have proposed histogramming colour ratios. The basic assumption in this case is that the ratios of colour *rgb* triples from neighbouring locations are relatively insensitive to changes in the incident illumination.

### 2.2.3 Multidimensional Receptive Field Histograms

Schiele and Crowley [SC96b, SC96a] presented an approach which can be classified as a combination of multiple filtering (e.g. [RB95]) with histogramming (e.g. [SB91]). They used histograms of local shape properties which were calculated as a vector of linear local neighbourhood operators, or receptive fields. In a first work [SC96b] they tested Gaussian derivatives, the Gabor filter and Gradient and Laplacian operators as the receptive field function. To perform histogram comparison, they tested the sum of squared distances, the  $\chi^2$  method, intersection (as in [SB91]) and the Bayes rule.

Instead of using histogram comparison, a different approach is to use the Bayes rule assuming that the probability density distribution of an object is the object's histogram itself apart from a normalisation factor [SC96a]. This approach allows to determine the probability that an object is in a scene based on arbitrarily selected points in the image, and differs from the previous approach in the sense that it works without direct correspondence between the object database and the test image.



As a continuation of the previous work, now in the context of appearance-based recognition, instead of histogramming, Verdière and Crowley [VC99] projected all overlapping neighbourhoods from an image onto a linear subspace (using PCA) to give a sampling of a surface in the linear subspace. Different viewing positions of an image define a family of surfaces which represents the possible appearances. The recognition process consisted in projecting small neighbourhoods from a test image onto the local appearance manifold and associating them to nearby surfaces. Hall et al [HVC00] extended this approach to deal with colour images.

## 2.3 Symbolic or Geometric Approaches

This section presents the symbolic or geometric approaches that are related to the iconic model creation mechanism investigated in this thesis.

In typical model-based vision systems, a collection of features is extracted from the image, a correspondence between these features and an appropriate set of model features is hypothesised, and finally the position and orientation (pose) of the model is determined from this hypothesis. This can be done either by the use of interpretation tree techniques (e.g. [Gri90]) or by transformation determination methods (e.g. [UII96]). Some drawbacks associated with this approach are that all the models have to be tested in order to identify model instances in the image and there are many possible matches between image and model features, each match being expensive to verify [RZFM91].

### 2.3.1 IMAGINE System

IMAGINE I and II [Fis86, Fis89] are 3-D object recognition systems that employ model invocation and geometrical matching using range images. The importance of reviewing this typical symbolic/geometric approach is the fact that the model creation and invocation mechanisms can be modified or adapted to work in an iconic vision context. An approach inspired by these mechanisms is presented in Chapter 5.

## Overview

The first processing of these systems is to segment the range images into surface patches. Obscured portions of surfaces are reconstructed by a surface completion process, which joins extrapolated surface boundaries. This process tries to produce a complete set of surface hypotheses that are then joined to form surface clusters. Surface clusters represents 3-D solids whose identities are not yet known, and are used for aggregating image features into contexts for model invocation and matching. The next step is to compute some important viewpoint invariant 3-D feature properties like, for example, surface area and relative orientation.

Models are built as a set of compact and connected solids that have definable shapes. Models may also be defined, depending upon the nature of the objects, as subcomponents joined by interconnections with given degrees of freedom. In IMAGINE II, models are defined using a special language called SMS and represent a wide range of geometric properties like surfaces, boundaries etc. A model is structured in terms of assemblies which can have relationships with other assemblies, in a way similar to that used in the geometric descriptions for machine learning shown by Winston [Win77], which is discussed later on in this section.

The following step is model invocation, which selects a few candidate models for further consideration. The model invocation process is justified because of the impossibility of finding the correct model by sequential comparison of an object hypothesis with all known models. Moreover, an exact matching may not occur because of sensor noise, object variation or the use of a generic description. The two final processes carried out in the IMAGINE approach are hypothesis construction and identity verification. The former tries to find evidence for all model features and the latter helps to ensure that instantiated hypotheses are valid objects and have the correct identity.

## Model Invocation

Model invocation works as a flow of evidence in a network. This network is a kind of surface cluster tree which is derived from the input image and from the model description. Each node of this network represents a possible pairing of model-to-data and is valued by a plausibility metric. This plausibility represents the degree to which an object model explains an image structure.

Plausibility comes from accumulating evidence in the context of either surface patches or surface clusters. There are two kinds of evidence: property evidence, which comes from measured features, and relationship evidence. The links between the nodes in the invocation network represent the different relationships between them. Below, we list the kinds of relationship evidence that were proposed by Fisher [Fis86, Fis89] (M is the model of current interest):

1. Supercomponent: A is a structure of which M is a subcomponent.
2. Subcomponent: A is a subcomponent of structure M.
3. Superclass: A is a more generic class of object than M.
4. Subclass: A is a more specific class of object than M.
5. Description: Every property of A is a property of M.
6. Inhibition: Identity A competes with identity M.
7. Association: The presence of object A makes the presence of M more likely.

The surface measurements produce evidence which is propagated through the network. There is a continuous update of the plausibility values until the network goes to a stable state. Given the plausibility ranking, the data-to-model nodes that have a plausibility higher than a given threshold are then invoked.

Further improvement to the superclass evidence was made by Paechter [Pae87]. He changed the way that this kind of evidence is integrated in the network. In summary, these changes allowed the indirect inheritance of properties, like surface descriptions or subcomponents, through the superclass link. The result of this was a system that could discriminate better between correct and incorrect model-to-data pairings.

### 2.3.2 Projective Invariants

The importance of indexing in object recognition problems involving large numbers of models is a subject frequently discussed nowadays (see e.g. [Nel96] and [Gri90]). This was mainly discussed in the context of geometric representations.

A different way of implementing a model-based vision system is through the use of projective invariants [RZFM91, RZMF92, Rot95], which neither rely on the pose consistency of image

feature to model feature assignments in order to form hypothesis, nor on camera calibration. Projective invariants are used in this context as properties of planar shapes which are unaffected by perspective imaging. The invariant values are computed from algebraic curves fitted to edge data of a 2-D image. An example of a planar projective invariant is the cross-ratio  $C$  for four coplanar points  $(x_1, x_2, x_3, x_4)$  along lines with linear parameterisations  $(\theta_1, \theta_2, \theta_3, \theta_4)$ , see Equation (2.1). It is possible to establish invariant relations involving five coplanar points (or lines), a conic and three points (or lines), a pair of conics and so on.

$$C(x_1, x_2, x_3, x_4) = \left( \frac{\theta_1 - \theta_3}{\theta_1 - \theta_4} \right) / \left( \frac{\theta_2 - \theta_3}{\theta_2 - \theta_4} \right) \quad (2.1)$$

Rothwell et al [RZMF92] experimented with a model based vision system for plane objects that uses projective invariants. The main processes of the system are feature extraction, model acquisition, hypothesis generation and hypothesis verification. In the feature extraction process, the conics and lines needed to form the invariants are extracted from image edge data. In the model acquisition stage, two images of an unoccluded object at different views are used for the computation of the model invariants. All of the invariants that are similar in both views are stored in the library jointly with edge data from one of the acquisition images. In the hypothesis generation process, invariants for groups of features in a scene are extracted and then used as indexes to the model library using a hash table. All the potential matches found for a given object produce recognition hypotheses for that object. Finally, in the hypothesis verification process, model features and edge data from an acquisition image are projected into the test scene. If these data are similar then the potential match is confirmed. Figure 2.1 illustrates simple objects recognised by this system, despite the strong perspective distortion.

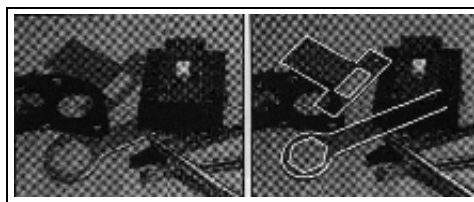


Figure 2.1: Graphical result of the recognition of two objects, despite perspective distortion, in the system proposed by Rothwell et al [RZMF92].

The use of projective invariants together with this kind of hypothesis verification make the system particularly robust to the recognition of occluded objects. A review of a more complete

object recognition system called LEWIS, which was derived from the above system, can be found in [Rot95].

The main advantages of the projective invariant approach is that models can be acquired directly from images and invariant values can be used to index a model instead of exhaustively consulting all the models in the model base or having to calculate pose and camera calibration parameters. However, models have to be selected manually. Moreover, it can only be applied to images containing planar objects, or objects with plane surfaces. Therefore, it is still unclear how this approach could be applied to 2-D image projections of real 3-D arbitrary shape objects. Moreover, the existing systems were tested over model bases consisting of only tens of models and this is still too small to experience the real difficulties with large model libraries.

### 2.3.3 Comparing Geometric Invariants and Appearance-Based Systems

An experimental comparison between geometric and appearance based approaches applied to a problem involving very simple objects (a planar object and a rotationally symmetrical one) can be found in [MLP<sup>+</sup>96]. An appearance-based approach is merely an iconic approach in which several views of a same object are taken into account when building a visual model.

The systems used for comparison were SLAM [NNM94] (appearance-based) and Lewis [Rot95] / Morse [ZFM<sup>+</sup>95] (based on geometric invariants). Four objects were used in the experiments: three SORs (Surfaces of Revolution) and one planar shape (a floppy disk). Two kinds of experiments were carried on the data. The first was to evaluate the systems' ability to recognise objects when they are given the same model libraries. The results of this test was that the geometric invariant system (Morse) presented no false positives whereas the SLAM system presented 4 false positives amongst 15 test images. The second experiment aimed to investigate how an increase in the number of objects in the model libraries would affect performance. Again, the geometric systems proved to be fail-safe because they produced no false positives, which wasn't the same with the appearance-based one. The performance of the appearance based system was affected by the addition of some particular library models, whereas in the geometric systems the performance was not affected by adding additional objects to the library.

Despite of these interesting results, the most important contribution of the paper was the general observations about the problem. It could be said that the results represent a comparison

of the two recognition approaches under identical conditions, i.e. using the same set of test images and model libraries. However, the use of a very small model library can be a cause for the apparent better performance of the geometric system. Moreover, it is possible that the use of the three SOR models, which are very convenient to the geometric systems, against just one planar object, could have influenced the quantitative results.

Mundy et al [MLP<sup>+</sup>96] argued that appearance-based models take advantage from the fact of not having to build a formal description of the constraints associated with an object. On the other hand, with geometric-based models, this description is important to the process of pose invariant recognition. The problem of building geometric descriptions of objects is that they can not be currently applied to arbitrary-shape objects, in fact they can only be used to a limited set of geometric classes such as SORs, planar surfaces and canal surfaces. It is claimed that one of the advantages of the geometric approach is that geometric class models also provide constraints which support figure-ground segmentation, and that in appearance-based systems figure-ground separation is a harder task. The presence of mutual illumination and shadows can lead to complex and unpredictable patterns of intensity in real scenes, which is clearly a drawback of the appearance-based approach. It is expected that geometric boundary descriptions tend to be more invariant than normalised intensity patterns.

## 2.4 Learning

Machine Learning is an area that is receiving great interest by Computer Vision researchers in recent years. Learning can offer effective methods to a wide variety of Computer Vision problems ranging from the extraction of low level features to the acquisition and recognition of high level visual (2D or 3D) models and behaviours.

There exists a fair number of well founded learning technologies available, including Neural Networks, Bayesian and Probabilistic Modelling, Principal Component Analysis, Support Vector Machines amongst others.

The first part of this thesis addresses the problem of extracting low level (primal sketch) features from a log-polar image representation using a neural network approach. Later, we develop a method to automate the acquisition of geometric single-view primitive object models and show how larger structured models can be learnt/inferred from these primitive models.

Thus, we decided to limit the scope of this section to the issues of learning features with neural networks and learning geometric descriptions, which are closely related to this thesis.

### 2.4.1 Learning Low-Level Features

Earlier work on Neural Network learning of edge features has obtained only a limited success, as for example [PBC92] in which a concatenation of two Perceptrons was used: one for noise filtering and another one for edge detection. The edge detection network was trained to recover a given edge component within a 3x3 window at 8 different orientations (the position of an output neuron represented the orientation, and the neuron's output value corresponded to the edge intensity value). The results showed that the neural network approach had a performance slightly inferior to that of the Sobel edge detector.

Chen *et al* [CTR95] presented an edge labelling process in which a neural network was trained with synthetic data from a model of an ideal step edge. The aim of the system was to label the central pixel of a 5x5 image patch as edge or non-edge. Illumination and rotation normalisation were performed over the image patch before feeding it into the neural network, which reduced the complexity of the problem to be learnt. They compared the visual output of their system with the output produced by the Canny edge detector when applied to the same noisy data, and found that the neural network had better noise tolerance ability than the Canny edge detector.

### 2.4.2 Learning Geometric Descriptions

Winston [Win77] did some of the first research showing how to learn structural descriptions from examples. He represented simple scenes as networks of linked concepts. The starting point is a scene composed of simple building blocks like bricks, wedges etc, plus a set of relations between each of these blocks. An example is given in Figure 2.2.

Learning is driven by categorised differences between descriptions. The idea is to gradually refine the model by presenting a sequence of sample objects. The first step is to build symbolic descriptions of the objects. Next, each sample description is matched against the model in order to discover differences and similarities; this is done by deciding how nodes in a network corresponds to nodes in another. Once the pairing of nodes is concluded, a skeletal framework, which reflects the common structure found in both networks, is built. The following step is to

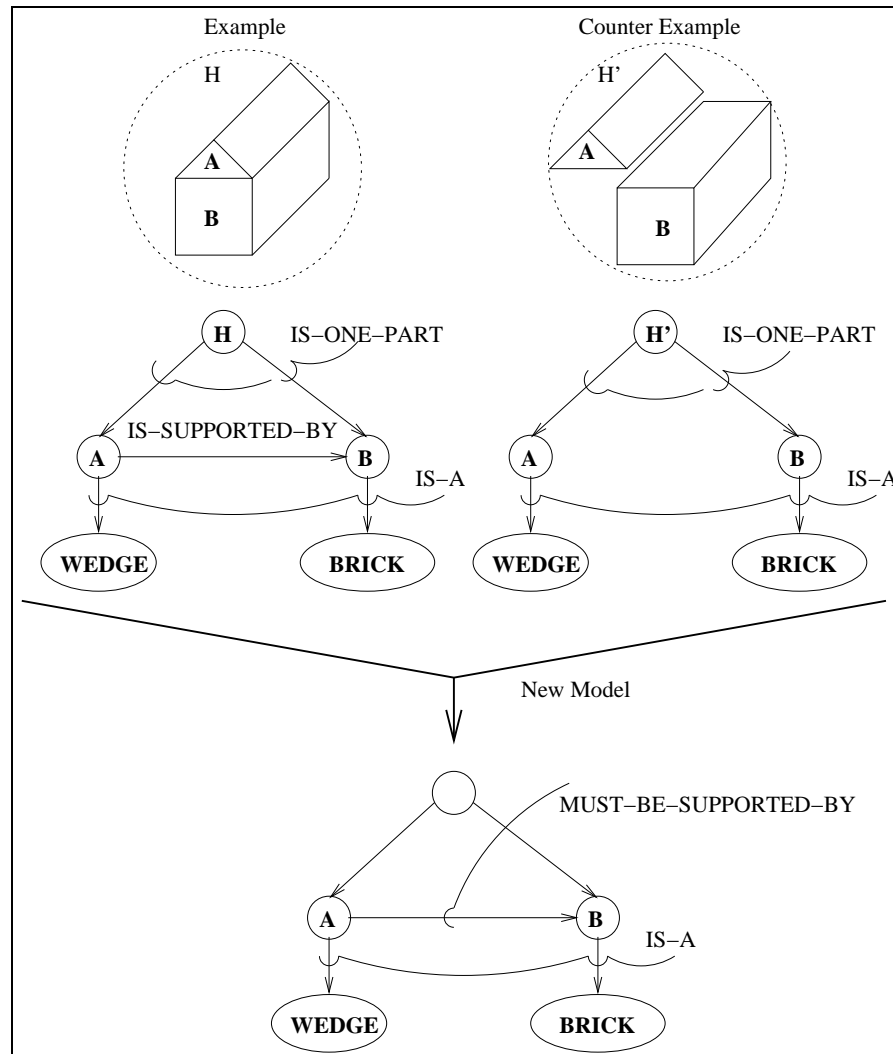


Figure 2.2: Three kinds of relation involving parts of objects are shown in the figure. Analysing the upper left side of the figure: block **A** IS-SUPPORTED-BY block **B**; block **A** IS-A **WEDGE** and block **B** IS-A **BRICK**; block **A** IS-ONE-PART of an instance of a house **H** and block **B** IS-ONE-PART of an instance of a house **H**. This figure also shows how a new model can be derived from an example (a house) and a counter example (a house where the support relation is gone), the result is the learning of the relation MUST-BE-SUPPORTED-BY (adapted from [Win77]).



describe the individual differences; this is done by a set of annotations attached to the skeleton indicating whether a relationship is in a network but not in the other.

Differences between scenes can be expressed in terms of a few types of annotations (difference notes). An example of a difference note is an EXIT note: it is placed whenever one network extends a relation and the other does not (see Figure 2.3). There is a small set of rules that are used to transform the skeleton and its difference notes into a new model. Finally, learning happens when the skeleton and its difference notes are transformed into a new model.

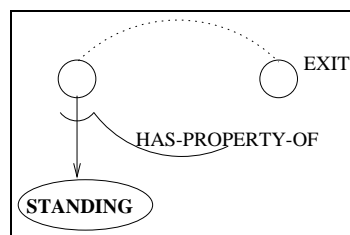


Figure 2.3: The EXIT difference note is placed in a skeleton whenever one network extends a pointer (relationship) while the other does not (adapted from [Win77]).

One problem with this approach is that the learning process becomes highly dependent upon the quality and presentation order of the examples and counter-examples. The presentation of a single incorrect example of an object amongst several correct ones can create a completely incorrect model.

## 2.5 Vision Frameworks

This section presents some theoretical background information associated with the thesis. Two relevant frameworks proposed by Marr [Mar82] and Feldman [Fel85] are discussed. In the following section, the iconic vision system developed by Grove and Fisher [GF96], which has been used as the initial framework for the proposed research, is also discussed.

### 2.5.1 From Primal Sketch to 3-D Models

Marr [Mar82] proposed three different levels for the understanding of information processing systems (having vision systems as the target example): computational theory; representation and algorithm; and hardware implementation. One of Marr's most important contributions

was made in the level of representation and algorithm when he proposed a representational framework for vision (Figure 2.4). He concentrated on the vision task of deriving shape information from images.

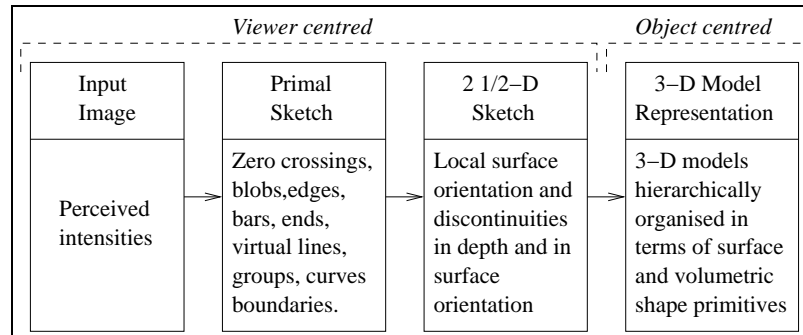


Figure 2.4: Marr's representational framework.

It is known that the intensities perceived by any visual system are a function of four main factors: the geometry (meaning shape and relative placement); the reflectance of the visible surfaces; the illumination; and the viewpoint. According to Marr's theory [Mar82], the early visual system derives representations in which these factors are separated. The first two representations in Marr's framework, the primal sketch and the  $2\frac{1}{2}$ -D sketch, are intended to record that separation.

The detection of intensity changes, the representation and analysis of local geometric structures and the detection of illumination effects take place in the process of generation of the primal sketch. One important principle of the primal sketch is that independent spatial organisations of the viewed intensities in a scene reflects the structure of the visible surfaces. Marr proposed to capture these organisations by using a set of "place tokens", or low level features, which correspond to oriented *edges*, *bars*, *ends* and *blobs*, which were represented by a 5-tuple: (*type*, *position*, *orientation*, *scale*, *contrast*). There are two additional representation levels, the  $2\frac{1}{2}$ -D sketch and the 3-D model, based on the primal sketch, but these are not particularly relevant to our research and therefore are not discussed further.

### An Expansion of the Marr's Theory

More recently, Watt [Wat88] built a theory about vision based on Marr's theory. Watt reformulated the early visual process interpretation in terms of signal processing filters.

However, the main ideas regarding the sketch hypothesis were still maintained.

### 2.5.2 Four Frames

Feldman [Fel85] proposed a theory to help both the understanding of the mammalian visual system and to serve as a framework for vision and space. He tried to show how each part of his system could be explained in terms of neural connections and hypothesised that four frames of reference would suffice in explaining the vision process: retinotopic frame; stable feature frame; world knowledge formulary; and environmental frame (Figure 2.5).

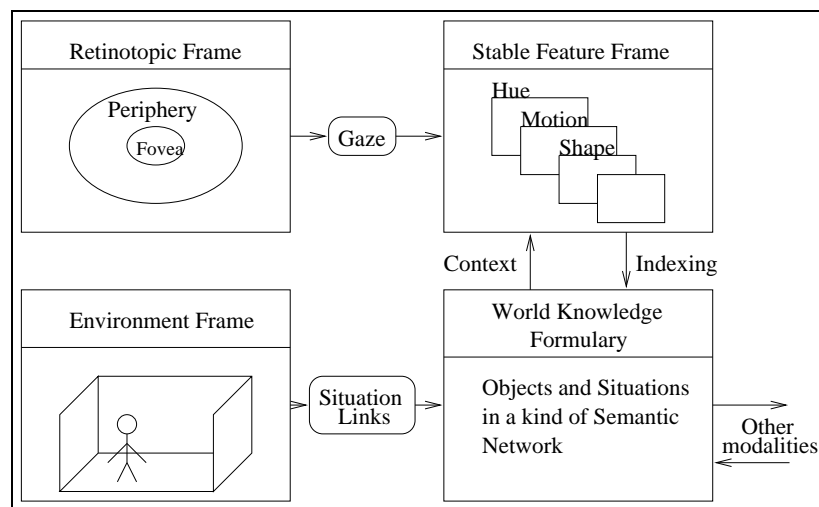


Figure 2.5: Feldman's four frames model (adapted from [Fel85]).

The retinotopic frame models the view of the world that changes with each eye movement and it is essentially a viewer-centred representation. The stable feature frame deals with the idea of keeping a stable mental visual world (viewer centred representation) independent of saccadic retinotopic activity. It is composed of planes encoding lightness, hue, texture, shape, motion, size and depth. The feature frame incorporates two retinotopic frame readings and disparity information when available (simulating the two eyes). The world knowledge formulary is the observer general knowledge (modelled by a kind of semantic network) of the world, including objects and situations. The visual appearance of objects is encoded here as a collection of relationships between primitive parts. This frame is understood as being object centred. Finally, the environmental frame is the representation of the space around the animal or system at a given time, and is viewer centred.

## Comparison with the Marr's Approach

Feldman's approach [Fel85] is in some aspects similar to that of Marr [Mar82]. Marr's primal sketch (augmented with motion, colour, and disparity data) can be compared to the Feldman's retinotopic frame. Also, the two approaches agree on the use of hierarchical object oriented descriptions. However, Marr's approach does not have a structure equivalent to the stable feature frame (Marr's final representation is object centred) and does not consider context and visual cues other than shape.

## 2.6 A Practical Iconic Vision System

Grove and Fisher [GF96] presented an iconic vision system based on primal sketch features extracted from a log-polar representation. The system was based on initial work developed by Jennens [Jen94].

### 2.6.1 The Initial System

The system's architecture is composed of several representations and processes, which correspond to boxes and rounded boxes, respectively, in Figure 2.6. Input to the system is a large, static colour Cartesian image. Part of this image is selected by an attention process and then is foveated (transformed into a log-polar representation). An uniform average function is used for computing the receptive field values (each pixel in the log-polar image).

After foveating the input image, feature extraction operators are applied to produce different primal sketch feature image planes (*edges*, *bars*, *blobs* and *ends*). In order to locate features at different spatial frequencies, three different scales (1, 1/2 and 1/4 of the original image size) are used on the input image. Both the foveated original *rgb* image and the primal sketch images are stored in a structure named the image stack. In order to obtain information about nearby items already matched, the image stack also contains data obtained from foveated label planes of the stable feature frame (described later in this section). The operators were manually defined as logical expressions involving the pixels of a 3x3 window which is applied through the log-polar image. Figure 2.7 illustrates the 3x3 mask used by the operators in a rectangular retinal tessellation (i.e., the receptive fields are radially aligned). There was also a similar set

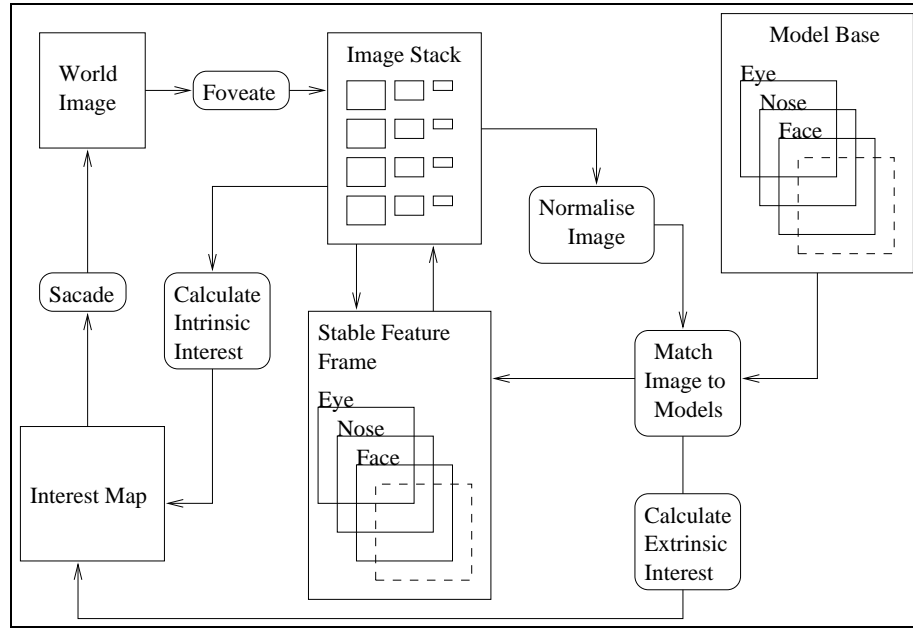


Figure 2.6: Grove and Fisher architecture (adapted from [GF96]).

of operators to use in a triangular retinal tessellation (i.e., the receptive fields are shifted by half a sector every 2 rings). Some operators defining *negative corner*, *positive blob* and *positive vertical bar* features are exemplified in Equations (2.2), (2.3) and (2.4), respectively.

$$-Corner = \min(p1 - x, p3 - x, p4 - x) - \frac{|x - p6|}{2} \quad (2.2)$$

$$+Blob = \min(x - p0, x - p1, x - p2, x - p3, x - p4, x - p5, x - p6, x - p7) \quad (2.3)$$

$$+Bar@90 = \min(p1 - p0, x - p3, p6 - p5, p1 - p2, x - p4, p6 - p7) \\ -\max(|x - p1|, |x - p6|) \quad (2.4)$$

A set of models that may be matched against the current image stack is maintained in a model base. Models have the same structure as the entries of the image stack. An extended multi-variate cross-correlation matching algorithm is used for correlating models to data. The models are created through the manual selection of pictures which are supposed to be the best representatives of the class the model denotes. Weights are associated with each model plane in order to indicate the importance of the feature in the discrimination of a particular object. The weights were learnt using a MLP-backpropagation neural network.

The image stack and the model base are essentially retinocentric representations because they

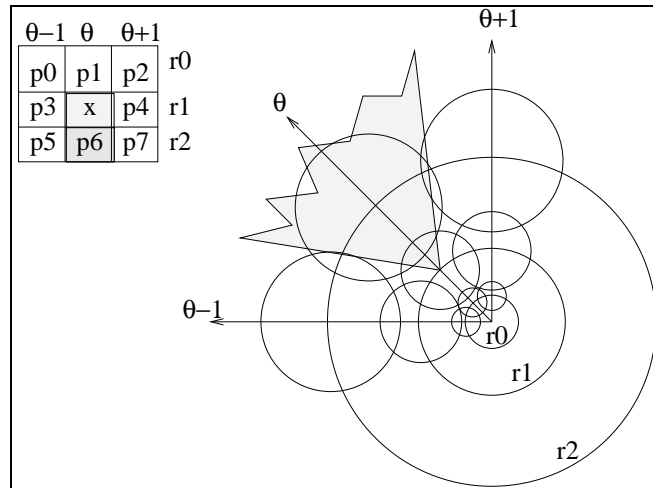


Figure 2.7: The 3x3 mask used by the Grove and Fisher system [GF96] to detect features. Each pixel in the mask corresponds to a particular receptive field output in the polar coordinate system. A triangular retinal tessellation was used.

are registered on the gaze position. But the stable feature frame and the interest map are world-centred representations.

The stable feature frame (SFF) is a stack of label planes, which works as the system's visual short-term memory and has the size of the input image. Each model label's plane stores the locations of matched objects and their match scores of that model type.

The interest map is the data representation for the attention mechanism. The highest valued pixel in the interest map indicates the next point to be foveated in the input image (saccade). The interest map is updated according to intrinsic interest, extrinsic interest and suppression. The extrinsic interest is obtained from subcomponent evidence accumulation which is achieved by the process of correlating the image stack with the label stack. The intrinsic interest is obtained from the accumulation of the detected primal sketch features. Suppression is used to avoid constantly re-foveating the same region. Suppression is implemented by adding a large negative constant to the region of the interest map corresponding to the location of a recently matched model. The system also implements micro-saccades, which consists in foveating at some extra points nearby the foveation point, because the localisation of interesting features in the low spatial frequency channels and in the periphery of the image is usually imprecise.

### 2.6.2 Some Criticism

The retina was entirely modelled as log-polar image, which simplified the matching and attention algorithms, but because receptive fields within the centre of the field of view had varying sizes, it became difficult finding and matching small, high resolution objects. Also, there was no treatment for estimating the reflectance information in the images in order to obtain some degree of invariance with respect to scene illumination conditions.

One problem with the use of an average function in the receptive field computation is that it introduces loss in the transformation process. More interesting functions could have been used. For instance, Gaussians and Difference of Gaussians (DOG) functions produce an output that is sensitive to intensity changes in the input image [Mar82]. Moreover, the use of more powerful functions, such as the Gabor functions, of which the DOG is a particular case, could improve the outputs of the receptive field calculation. When using an appropriate family of Gabor functions, with different scales and overlapping, the resulting representation is so rich that it would be possible to reconstruct totally the input image. This is particularly investigated by Lindberg [Lin94] in his scale-space theory.

It is possible to identify two main disadvantages with the way feature extraction has been made in this system. The first disadvantage is that the operators are heuristically defined and then there is no guarantee that they will work correctly with all the possible cases and that they will allow graceful degradation. The second is that whenever one needs a different window size or window shape it will be necessary to design new logical expressions for the operators which can produce mistakes as the operators are defined by hand. In Chapter 4, a learning-based approach which deals with the above problems is presented.

Two deficiencies can be seen in the way the model base was defined in this system. The first one is the manual selection of sample images to form the model base doesn't assure that the images are really good representatives of the class the model will denote. The second is that all the models have to be accessed in a first execution of the system before a matching can occur, and this contradicts the principle of model invocation: not all the models need to be accessed prior to a matching because of the potentially huge amount of models that can be present in the model base.

The intrinsic interest, which is obtained from primal sketch planes, could be improved by the

insertion of other context free cues, like symmetry (e.g. [YRW92]). Hierarchical associations between the models could enhance the extrinsic interest, which is currently obtained only from general associations. More details about automatically building geometric models from the image evidence and inferring relationships between these models can be found in Chapter 5.

### 2.6.3 Relation with the Theoretical Systems

Marr's primal sketch notion was used in this system in order to both provide cues to an attention mechanism and to define intermediate representation of objects. The  $2\frac{1}{2}$ -D sketch and the 3-D model representation proposed by Marr were not used in this system, mainly because the system deals only with iconic representations and Marr's representations are clearly geometric ones. In fact, these are complementary representations that are used in different contexts: the iconic system works with 2-D appearance and Marr's proposal deals directly with 3-D scenes.

The idea employed in this iconic system, consisting of a retina-like mask which foveates regions of the input image according to an attention mechanism, is very similar to Feldman's retinotopic frame. Feldman's stable feature frame was directly used in this iconic system but having the original lightness, hue, texture, shape, motion, size and depth substituted by primal sketch and model feature planes.

There is a close relationship between the Feldman's world knowledge formulary and the representations associated with model invocation in [Fis86, Fis89] and, more recently, the full subcomponent evidence improvement made by MacKirdy and Fisher [Mac97, FM98] to Grove and Fisher's system [GF96], which is discussed later on in this section.

### 2.6.4 Adding Parallelism

Marques [Mar96] built a parallel implementation of the system using the master-slave model of parallelism. Based on the fact that each micro-saccade performs tasks that are independent of each other, he chose a micro-saccade as the unit of parallelism.

A master process centralises the operation of the system by sending foveation points, denoting the micro-saccades, to corresponding slave processes which effectively foveates the images and perform the correlation between model and data. Finally the master process decides the



best match amongst the correlation results received from the slave processes.

One problem with Marques' implementation is that the label planes of the stable feature frame were not updated in the slave side of the program, and thus the parallel implementation of the system could not properly use the information stored in the stable feature frame.

### **2.6.5 Improving Parallelism and Adding Full Subcomponent Evidence**

More recently, MacKirdy [Mac97] made improvements to the parallel implementation through the change of the master-slave model of parallelism into a task-farming one. Within the new system's architecture he implemented an efficient version of the Stable Feature Frame (SFF) and the associated model's labels based on the use of a hash table indexed by the model name or feature type, e.g. nose, face, eyes, mouth.

In the original system, each label plane in the SFF represents matched features with areas of intensity at the position of their match. In MacKirdy's implementation, each label plane is now a list of previously recognised items and then real world size SFF images are no longer needed. In the case of associated model's labels, each entry in the hash table represents a type of relative feature as a list of relative vectors where these other features were found in a scene.

Using the new SFF, the subcomponent evidence mechanism was re-implemented in a symbolic way. He also solved the inconsistency problem of the previous parallel implementation, which did not properly update the SFF amongst master and slave processors. In this new version of the system, the models can also be learnt from objects selected from multiple image examples rather than using only a single example, which happened in the previous version of the system. However, exemplars of objects for defining the models were still manually selected, segmented and parametrised.

## **2.7 Summary**

In this chapter a number of related works has been reviewed under the three main context areas that this thesis fits in: iconic, property and geometric based recognition. An overview of feature and structure learning is also included. This chapter also reviews two interesting theoretical frameworks for vision and the implementation of a practical iconic vision system

which embodies some key aspects of those frameworks: the retinotopic frame and primal sketch; the notion of a stable visual world; and hierarchical object oriented descriptions. The iconic vision system improved with parallelism and full subcomponent evidence has overcome some of the performance limitations of the initial system, but there were still a lot of improvement to be made. This system received a particular interest because the kind of improvement which was required suited perfectly the scientific questions investigated in this research, i.e., the system lacked a good primal sketch feature extraction process and a mechanism for automated creation of visual models.

Marr [Mar82] investigated ways of deriving primal sketch image-based features using the Laplacian of Gaussian operator to find intensity changes in images. Many other mathematical operators have been used in the literature to detect edges, find the skeleton of shapes and so on, as for example the Canny operator, mathematical morphology etc [GW92]. Most of the state of the art work on object recognition has been concentrated on finding property based features, see for example Rao and Ballard [RB95] with their multiple Gaussian derivative filters and Schiele and Crowley [SC96b, SC96a] with their multidimensional receptive field histograms. Although learning has been used by some researchers to classify edges, e.g. [CTR95], it was not used before to learn primal sketch features from a log-polar representation.

The fundamental work by Koenderink and van Doorn's on *generic neighbourhood operators* is related to ours in the sense that we too are interested in feature extraction operators but use a different approach. The response of feature extraction operators based on derivative of Gaussians proved useful to the design of 'real' operators that respond to the local structural information of an image. Differential geometry is a mathematical tool that usually allows a formal construction of such operators in an isotropic neighbourhood. Since the image representation we used was non-isotropic and at the time we started this work there was practically no differential geometry results available for log-polar images (the preliminary steps toward this have been taken by Fischl et al [FSC97] in the complex-log domain), we decided to take an alternative route, based on learning.

The log-polar representation used in this thesis can also be seen as a special case of a scale-space in which scale changes are space variant. Pyramid techniques, like that proposed by Burt and Adelson [BA83], have also some relation to our work since the Gaussian filtering that takes place at multiple scales when deriving the pyramid image structure may be compared to the

overlapping Gaussian receptive fields of our image representation. Likewise with Gaussian / Laplacian pyramids, our representation also involves blurring the image.

A general difference between previous approaches and the one adopted in this thesis is that we have used a model which tries to capture interesting properties of the primate visual system architecture. Most previous research has used a Cartesian feature space whereas we are detecting features in a log-polar space, which is not so convenient for the extraction of features like edges or bars in the traditional sense used in Machine Vision. Another specific difference is that our aim is to classify several different features, in addition to edges, at a number of different orientations and contrasts, which makes modelling the features a difficult task.

Learning classes of object models is a problem that has received great attention from both Machine Vision and Pattern Recognition research. However, to date, no one has yet studied the autonomous acquisition of visual geometric 2-D models by taking advantage of the interesting properties of an attention changing vision system with a primal-sketch log-polar based image representation. Moreover, the inference of rigid body relations between these models is also a subject that has received little or no attention in the literature.

Given the above, one of our two main goals is to use a learning-based approach to extract primal sketch features from log-polar images in order to derive a rich representation that could be used for attention, matching and structure learning. The other main goal is twofold: (a) to design an algorithm for using the above image representation to construct a database of geometric object models from a sequence of scenes; and, most importantly, (b) to devise a solution to the problem of how to infer rigid body object relations by looking at this database.

In order to achieve these goals, our first step was to design and implement a biologically inspired retina-like image representation that would take advantage of the inherent properties of log-polar images discussed above. The following chapter details the structural design and functional aspects of this image representation.

## *Chapter 3*

# *Retina-Like Image Representation*

Human Vision is both active and space-variant. Recent interest in exploiting these characteristics for machine vision naturally focuses attention on the design parameters of a space variant sensor.

*A. Rojer and E. Schwartz [RS90]*

### **Contents**

---

<b>3.1 Motivations</b> . . . . .	<b>37</b>
<b>3.2 Structural Design</b> . . . . .	<b>38</b>
<b>3.3 Functionality</b> . . . . .	<b>54</b>
<b>3.4 Retina Parameters and Example</b> . . . . .	<b>59</b>
<b>3.5 Summary</b> . . . . .	<b>60</b>

---

This chapter presents an image representation which is inspired by the mammalian retina. This representation improves the image representation used as main input to the iconic vision system considered in this thesis. Initially, the motivations and considerations for developing such a representation are given. Then the structural details are described and finally the overall functionality and properties are presented and exemplified.

### 3.1 Motivations

In the previous chapter, log-polar images were presented as a biologically inspired alternative to the traditional Cartesian image representation in iconic vision.

Some authors, like Sandini and Tristarelli [ST92], Grove and Fisher [GF96] and many others, preferred a simpler, yet more tractable, formulation for the log-polar representation. Usually, the choice for a simpler model occurs because the primary goal is to implement principles that can produce a clear practical benefit and not just to build a detailed model of a biological system. Sometimes this choice is also linked to implementation and performance constraints.

On the other hand, authors like Rojer and Schwartz [RS90], demonstrated interest in modelling parts of the biological vision system not only with the purpose of incorporating its interesting properties into an artificial system, but also to help improve the understanding of the natural one. In the latter case, models sometimes are more complex, computationally expensive and their components don't necessarily have a clear practical use in a real artificial system.

Because the main objectives of this thesis are mostly related to the area of Machine Vision, there is no explicit intention in creating an image representation that accurately mimics the one humans and other animals use. Therefore, the image representation developed in this chapter, although inspired by the human retina, is mainly intended to help improve the representation which is input to the iconic vision system [GF96] used as framework to the practical issues of this thesis.

At the end of Chapter 2, this iconic vision system was discussed and some criticism presented. Amongst the problems encountered, some were directly related to the image representation used. The retina was entirely modelled as a log-polar image, which simplified the matching and attention algorithms, but because receptive fields within the centre of the field of view had varying sizes, finding and matching high resolution features was not possible. Also, a poor receptive field function was used and no treatment was given for estimating the reflectance information in the scenes in order to acquire some degree of invariance with respect to scene illumination conditions.

The main purpose of this chapter is to deal with the above problems and present a mathematical specification for the image representation developed. In the next section, the structural aspects

are presented, concentrating on the functions for transforming and accessing pixels in this new representation. Then, in Section 3.3, the aspects relating to the receptive field function and reflectance information are discussed. This chapter concludes with an example showing the image transformations applied to real images.

## 3.2 Structural Design

Internally, the human eye (as well as the eye of many other animals) has a photo-receptor cell mosaic, named the retina, which is responsible for receiving input light signals and passing them to layers of ganglion neurons. These neurons perform some kind of pre-processing before sending signals to the visual cortex. Each of these neurons collects the outputs of many photo-receptors over an approximately circular area called its receptive field. Roughly, the human retina is formed by two concentric regions. The innermost region, named the fovea, can be approximated as a set of receptive fields hexagonally distributed in a maximal uniform packing density. The outer region can be approximated by receptive fields distributed hexagonally with an exponentially decreasing sampling density. Figure 3.1 presents a diagram of the retina structure described in this chapter.

### 3.2.1 Fovea

The fovea is the innermost and most dense part of the retina. In the fovea, receptive fields have the same radius  $\hat{r}$ . As they are circular regions, to assure a complete sampling of the input image, each receptive field overlaps by a certain percentage  $\hat{\delta}$  of its diameter with all of its neighbours.

A polar-like coordinate system is used in the fovea to facilitate the naming and accessing the neighbourhood of a receptive field and also to maintain consistency with the coordinate system used outside the fovea, which is naturally polar. A neighbourhood is defined as a set of rings (“hexagonal rings”) surrounding a receptive field, and the position of a specific neighbour in a ring is given by an angular displacement. The hexagonal distribution of receptive fields can be viewed as a sequence of concentric rings in which each ring contains a multiple of six receptive fields. Within this context, the whole fovea structure is composed of a central receptive field surrounded by a number of receptive field rings, see the central portion of Figure 3.1.

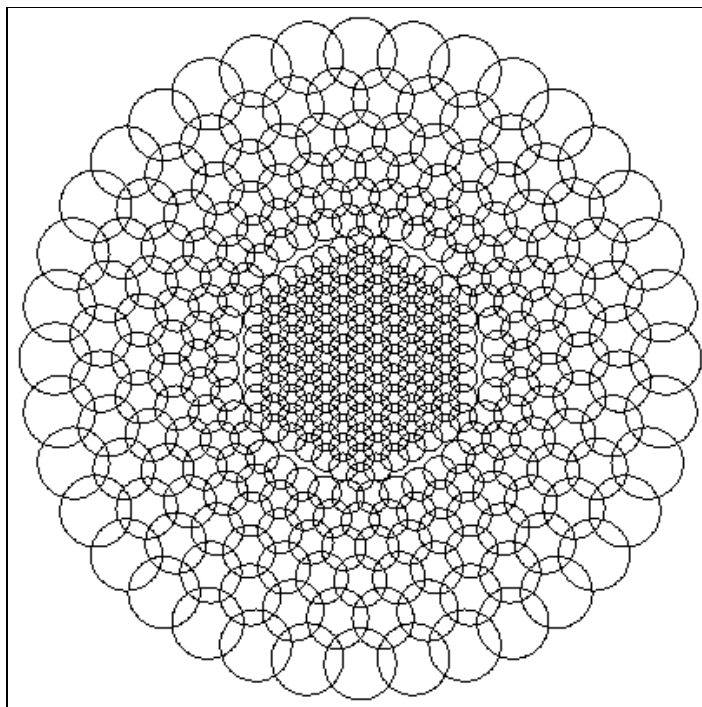


Figure 3.1: Retina structure. In order to enhance details, the retina parameters used to draw this picture are different from those actually chosen for the experiments.

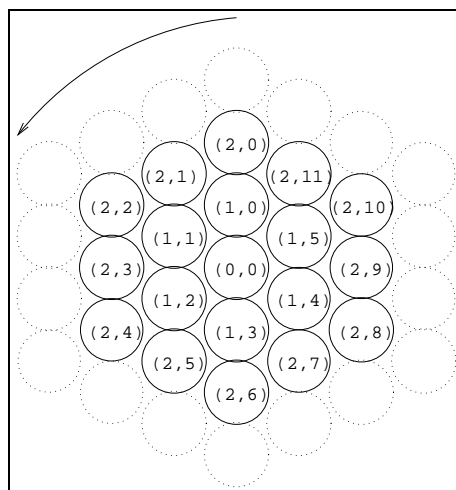


Figure 3.2: Polar coordinates in the fovea.

The radius in the polar coordinate system is represented by an integer indicating the ring. The angle is represented by an integer which indicates the position of a receptive field within a ring. This position is obtained by numbering the receptive fields anti-clockwise from the top middle receptive field (at  $90^\circ$ ). The neighbours of a specific receptive field are named and accessed by using a local polar coordinate system which is centred on that receptive field. Receptive fields in the context of the whole fovea are accessed by using a global polar coordinate system centred on the innermost receptive field of the fovea. For example, in Figure 3.2, the first six local neighbours (in local polar coordinates)  $(1,0)$ ,  $(1,1)$ ,  $(1,2)$ ,  $(1,3)$ ,  $(1,4)$ ,  $(1,5)$  of the receptive field centred at global location  $(1,1)$  are, in the global polar coordinate system:  $(2,1)$ ,  $(2,2)$ ,  $(2,3)$ ,  $(1,2)$ ,  $(0,0)$ ,  $(1,0)$ , respectively (see Figure 3.3).

An auxiliary non-orthogonal coordinate system, which is called here the hexagonal coordinate system, is used for easing the conversion between the local coordinate system and the global one, see Figure 3.3. Note that the figure shows one of the six possible orientations for the coordinate system, which has been chosen arbitrarily.

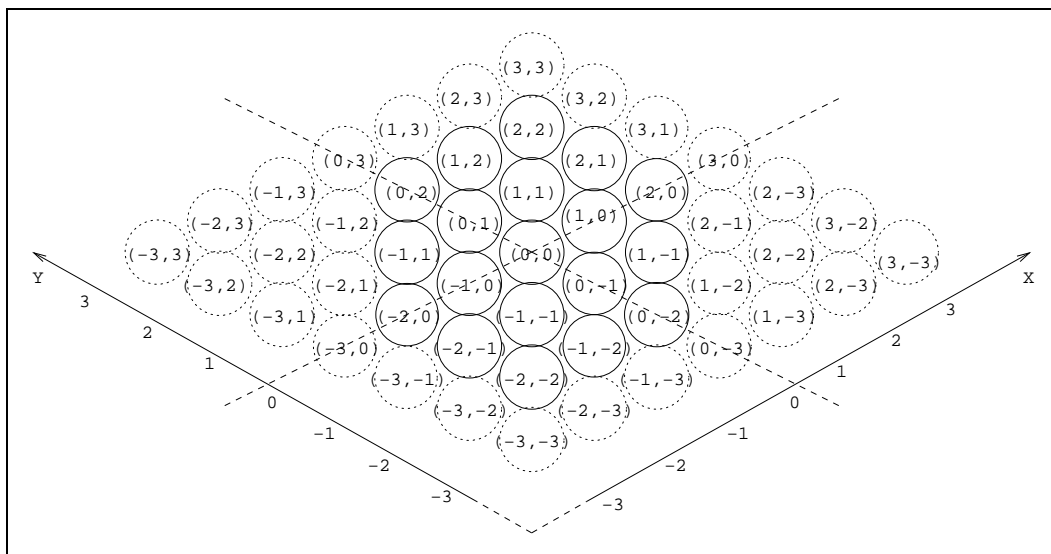


Figure 3.3: Hexagonal coordinate system in the fovea.

The following equation shows how to convert a point  $(n, s)$  in polar coordinates (Figure 3.2) to hexagonal ones (Figure 3.3):



$$\hat{H}(n,s) = \begin{cases} (n-s, n) & \text{if } 0 \leq s < n \\ (n-s, 2n-s) & \text{if } n \leq s < 2n \\ (-n, 2n-s) & \text{if } 2n \leq s < 3n \\ (s-4n, -n) & \text{if } 3n \leq s < 4n \\ (s-4n, s-5n) & \text{if } 4n \leq s < 5n \\ (n, s-5n) & \text{if } 5n \leq s < 6n \\ (0, 0) & \text{if } r = s = 0 \end{cases} \quad (3.1)$$

where  $n$  is an index to the ring and  $s$  is an index to the sector within the fovea.

The inverse transformation  $\hat{H}^{-1}$  is:

$$\hat{H}^{-1}(x,y) = \begin{cases} (y, y-x) & \text{if } 0 < x \leq y \\ (y-x, y-2x) & \text{if } x \leq 0 < y \\ (-x, -2x-y) & \text{if } x < y \leq 0 \\ (-y, -4y+x) & \text{if } y \leq x < 0 \\ (x-y, 5x-4y) & \text{if } y < 0 \leq x \\ (x, y+5x) & \text{if } 0 \leq y < x \\ (0, 0) & \text{if } x = y = 0 \end{cases} \quad (3.2)$$

The next step is to express the neighbours of an arbitrary receptive field in the fovea in terms of the global polar coordinate system. The number of receptive fields in a given ring  $n$  of the fovea is:

$$\hat{F}(n) = \begin{cases} 1 & \text{if } n = 0 \\ 6n & \text{if } 0 < n < \hat{N} \end{cases} \quad (3.3)$$

where  $\hat{N}$  is the total number of rings in the fovea.

The local neighbour  $(n', s')$  of the receptive field  $(n, s)$  can be viewed in global coordinates by converting both coordinates to hexagonal ones, translating the neighbour coordinates by the central receptive field coordinates, and then converting the result back to polar coordinates. Equation (3.4) specifies this operation:

$$\hat{G}(n', s', n, s) = \hat{H}^{-1}(\hat{H}(n', s') + \hat{H}(n, s)) \quad (3.4)$$

where  $0 \leq n' < (\hat{N} - n)$  and  $0 \leq s' < \hat{F}(n')$ .

The mapping from the hexagonal coordinate system to the Cartesian one can be viewed as a rotation of the y axis by an angle of  $\pi/3$  radians and a rotation of the x axis by an angle of  $\pi/6$  radians:

$$\hat{C}(x, y) = (x \cos(\pi/6) - y \sin(\pi/3), x \sin(\pi/6) + y \cos(\pi/3)) \quad (3.5)$$

So far, receptive fields have been considered as points of infinitesimal area in a given coordinate system. In order to complete the fovea structure, it is now necessary to take into account the receptive field radius  $\hat{r}$ , which is constant all over the fovea, and also the percentage overlap ( $\hat{\delta}$ ) with its neighbours. This can be trivially done by introducing a scale factor into Equation (3.5). It is reasonable to think that Equation (3.5) gives coordinates for the receptive field centres. The appropriate displacement between the centres is obtained by scaling up those coordinates by a factor which is equal to the receptive field diameter  $2\hat{r}$ . It is also necessary to scale down the original coordinates by factor of  $(1 - \hat{\delta})$  which is proportional to the overlap  $\hat{\delta}$  along the diameters of two adjacent receptive fields. The adjusted mapping function is shown in Equation (3.6).

$$\hat{C}(x, y) = 2 \hat{r} (1 - \hat{\delta}) (x \cos(\pi/6) - y \sin(\pi/3), x \sin(\pi/6) + y \cos(\pi/3)) \quad (3.6)$$

The last coordinate system conversion that needs to be specified is the one that obtains the Cartesian coordinates of the receptive field centres from their ring and sector indexes. This is easily implemented by a composition of two of the functions already specified. First, the ring and sector inputs are converted to hexagonal coordinates, Equation (3.1). Then, the result is converted to Cartesian coordinates, Equation (3.6):

$$\hat{Q}(n, s) = \hat{C} \circ \hat{H}(n, s) \quad (3.7)$$

Finally, the radius of the ring which passes through the receptive field centres of a given ring  $n$  within the fovea is given by:

$$\hat{R}(n) = n 2 \hat{r} (1 - \hat{\delta}) \quad (3.8)$$

Three input parameters can be identified from the definition of the fovea structure: the radius of the receptive fields  $\hat{r}$ , the total number of rings  $\hat{N}$  and the overlap between adjacent receptive fields  $\hat{\delta}$ . All the fovea parameters are summarised in Table 3.1:

Parameter	Type	Description
$\hat{r}$	input	radius of each receptive field
$\hat{N}$	input	number of concentric hexagonal rings
$\hat{o}$	input	overlap between recep. fields
$\hat{F}(n)$	Eq. (3.3)	number of recep. fields in a given ring $n$
$\hat{G}(n', s', n, s)$	Eq. (3.4)	recep. field neighbour in polar coordinates
$\hat{Q}(n, s)$	Eq. (3.7)	recep. field centre in Cartesian coordinates
$\hat{R}(n)$	Eq. (3.8)	radius of ring $n$

Table 3.1: Fovea parameters

### 3.2.2 Outside the Fovea

Outside the fovea, it is convenient to have receptive fields also organised in an overlapping hexagonal structure which causes neighbours of a receptive field to be uniformly displaced, though the primate retina does not necessarily implement that<sup>1</sup>. As inside the fovea, receptive fields may be viewed as a sequence of concentric rings, differing from the fovea in the sense that here each ring contains the same fixed number of receptive fields  $\bar{F}$ , see Equation (3.9). The radius of the first ring in the outer region  $\bar{R}(0)$  is equal to the last ring radius in the fovea, see Equation (3.10).

$$\bar{F} = \bar{F}(n) = \hat{F}(\hat{N} - 1), \forall n \in 1 \dots \bar{N} \quad (3.9)$$

where  $\bar{N}$  is an input parameter which defines the number of rings outside the fovea.

$$\bar{R}(0) = \hat{R}(\hat{N} - 1) \quad (3.10)$$

A new enumeration from 0 to  $\bar{N}$  is used outside the fovea. The ring  $\hat{N} - 1$ , seen within the fovea, is the same as the ring 0 seen from the out-fovea perspective. In a subsequent section of this text, the notation being used will then be unified to describe the entire retina.

To assure a complete sampling of the field of view, receptive fields overlap by a certain percentage of their diameter with both radial and angular (concentric) neighbours. We assume this overlap will be identical to the one used in the fovea:

<sup>1</sup> Overlapping and hexagonal packing might be valid for *parasol* ganglion neurons in the retina, but not for *midget* neurons, for further details see [Wan95].

$$\bar{o} = \hat{o} \quad (3.11)$$

In contrast to what happens within the fovea, where the receptive field radii are constant, in the outer region the radii increase geometrically. As a result, the radius of a ring  $n$  is a constant  $B > 1$  times the radius of the previous ring.

$$\bar{R}(n) = B\bar{R}(n-1) \quad (3.12)$$

Thus it is possible to express the radius of the  $n_{\text{th}}$  ring in terms of the radius of ring 0 outside the fovea:

$$\begin{aligned} \bar{R}(n) &= B\bar{R}(n-1) \\ &= B(B\bar{R}(n-2)) \\ &= \dots \\ &= B^n \bar{R}(0) \end{aligned} \quad (3.13)$$

Each pair of adjacent receptive fields in a specific ring defines an angle between their centres and the retina centre, which is called by the characteristic angle  $\bar{\theta}$ . This is  $2\pi$  (the full circle) divided by the number of receptive fields in any ring outside the fovea:

$$\bar{\theta} = \frac{2\pi}{\bar{F}} \quad (3.14)$$

A hexagonal packing outside the fovea is obtained by shifting the angle of the  $s_{\text{th}}$  receptive field in ring  $n$  by half of the characteristic angle in all even rings (this disposition is also known as triangular tessellation):

$$\bar{A}(s, n) = s \bar{\theta} + (1 - n\%2) (\bar{\theta}/2) \quad (3.15)$$

where % is the remainder operator.

As far as coordinate systems are concerned, this tessellation is not compatible with the one inside the fovea due to the fact that it doesn't comply with the structure of a central absolute receptive field surrounded by rings of neighbours in multiples of six - there is no such a central receptive field outside the fovea. Therefore, we cannot reuse the neighbouring scheme developed for the fovea. Thus, it is necessary to specify a new function for converting local coordinates to global ones outside the fovea. This can be accomplished more easily with the support of an auxiliary coordinate system in the form of a helix which is shown in Figure 3.4. The first coordinate  $h$  is the number of the segment of spiral and the second,  $n$ , is the same as the radius used in the polar coordinate system.

Initially, it is important to specify how to convert from the global polar coordinate system  $(n, s)$  to the auxiliary one  $(h, n)$  and vice-versa, this is shown in Equations (3.16) and (3.17), respectively. The reason for using the  $CIRC()$  function is to take into account the fact that the axis  $h$  and  $s$  are circular, i.e., after reaching the value of  $\bar{F} - 1$ , they jump back to the value 0.

$$\bar{L}(n, s) = (CIRC(s - INT((n + 1)/2)), n) \quad (3.16)$$

$$\bar{L}^{-1}(h, n) = (n, CIRC(h + INT((n + 1)/2))) \quad (3.17)$$

where the function  $INT()$  truncates its argument downward to the nearest integer. The function  $CIRC()$  is defined as follows:

$$CIRC(x) = (x + \bar{F}) \% \bar{F} \quad (3.18)$$

$\bar{F}$  is added to  $x$  in order to convert some negative coordinates generated by equation 3.16 to positive ones, it has no effect when  $x$  is positive. This conversion is needed because the  $h$ -axis (Figure 3.4) accepts only positive coordinates.

The next step is to specify how to convert local polar coordinates to relative displacements in the auxiliary coordinate system. This is presented in Equation (3.19).

$$\bar{D}(n', s') = \begin{cases} (s', n' - s') & \text{if } 0 \leq s' < n' \\ (n', n' - s') & \text{if } n' \leq s' < 2n' \\ (3n' - s', -n') & \text{if } 2n' \leq s' < 3n' \\ (3n' - s', s' - 4n') & \text{if } 3n' \leq s' < 4n' \\ (-n', s' - 4n') & \text{if } 4n' \leq s' < 5n' \\ (s' - 6n', n') & \text{if } 5n' \leq s' < 6n' \end{cases} \quad (3.19)$$

In order to obtain the neighbour  $(n', s')$  of a receptive field  $(n, s)$  in global coordinates, one needs to convert  $(n, s)$  to the auxiliary coordinate system and  $(n', s')$  to displacements in the auxiliary coordinate system. The final output is the sum of the previous two results converted back to the global polar coordinate system, see Equation (3.20).

$$\bar{G}(n', s', n, s) = \bar{L}^{-1}(\bar{L}(n, s) + \bar{D}(n', s')) \quad (3.20)$$

where  $0 \leq n' < \text{MIN}(n, \bar{N} - n)$  and  $0 \leq s' < \hat{F}(n')$ ;  $\text{MIN}$  returns the minimum value amongst its arguments.

The Cartesian coordinates of a receptive field centre indexed by ring and sector  $(n, s)$  is simply a standard polar to Cartesian transformation of its ring radius and actual angle:

$$\bar{Q}(n, s) = (\bar{R}(n)\cos(s\bar{\theta}), \bar{R}(n)\sin(s\bar{\theta})) \quad (3.21)$$

The final parameter to be calculated is  $B$  which is constrained by the following question: what is the value of  $B$  which assures the same radial and angular (concentric) overlap between adjacent receptive fields? The answer to this question is presented in the following paragraphs.

### Angular overlap

The characteristic angle can be used to specify the angular overlap between two receptive fields. Figure 3.5 shows the distance  $d(n)$  between two adjacent receptive fields of centres  $p1$  and  $p2$ . By means of simple trigonometry  $p1$  and  $p2$  can be expressed as:

$$p1 = (\bar{R}(n)\cos(\theta), \bar{R}(n)\sin(\theta)) \quad (3.22)$$

$$p2 = (\bar{R}(n)\cos(0), \bar{R}(n)\sin(0)) = (\bar{R}(n), 0) \quad (3.23)$$

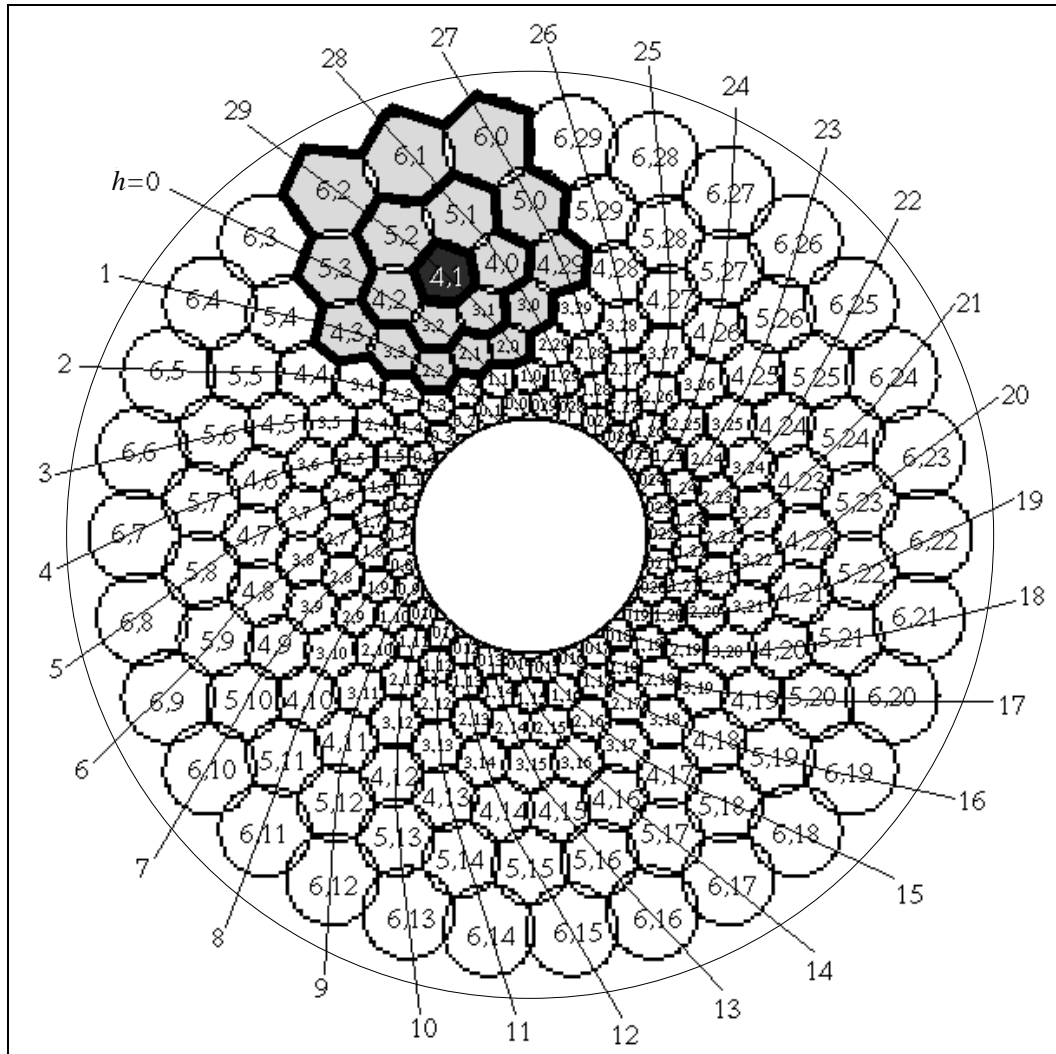


Figure 3.4: Accessing receptive fields outside the fovea. The polar coordinates  $(n,s)$  denoting rings and sectors, respectively, are indicated by the pairs of numbers printed inside each receptive field. The coordinates  $(h,n)$  are used to ease the process of calculating the neighbourhood of a receptive field in polar coordinates,  $n$  denotes the rings, as before, and  $h$  are the numbered lines resembling a helix in the picture. The shadowed receptive fields represent a neighbourhood of radius 2 centred on the point  $(4,1)$ .

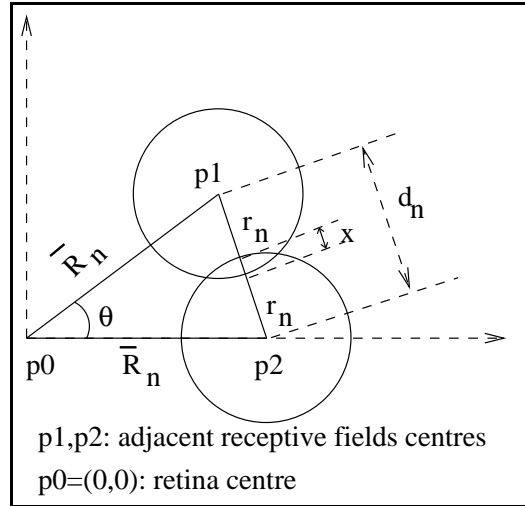


Figure 3.5: Angular (concentric) overlap.

An expression for  $d(n)$  is easily obtained from the distance between two points in the Euclidean space:

$$\begin{aligned}
 d(n) &= \|p1, p2\| \\
 &= \sqrt{(\bar{R}(n)\cos(\theta) - \bar{R}(n))^2 + (\bar{R}(n)\sin(\theta) - 0)^2} \\
 &= \bar{R}(n)\sqrt{2(1 - \cos(\theta))} \\
 &= 2\bar{R}(n)\sin(\theta/2)
 \end{aligned} \tag{3.24}$$

Substituting the value of  $\bar{R}(n)$  above from Equation (3.13) leads to:

$$d(n) = 2 B^n \bar{R}(0) \sin(\theta/2) \tag{3.25}$$

The next step is to calculate the receptive field radius in terms of  $d(n)$  and  $\bar{o}$ . Figure 3.5 shows the parameters defining the angular overlap. The value of  $x$  (the absolute overlap) is defined as the overlap factor  $\bar{o}$  times the receptive field diameter  $2\bar{r}(n)$ :

$$x = \bar{o} (2 \bar{r}(n)) \tag{3.26}$$

Thus, the distance  $d(n)$  between two adjacent receptive fields in a given ring  $n$  is the sum of



the two radii minus the absolute overlap  $x$ :

$$\begin{aligned}
 d(n) &= \bar{r}(n) + \bar{r}(n) - x \\
 &= 2\bar{r}(n) - \bar{o}(2\bar{r}(n)) \\
 &= 2\bar{r}(n)(1 - \bar{o})
 \end{aligned} \tag{3.27}$$

By combining Equations (3.25) and (3.27), it is possible to derive an expression for the receptive field radius in ring  $n$ :

$$\bar{r}(n) = \frac{B^n \bar{R}(0) \sin(\theta/2)}{(1 - \bar{o})} \tag{3.28}$$

### Radial overlap

The radial overlap can be defined in a similar way. Figure 3.6 shows the radial overlap parameters for producing a triangular tessellation. The expressions for  $p1$  and  $p3$  are trivially calculated as:

$$p1 = (\bar{R}(n)\cos(\theta/2), \bar{R}(n)\sin(\theta/2)) \tag{3.29}$$

$$p3 = (\bar{R}(n+1)\cos(0), \bar{R}(n+1)\sin(0)) = (\bar{R}(n+1), 0) \tag{3.30}$$

The distance  $d'(n)$  between the two adjacent receptive fields, with centres in rings  $n$  and  $n+1$ , is again defined as the distance between two points in the Euclidean space:

$$\begin{aligned}
 d'(n) &= \|p1, p2\| \\
 &= \sqrt{(\bar{R}(n)\cos(\theta/2) - \bar{R}(n+1))^2 + (\bar{R}(n)\sin(\theta/2) - 0)^2} \\
 &= \bar{R}(n)\sqrt{B^2 - 2B\cos(\theta/2) + 1}
 \end{aligned} \tag{3.31}$$

In Figure 3.6, the absolute overlap  $x'$  is, by definition, the overlap factor  $\bar{o}$  times the sum of the two receptive field radii.

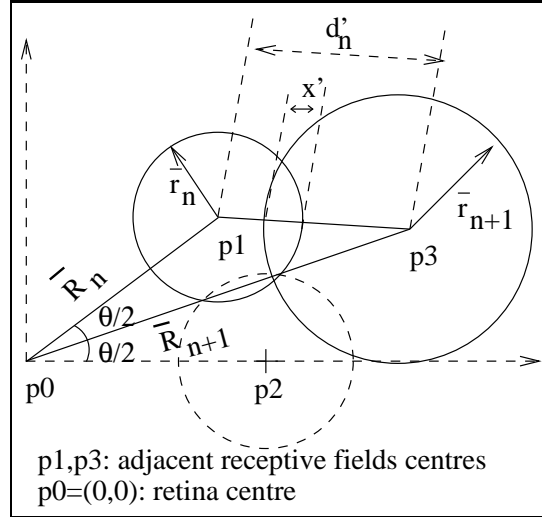


Figure 3.6: Radial overlap.

$$x' = \bar{\delta}(\bar{r}(n) + \bar{r}(n+1)) \quad (3.32)$$

As in the angular overlap calculation,  $d'(n)$  is again the sum of the two radii minus the absolute overlap  $x'$ , see Equation (3.33). The main difference here is that the receptive fields have distinct radii.

$$d'(n) = \bar{r}(n) + \bar{r}(n+1) - x' \quad (3.33)$$

Values for  $\bar{r}(n)$  follow the same geometrical progression of  $\bar{R}(n)$ , therefore from Equation (3.12):

$$\bar{r}(n+1) = B\bar{r}(n) \quad (3.34)$$

By substituting Equations (3.32) and (3.34) into Equation (3.33) it is possible to derive another expression for  $d(n)$ :

$$\begin{aligned} d'(n) &= \bar{r}(n) + B\bar{r}(n) - \bar{\delta}(\bar{r}(n) + \bar{r}(n+1)) \\ &= \bar{r}(n) (1+B)(1-\bar{\delta}) \end{aligned} \quad (3.35)$$

It is possible to derive an expression for  $\bar{r}(n)$  by combining Equations (3.31) and (3.35):

$$\bar{r}(n) = \frac{\bar{R}(n)\sqrt{B^2 - 2B\cos(\theta/2) + 1}}{(1+B)(1-\bar{o})} \quad (3.36)$$

Substituting the value of  $\bar{R}(n)$  from Equation (3.13) leads to:

$$\bar{r}(n) = \frac{B^n \bar{R}(0)\sqrt{B^2 - 2B\cos(\theta/2) + 1}}{(1+B)(1-\bar{o})} \quad (3.37)$$

### Calculating $B$

Finally, we combine the equations for  $\bar{r}(n)$  in both radial and angular overlap formulations, Equations (3.28) and (3.37), respectively:

$$\frac{B^n \bar{R}(0)\sqrt{\sin(\theta/2)}}{(1-\bar{o})} = \frac{B^n \bar{R}(0)\sqrt{B^2 - 2B\cos(\theta/2) + 1}}{(1+B)(1-\bar{o})} \quad (3.38)$$

Solving the above equation for  $B$ , leads to a pair of solutions (a plus  $\oplus$  solution and a minus  $\ominus$  solution):

$$B = \frac{(\sin^2(\theta/2) + \cos(\theta/2) \pm \sin(\theta/2)\sqrt{2\cos(\theta/2) + 1}}{\cos^2(\theta/2)} \quad (3.39)$$

It is easy to conclude from subsection 3.2.1 that the smallest number of receptive fields in the last ring of the fovea is 6, and therefore  $\theta$  is constrained to the range  $0 < \theta \leq \pi/3$ . As a result, the  $\ominus$  solution of the above equation implies  $0.33 \leq B < 1$ , which is inappropriate as  $B < 1$  causes rings of receptive fields to grow inward (Equation (3.13)) and receptive field areas to become smaller (Equation (3.34)). The  $\oplus$  solution implies  $1 < B \leq 2.24$ , which causes rings to be positioned away from the retina centre and receptive field areas to get bigger, which is the desired behaviour.

From Equations (3.39), (3.14) and (3.9), it is clear that  $B$  is a function of the number of rings in the fovea. Thus, there is only one input parameter needed to define the outer retinal region, which is the total number of rings outside the fovea ( $\bar{N}$ ). All the remaining parameters are obtained from the fovea definition. Additional measurements can be easily derived. For

example, the diameter  $D$  of the entire retina, including the radius of the receptive fields in the last ring can be expressed in terms of the already calculated parameters, see Equation (3.40). Table 3.2 summarises the out-of-fovea parameters.

$$D = 2(\bar{R}(\bar{N} - 1) + \bar{r}(\bar{N} - 1)) \quad (3.40)$$

Parameter	Type	Description
$\bar{N}$	input	number of rings outside the fovea
$\bar{F}(n)$	Eq. (3.9)	number of recep. fields in a given ring $n$
$\bar{R}(0)$	Eq. (3.10)	radius of the first out-fovea ring
$\bar{o}$	Eq. (3.11)	overlap equals to fovea overlap
$\bar{R}(n)$	Eq. (3.13)	radius of a given ring $n$
$\bar{r}(n)$	Eq. (3.34)	recep. field radius of a given ring $n$
$B$	Eq. (3.39)	exponential base for recep. fields sizes
$\bar{\theta}$	Eq. (3.14)	characteristic angle
$\bar{A}(s, n)$	Eq. (3.15)	angle of recep. field in sector $s$ of the ring $n$
$\bar{G}(n', s', n, s)$	Eq. (3.20)	recep. field neighbour in polar coordinates
$\bar{Q}(n, s)$	Eq. (3.21)	recep. field centre in Cartesian coordinates

Table 3.2: Out-fovea parameters.

### 3.2.3 The Entire Retina

A unified notation for the entire retina is provided here. The number of receptive fields in a given ring  $n$ , from Equations (3.3) and (3.9), is now:

$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ 6n & \text{if } n < \hat{N} \\ F(\hat{N} - 1) & \text{otherwise} \end{cases} \quad (3.41)$$

As the overlap factors  $\hat{o}$  and  $\bar{o}$  are the same for the entire retina, they receive the same denomination  $o$ . From Equations (3.8), (3.10) and (3.13), the radius of a given ring  $n$  is:

$$R(n) = \begin{cases} 2 \hat{r} n (1 - o) & \text{if } n < \hat{N} \\ B^{n-\hat{N}} R(\hat{N} - 1) & \text{otherwise} \end{cases} \quad (3.42)$$

From combining Equations (3.7) and (3.21), the receptive field centre in Cartesian coordinates can be expressed as:

$$Q(n, s) = \begin{cases} \hat{C} \circ \hat{H}(n, s) & \text{if } n < \hat{N} \\ (R(n) \cos(s\bar{\theta}), R(n) \sin(s\bar{\theta})) & \text{otherwise} \end{cases} \quad (3.43)$$

From Equation (3.34) and Table 3.1, the receptive field radius of a given ring  $n$  is:

$$r(n) = \begin{cases} \hat{r} & \text{if } n < \hat{N} \\ r(n-1) B & \text{otherwise} \end{cases} \quad (3.44)$$

Equations (3.4) and (3.20) can be unified as follows:

$$G(n', s', n, s) = \begin{cases} \hat{H}^{-1}(\hat{H}(n', s') + \hat{H}(n, s)) & \text{if } n < \hat{N} \\ \bar{L}^{-1}(\bar{L}(n, s) + \bar{D}(n', s')) & \text{otherwise} \end{cases} \quad (3.45)$$

It is more intuitive to have an input parameter defining the total number of rings present in the retina. Lets call this parameter by  $N$ , and discard the older  $\bar{N}$ , which defines the number of rings in the outer retina region. The following equation shows the relationship  $N$  and the previous parameter  $\bar{N}$  (the -1 value in the equation is to take in account the fact that the first ring outside the fovea is equal to the last ring inside the fovea):

$$N = \bar{N} + \hat{N} - 1 \quad (3.46)$$

Finally, Table 3.3 summarises the unified retinal notation.

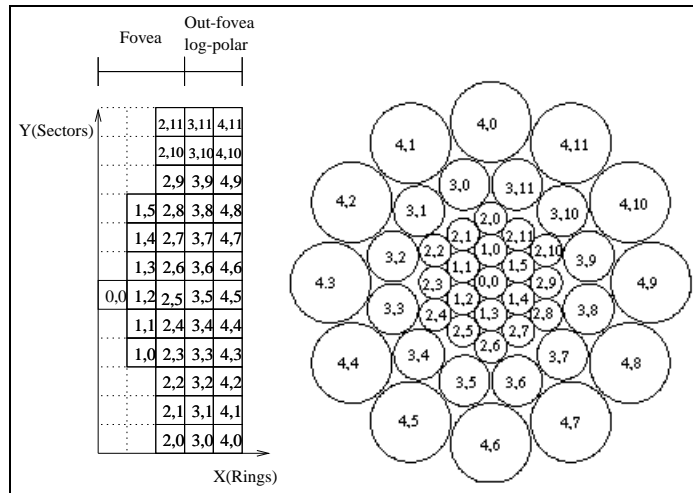
Parameter	Type	Description
$\hat{r}$	input	fixed radius of each foveal receptive field
$\hat{N}$	input	number of concentric rings in the fovea
$N$	input	number of rings of the entire retina
$o$	input	overlap between receptive fields
$r(n)$	Eq. (3.44)	receptive field radius of a given ring $n$
$F(n)$	Eq. (3.41)	number of receptive fields in a given ring $n$
$R(n)$	Eq. (3.42)	radius of a given ring $n$
$Q(n, s)$	Eq. (3.43)	recep. field centre in Cartesian coordinates
$G(n', s', n, s)$	Eq. (3.45)	recep. field neighbour in polar coordinates
$B$	Eq. (3.39)	exponential base for the receptive field size
$D$	Eq. (3.40)	retina diameter

Table 3.3: Unified retinal parameters.

### 3.2.4 Log-Polar Representation

A log-polar representation can be seen as a Cartesian image whose horizontal axis (log) represents the logarithm of the distance of a receptive field centre from the retina centre, the vertical axis (polar) represents the receptive field angular displacement. Thus, a rotation equivalent to  $k$  entire sectors in the retinal space is equivalent to a translation of  $k$  units in the polar axis (1 unit denotes the angle between any two adjacent sectors), whereas a change in scale of  $B^p$  in the retinal space is equivalent to a translation of  $p$  units in the log axis (1 unit denotes  $\log(B)$ ). As discussed in Chapter 2, a log-polar representation can be regarded as a mathematical approximation to the mapping from receptive fields in the mammalian retina to neurons in the visual cortex (also known as the retino-cortical transform) [Sch77, Wil83].

Figure 3.7 shows the mapping of a 5 ring retina containing a 3 ring fovea.



## 3.3 Functionality

In this section the functional aspects of the representation are discussed. Firstly, the formulation of the receptive field function is presented. Then, a method for estimating the local reflectance information from images is proposed. The section ends with the mapping from Cartesian to retinal images and vice-versa.

### 3.3.1 Receptive Field Function

There has been much study about the mammalian retinal ganglion cells [Tro93, EC93]. In general, these cells compute some kind of integration over circular areas (or receptive fields) of photoreceptors. In our representation, the ganglion computation is approximated by the following equation:

$$O((\bar{x}, \bar{y}), r) = \sum_{(x-\bar{x})^2+(y-\bar{y})^2 \leq r^2} I(x, y) F(x - \bar{x}, y - \bar{y}) \quad (3.47)$$

where  $O$  is the output for the receptive field of radius  $r$  centred at the point  $(\bar{x}, \bar{y})$  of the input image,  $I$  is the perceived intensity and  $F$  is the receptive field function.

One of the simplest functions that can be used in the computation of the receptive values is the uniform averaging function [GF96]. A more biologically plausible alternative, which is investigated in this work, is the normalised Gaussian function, Equation (3.48), whose integral is approximately equal to 1 when integrated over a circular area of radius  $r = 3\sigma$ :

$$F(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.48)$$

### 3.3.2 Providing Support for Estimating the Reflectance Information

The human visual system computes an approximate colour constancy, and this implies that the visual system is somehow able to extract information about the invariant surface reflectance of objects (almost) independently of changes in illumination and scene composition [Hur92]. It is not yet clear to what extent this mechanism depends on processes that take place within single photo receptors and to what extent it depends on processes associated with the spatial interaction within the retinal network [SKP93].

A method used to help estimate the original reflectance information of objects is proposed here. It is based on the hypothesis that light adaptation occurs at the sensor level. In the simple situation where a planar scene is only composed of a group of patches of constant reflectivity (Mondrian world), the scene radiance (or brightness  $I(x, y)$ ) at a particular point is proportional to the product of the irradiance falling on the object ( $E(x, y)$ ) and the reflectance of the surface

$(R(x,y))$ :

$$I(x,y) = E(x,y)R(x,y) \quad (3.49)$$

Each receptive field in the retina can be viewed as a locally planar representation of the three-dimensional real world. By assuming the image projected into each receptive field is from a Mondrian scene, it is possible to estimate the original reflectance information from the surface segments represented in each receptive field [HB89]. The method commonly used for extracting the reflectance information from Mondrian images is based on the hypothesis that sharp changes in intensity of the images correspond to surface reflectance changes, while smoother changes in intensity correspond to illumination changes. The goal is to develop a modified method for extracting the reflectance which takes advantage of some peculiarities of the image representation adopted.

Assuming the above, it is possible to approximately isolate the logarithm of the reflectance information by taking the logarithm of the intensities in the integration process carried out in the receptive fields. Equation (3.51) shows the resulting receptive field computation. The  $\log(E)$  term in Equation (3.51) is nearly constant over local image regions and then makes the receptive field computation  $O'$  a good approximation for the weighted logarithm of the reflectance information. The derivation of this result is detailed below.

Applying the logarithm function to the Equation (3.49) leads to:

$$\begin{aligned} \log(I(x,y)) &= \log(E(x,y)R(x,y)) \\ &= \log(E(x,y)) + \log(R(x,y)) \end{aligned} \quad (3.50)$$

Then substituting Equation (3.50) in (3.47), and assuming that:

- $E = E(x,y)$  is nearly constant all over the receptive field area, and
- $F(x,y)$  is the Gaussian function with  $\sigma = \frac{r}{3}$ , which has approximately unit integral in the circular area of radius  $r$ ,

we have:



$$\begin{aligned}
O((\bar{x}, \bar{y}), r) &= \sum_{(x-\bar{x})^2+(y-\bar{y})^2 \leq r^2} (\log(E(x,y)) + \log(R(x,y))) F(x-\bar{x}, y-\bar{y}) \\
&\cong \log(E) + \sum_{(x-\bar{x})^2+(y-\bar{y})^2 \leq r^2} \log(R(x,y)) F(x-\bar{x}, y-\bar{y}) \quad (3.51)
\end{aligned}$$

which represents a reasonable approximation for the logarithm of the reflectance plus a slowly varying constant within the receptive field area. Since the feature extraction operators described in the next section have 1) linear pre-processing in the initial projection stage and 2) the projection weights sum to approximately zero, then the projection of the  $\log(E)$  terms in a feature neighbourhood will also be approximately zero. Thus the features will be detected primarily based on the reflectance structure present in the neighbourhood.

The function  $I(x,y)$  is quantised within the range 0..255, so it is necessary to deal with the singularity of the logarithm function at the pixels of intensity zero,  $\log(0) = -\infty$ . We simply assume  $\log(0) = \log(1) = 0$ , which has the only consequence of making the intensities 0 and 1 indistinguishable. When  $I(x,y)$  is zero, from Equation (3.49), it is impossible to determine whether the irradiance  $E(x,y)$  or the reflectance  $R(x,y)$  is zero, or both, therefore this point is not important in our computations. In order to normalise the output of the logarithm function to the range 0..255 we use the following function:

$$I'(x,y) = 255 \frac{\log(I(x,y))}{\log(255)} \quad (3.52)$$

### 3.3.3 Mapping from Cartesian to Retinal Images (Foveation)

In order to allow for maximum flexibility during the experimental phase of the thesis, the image representation discussed in the previous sections was implemented in software. Nevertheless, there would be no serious technological barriers in deriving a hardware implementation from it in the future (take for instance the CCD implementation found in [dSKC<sup>+</sup>89], and the newer CMOS one found in [Par94]).

It is assumed for the purposes of this thesis that images are acquired by means of the traditional sensor architecture, e.g., using a CCD camera. This implies that there has to be a way of converting input images into retina-like ones. The following equation shows how to compute

the value of a receptive field ( $V(n, s)$ ) given a pair of indexes  $n$  and  $s$  for their ring and sector, so that the entire field of view can be converted by just varying the indexes within the appropriate ranges:

$$V(n, s) = O(Q(n, s), r(n)) \quad 0 \leq n < N, \quad 0 \leq s < F(n) \quad (3.53)$$

where function  $O$  is defined in Equation (3.51).

### 3.3.4 Mapping from Retinal to Cartesian Images (Defoveation)

Because in Equation (3.53) many input Cartesian intensities under a receptive field are mapped into a single value, this function is not mathematically invertible. An inverse transform would be very useful for visualising results and also for producing data suitable for an attention mechanism. One way to get around the problem is by replicating each retinal pixel all over its corresponding reconstructed receptive field area:

$$W(x, y) = V(n, s) \quad (3.54)$$

where:  $0 \leq n < N, 0 \leq s < F(n)$  and  $(x - Q_x(n, s))^2 + (y - Q_y(n, s))^2 \leq r(n)^2$ .

One problem with this approach is that because of the receptive field overlapping, a pixel in the Cartesian space could be assigned a value more than once. A straightforward solution to this problem is to average any overlapping pixels. A more elaborate solution is to use the intersection points between the receptive field boundaries as the vertices for hexagon shaped pixels, so that no overlapping occurs. Figure 3.4 gives an idea on how these hexagonal pixels can be derived from the retina structure.

Finally, if one needs to use defoveated images to create an interest map of the scene, the previous two approaches might not be appropriate on their own. An interest map is essentially a Cartesian image, of the same size of the input scene, that indicates the most interesting regions to be analysed: these regions are indicated by the high valued pixels within the map. An interest map created using Equation (3.54) would have too many pixels with the same high values (as the equation uses a pixel replication strategy), therefore blurring the choice for a region to be attended. In order to minimise this problem, instead of replicating pixels all over

a receptive field, a bell shaped function (like a Gaussian function) aligned with the receptive field centre can be used to weight pixels so that their values decay as they move toward the receptive field periphery.

### 3.4 Retina Parameters and Example

This section presents an input image re-sampled through the use of a retina structure defined as before (see Figure 3.8). For this example and for the rest of the experiments described in this thesis, the fovea was defined as having 11 rings of receptive fields, with each receptive field having a radius of 0.5 of a pixel. The entire retina has 48 rings of receptive fields distributed according to Equation (3.42). Each receptive field overlaps with any of its neighbours by approximately 60.4% of its diameter. These parameters produce a  $B$  of approximately 1.095 and the diameter of the retinal mask is about 256 pixels covering a circular region of the input Cartesian image. The values of the receptive field overlap and  $B$  have both some biological plausibility (see [Wil83], [Sch77] and [RS90] for some insights), whereas the other parameters were chosen for the mere convenience of having a Cartesian coverage of 256 pixels diameter.

The fovea covers an area of approximately 9 pixels in diameter, which corresponds to about 0.12% of the whole retinal area. Since the area of the fovea we defined is very small when compared to area of the entire retina, we would like to briefly comment on our choice to explicitly model the fovea. When designing our image representation we had two options: 1) either to explicitly model the fovea as a uniform sampling grid like, for instance, [ST92] and [SL97] did; or 2) just assume that the receptive fields within the central area would be of approximately the same size due to the slowly varying exponential progression close to the origin, like in [GF96], [LWV97], [FCS98] and [FSC97], for example.

We decided for the first option, since we wanted our representation to be as generic as possible. At any time, we can get to the second representation by just choosing a sufficiently small receptive field radius or lower the number of rings in the fovea, as we actually did during the experiments, when choosing a very small fovea area. Since hexagonal packing happens both inside and outside the fovea, the primal sketch feature extraction operators described in Chapter 4 can be indistinguishably used in both regions, apart from the 2 rings located at the transition between the fovea and the outer region, where the hexagonal geometry breaks.

Receptive fields in this transition area always receive zeroes (lowest intensity) by the feature extraction operators, and therefore will cause no harm neither to an attention mechanism nor to a similarity function used to compare a pair of retinal images.

However, the log-polar translation properties for scale and orientation do not hold in the fovea, so special functions need to be defined to rotate and scale the sub-image within the fovea, if one needs to do object recognition that is invariant to planar scale and rotation. Instead of implementing these functions, we simply defined a fovea small enough that could be discarded. Since the use of a uniform sampling fovea was not crucial to the experiments we describe in the following two chapters, we left the use of the fovea for scale and rotation invariant matching as a future work.

### 3.5 Summary

This chapter presented a mathematical specification for the image representation used as basis for the vision system considered in this thesis. Structural and functional details were included in the specification. The new image representation, which is inspired by the mammalian retina, brought a number of improvements with respect to a previous one described in [GF96]:

1. only a few input parameters (four) are needed to specify the entire retina, all the other parameters are derived in a coherent way from the input ones (see Table 3.3);
2. uniform density fovea and log-polar exterior integrated into a single structure, with the flexibility of using just the fovea, just the log-polar exterior, or both, depending on the chosen parameters.
3. richer receptive field function;
4. support for estimating the reflectance information; and
5. a concise and elegant algorithm for the bi-directional mapping between Cartesian and retinal images.

The work presented in this chapter does not bring any particularly important contribution into the field, being mainly related to research previously published by [RS90], [ST92],

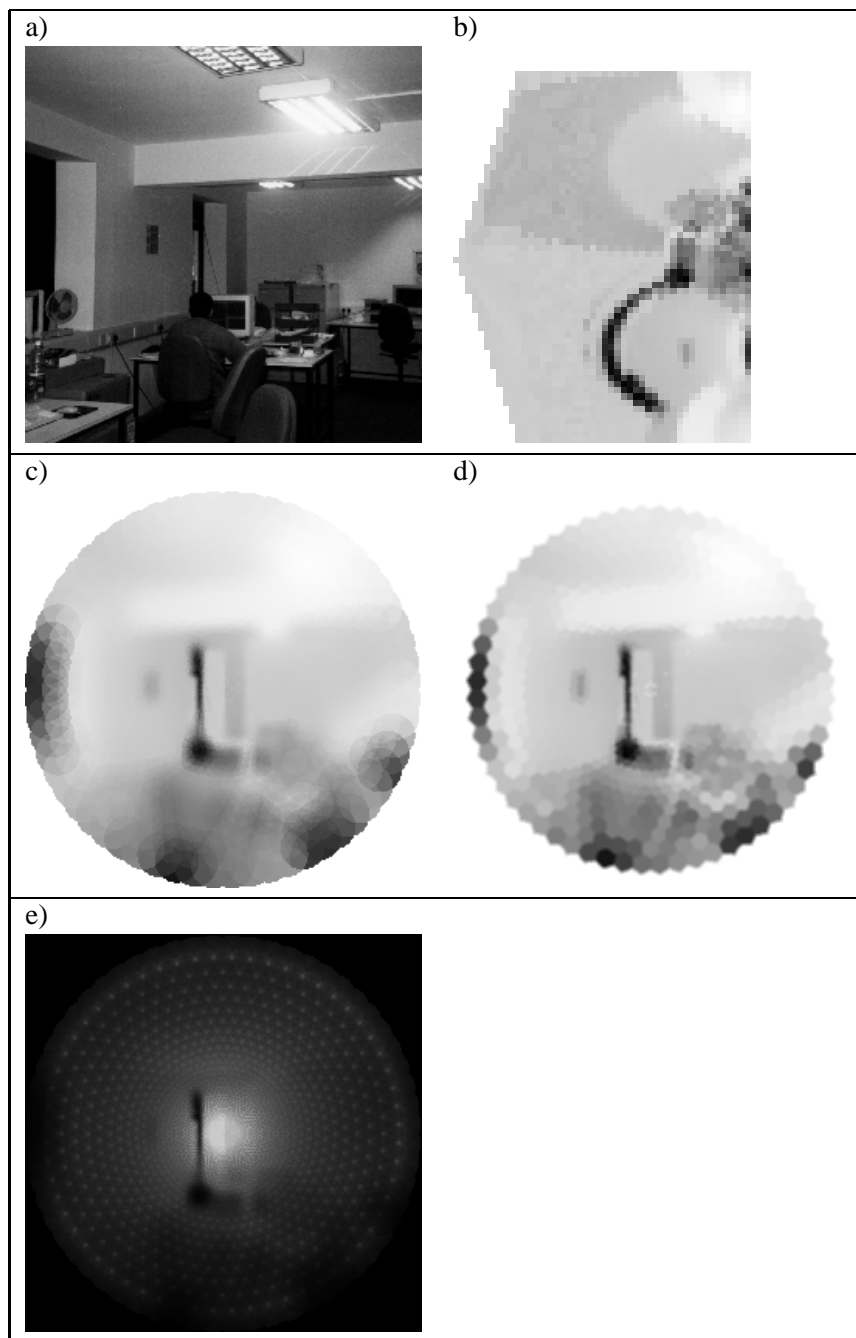


Figure 3.8: An example of applying the retinal representation to a real image: a) input image; b) log-polar image, magnified 4 times; c) reconstructed image using averaging only; d) reconstructed image using hexagonal pixels; e) reconstructed image using bell shaped weighting function. The intensities in images b) to e) are quite different from those in a) because of the logarithmic processing of the intensities.

[GF96], [LWV97] and [Jur99]. The main purpose of this chapter was to develop an image representation that was suitable for the most relevant components of the research presented in this thesis, which are discussed in the following two chapters.

Some factors motivated the choice for a biologically inspired representation in the log-polar form. One factor is the inherent reduced spatial complexity and log-polar property of the retinal image, which favours the implementation of faster matching algorithms. Another factor is that the space-variant nature of this kind of image can lead to a more robust matching process when comparing to conventional uniformly sampled images. Because of the use of an attention mechanism, objects or parts of objects are more likely to be found in the high resolution centre. The low resolution periphery, which occupies only a small number of pixels, will usually include parts of the background that are not so important for matching.

An important reason for the log and polar elements of our representation is the property that scaling and rotating an object located at about the centre of the retinal mask corresponds to translating the object in the log-polar image (Section 2.1.2 contains a more detailed explanation). This has been used to design systems that are scale and rotation invariant [ST92]. In this thesis, we specially took advantage of this property when creating visual models containing relative scale and orientation measurements for all pairs of the models' internal objects (see Section 5.1.4 for more details). The exponential scaling of receptive fields and their corresponding log projection in the visual cortex also helps achieve size constancy [TB95], which is a phenomenon present in animals with foveas, resulting in objects tending to appear constant in size regardless of their distance to the animal.

Log-polar images have also proved to be a representation that has some advantages over the traditional Cartesian sensor geometry in problems like optic flow estimation [DK95], time-to-contact calculation [ST92], robust object tracking [ONM00] and vergence control [PCS95, BSV96]. For instance, Bernardino and Santos-Victor [BSV96] and Panerai et al [PCS95] have shown that the correlation index for Cartesian images under a varying vergence angle exhibits several local minima and a non-uniform behaviour, whereas, for log-polar images, the curve is much smoother and has a sharp minimum in the vicinity of the correct vergence angle. In addition to the above, Oshiro and colleagues [ONM00] have shown that a zero disparity filter yields best results at reduced resolution and that a log-polar mapping of the image plane produces a more complete horopter map (region of zero disparity around the

observer) as compared to images using sub-sampling data reduction.

A final remark with regards to the choice of this particular image representation is related to the question of whether or not a hardware implementation would be required for fast or real-time image acquisition. Although there have been some previous works on VLSI and CMOS implementations of log-polar sensors (e.g. [dSKC<sup>+</sup>89] and [Par94]), a major drawback of these approaches is that, besides being more financially expensive than a software implementation, the sensor parameters are fixed and therefore the physical sensor design necessarily remain fixed. Despite the fact that the sequential software implementation used in this thesis has an inherent decreasing performance as the input image size increases, it does however provide a highly flexible transformation since the sensor parameters can be varied at will. Moreover, the limitations of a software sensor to process large scenes are gradually being overcome by faster computer hardware and software parallelisation techniques.

Although the image representation discussed in this chapter has some interesting properties, more has to be done if one needs a system capable of automatically learning models of objects from a number of generic scenes. More specifically, only colour or intensity information are not enough to build such models as this would imply a large conceptual gap between data and models and, therefore, this would complicate the design of algorithms. Both Machine Vision and Biology have proved that extracting properties and features from images, by means of building intermediate representations, is a useful way of reducing this conceptual gap. The following chapter shows a method for extracting a number of primal sketch features in a retinal representation, which uses a neural network to learn examples of the features. Primal sketch features are believed to be found in the human intermediate visual system, where they provide more compact descriptions for the important aspects of the image data and cues for an attention mechanism.

## Chapter 4

# Primal Sketch Feature Extraction

The raw primal sketch is a very rich description of an image, since it contains virtually all the information . . . Its importance is that it is the first representation derived from an image whose primitives have a high probability of reflecting physical reality directly.  
*D. Marr* [Mar82]

### Contents

---

<b>4.1 Motivations</b> . . . . .	<b>65</b>
<b>4.2 A Previous Approach</b> . . . . .	<b>65</b>
<b>4.3 Proposed Approach</b> . . . . .	<b>66</b>
<b>4.4 Training and Evaluation</b> . . . . .	<b>80</b>
<b>4.5 Concluding Remarks</b> . . . . .	<b>94</b>

---

This chapter presents a novel and more successful learning based approach to extracting low level features in a retina-like (log-polar) image representation. The low level features (*edges*, *bars*, *blobs* and *ends*) are based on Marr's primal sketch hypothesis for the human visual system [Mar82]. The feature extraction process was implemented using a neural network that learns examples of the features in a window of receptive fields of the image representation. An architecture designed to encode the feature's class, position, orientation and contrast has been proposed and tested. Success depended on the incorporation of a function that normalises the feature's orientation and a PCA pre-processing module to produce better separation in the feature space. A bootstrapping strategy that uses synthetic and real features has been used for the learning process. Training and performance evaluation are also discussed in this chapter.



## 4.1 Motivations

Traditional image feature extraction operators have usually been designed by hand, work independently of each other and act on Cartesian images (an artifact of sensor architecture). However, the architecture of the primate vision system seems to be quite different, and we can use this to produce interesting results in artificial vision systems.

From the log-polar image representation developed in the previous chapter, a number of raw primal sketch features are extracted using a different approach. The features (*edges*, *bars*, *blobs* and *ends*) are based on Marr's primal sketch hypothesis for the human visual system [Mar82]. The primal sketch represents a more compact representation for the image data and provides cues for an attention mechanism under the experimental evidence that these kinds of low level features seem to attract visual attention [Yar67].

Instead of trying to manually build a model for completely describing the features, which might be a complex task because of the unusual geometry of the image data and the receptive field integration, learning the features was a sensible option. In this thesis, a neural network approach was chosen due to its adequacy when learning data in which there is no obvious symbolic representation.

In Chapter 5, these features, together with the original log-polar image information, are used during the matching between pairs of image regions taken from a set of scenes with the purpose of automatically building geometric models of the objects found in the scenes.

## 4.2 A Previous Approach

As discussed in Chapter 2, in the system described in [GF96] operators were manually defined as expressions involving the pixels of a receptive field window that is applied throughout the log-polar image. Figure 4.1 illustrates the operators designed to detect *blobs* and *edges* using a triangular retinal tessellation, where the mask is a 1-ring window of 7 receptive fields.

Since the operators' outputs are continuous, any output below a given empirical positive threshold was set to zero to indicate the absence of a feature pattern. A fixed threshold equal to 8 was used for all operators, provided that image intensities were within [0...255].

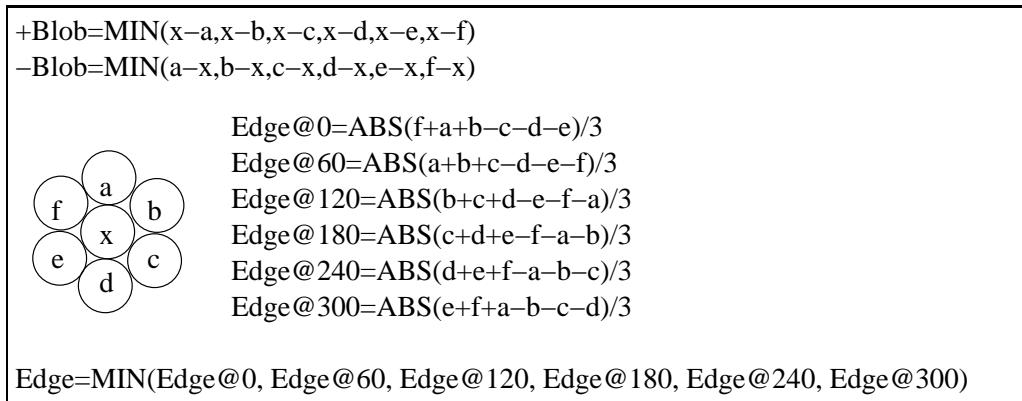


Figure 4.1: The mask used in Grove and Fisher’s system [GF96] to detect features. Each pixel in the mask corresponds to a particular receptive field output in the polar coordinate system. Detectors for *blobs* and *edges* are also shown in the picture.

One of the problems with the above approach is that operators are heuristically defined and then there is no guarantee that they will work correctly with all the possible cases and that they will allow graceful degradation. Also, if a different window size or window shape was needed, it would be necessary to design new logical expressions for the operators which can lead to mistakes as the operators are defined by hand.

### 4.3 Proposed Approach

Features are trained and detected in a window of receptive fields. We have chosen a window composed of a central receptive field plus its next 6 and 12 surrounding neighbours, totalling 19 receptive fields hexagonally distributed.

When centred within these windows, the oriented features (*edges*, *bars* and *ends*) can appear at several distinct orientations. As a result of the receptive field window structure, we have decided to detect *edges* and *ends* at 12 possible orientations and *bars* at 6 possible orientations. Since *bars* are indistinguishable by end direction, they have the same angular resolution as the *edges* and *ends*.

For training purposes, synthetic exemplars of the features are drawn in a fixed position on the input image corresponding roughly to a particular window of receptive fields. Then, the output of these 19 receptive fields is processed and used as input to the neural network classifiers.

As the receptive fields have variable sizes throughout the retinal image, one might think that different trainings would be needed for each scale and orientation of the receptive field windows. However, the receptive field computation produces normalised values as it applies a function whose integral is always 1 independently of the area. That is, if two different magnifications of the same image patch fall into two receptive fields of corresponding scales, the output of both receptive fields will be approximately the same.

Moreover, oriented features like *edges*, *bars* and *ends* will not suffer significant distortions when appearing at different orientations, since changes in scale of the receptive fields within a local neighbourhood are small (typically 9.5% of the radius from one ring to another). We use a symmetry operator to normalise features into a standard orientation, which is detailed in Section 4.3.2.

Although all images processed in this chapter are in a log-polar form, for clarity it is helpful to view the resulting images in a reconstructed Cartesian image when explaining processes and presenting results.

### 4.3.1 Feature Detector

Figure 4.2 shows the main data and processes of the feature detector and how they are linked together. The overall architecture is discussed here and details of the individual processes are given in subsequent sections. The first step is to normalise the feature orientation (see Figure 4.2.a and Section 4.3.2). Then, principal components are computed from a training set (counter-examples are not taken into account), see Figure 4.2.a and Section 4.3.3. Finally, only a subset of the principal components is chosen. This selection consists in choosing the *eigenvectors* associated to the highest *eigenvalues*. This is done for each of the 7 feature classes: *edge*,  $\pm$  *bar*,  $\pm$  *blob*,  $\pm$  *end* (a + sign is assigned to features that have a darker background and a – sign to features that have a brighter background, Section 4.4.1 details this separation).

The next step is to project exemplars of features onto the previously selected subset of *eigenvectors* and use this information as training inputs to neural network modules (see Figure 4.2.b and Section 4.3.4). The desired outputs for the neural networks are encoded from the feature's contrast information (see Section 4.3.5).

In order to extract features from a real image, a process similar to the training one is implemented with the difference that the feature's projection is fed into a set of trained networks and a classification rule is used to interpret the network outputs and build up the feature planes (see Figure 4.2.c and Section 4.3.6).

The last step (see Figure 4.2.d and Section 4.4.4) is to improve the feature sets by selecting incorrectly classified features from real images, which gives the 'fine tuning' aspect of the approach. The steps shown in Figures 4.2.b through 4.2.d are repeated until a satisfactory classification is achieved over a set of test images (see Figure 4.2.e). We iterate from step 4.2.b instead of from step 4.2.a because we assume the principal components are stable, and will not have considerable changes when increasing the size of the training sets.

### 4.3.2 Normalising the Feature Orientation

If a receptive field window could be normalised into a standard orientation before applying the PCA technique (explained in section 4.3.3), the problem could be simplified because now we would end up with a smaller set of principal components related only to the normalised orientation.

This normalisation can be achieved through the use of a symmetry operator. Such operators are well known in the literature of 3-D vision [Tho96]. The ones in the category of planar symmetry from a single view could well be applied, with some adaptations, to our problem. For instance, [Fle90] finds the symmetry of elongated regions by grouping pairs of edge points tangent to a common circle. Differently, [STZ89] used the theory of wave propagation and a diffusion process to find symmetry sets. In order to use the above techniques in our problem, as all work on boundary images, we would have to solve the impossible problem of defining the boundaries of a feature class in a tiny receptive field window.

Another option would be to use moments [PR92], which, amongst other applications, can be employed to calculate the orientations of the main axes of a binary or grey level image.

On the other hand, our problem is simpler than the above ones in the sense that: (i) we don't want to normalise generic shapes, (ii) the receptive field windows have only a small fixed set of pixels, and (iii) the interesting features are assumed to be centred at the receptive field window. Thus, we decided to develop a customised solution. Our symmetry operator was defined as

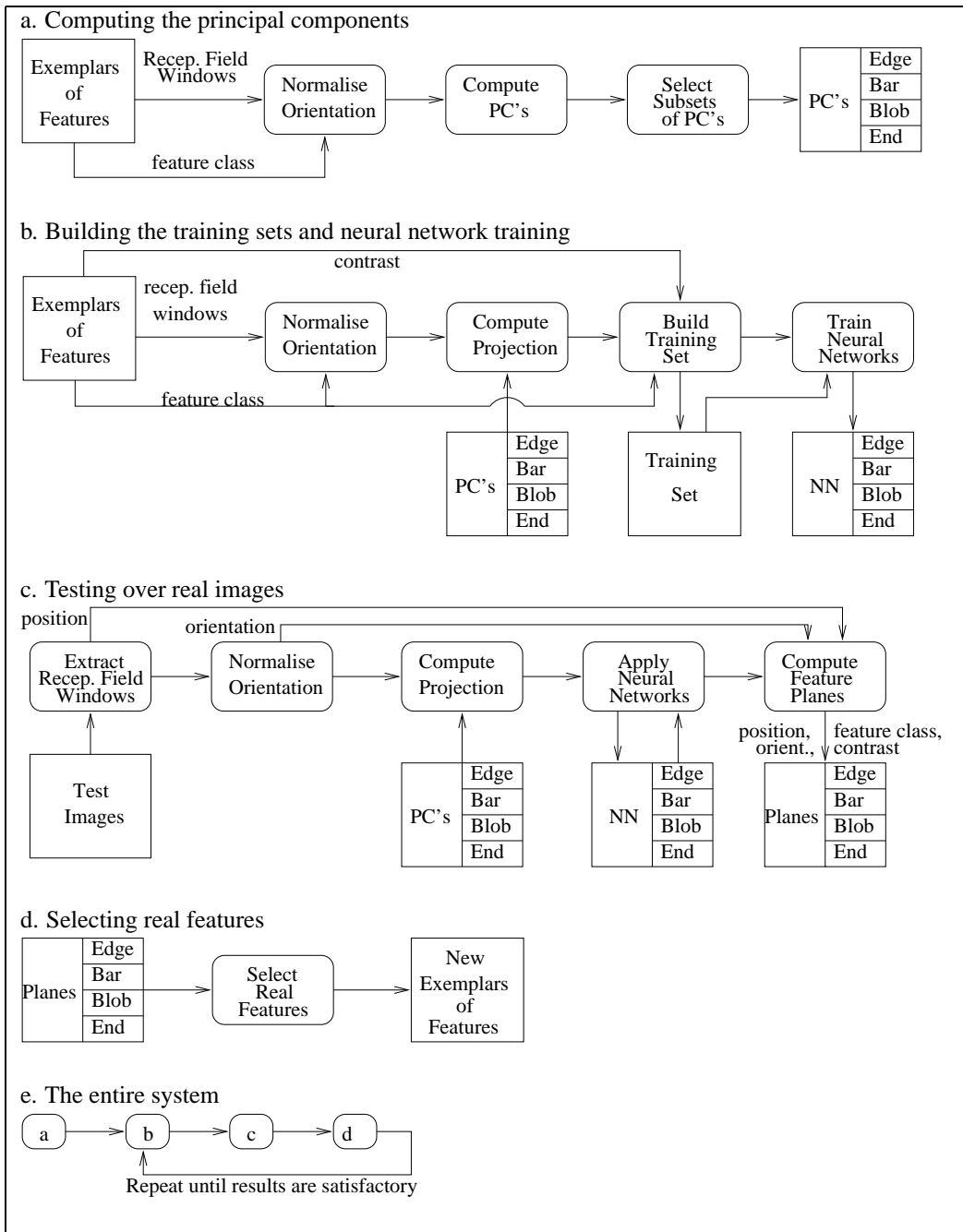


Figure 4.2: Overview of the main processes (rounded boxes) and data (rectangular boxes) of the system used for extracting primal sketch features. See main text for details.

a gradient operator by associating negative weights to a subset of the receptive fields in the retinal window, and positive weights to the remaining receptive fields. By iteratively rotating the symmetry operator with respect to the central receptive field and applying it to a receptive field window, the detected orientation will be the one which maximises the absolute value of convolution. The last step consists of rotating the feature to a standard orientation. The operator is applied at the 12 orientations defined by the receptive fields in the outer ring of the window.

We tried to design a generic operator that would be suitable for all the feature classes, however this was not possible. For instance, an operator which has the receptive fields in the half circle of the retinal window associated to negative weights and the other half to positive ones will be very suitable to detect the orientation of *edge* features, but it will be inappropriate to detect *bars* or *ends*. So we decided to implement a different symmetry operator for each of the oriented features. Figure 4.3 shows the details. The white circles in the figure represent a weight of  $\frac{1}{N}$ , and the darker ones represent a weight of  $-\frac{1}{M}$ , where  $N$  and  $M$  are the number of white and dark circles within the operator's mask, respectively, so that the weights always sum to zero.

By using these operators it is possible to detect orientations with 30 degrees of precision. Note that it isn't necessary to know which feature type we have before normalising, as we normalise with all feature types and apply the corresponding PCA and classification. More precisely, the operator's output for orientation  $\theta \in \{0, 30, \dots, 330\}$  is defined by:

$$Op_{\theta}^f(n, s) = \left| \sum_{n'=0}^2 \sum_{s'=0}^{6n} w_{\theta}^f(n', s') V(G(n', s', n, s)) \right| \quad (4.1)$$

where  $f$  is the feature type, *edge*, *bar* or *end*;  $(n, s)$  are the coordinates for the central pixel of the receptive field window;  $w_{\theta}^f(n', s')$  is the operator's weight at the local window's coordinate  $(n', s')$ ;  $G(n', s', n, s)$  is a function that maps from local to global coordinates within the retina as defined in Equation (3.45); and  $V$  is receptive field intensity as defined in Equation (3.53). Depending upon the feature type and orientation, the weights at each position can be either  $+\frac{1}{N}$  or  $-\frac{1}{M}$ , where  $N$  and  $M$  are constants that sum to 19, the number of receptive fields within the window. Due to the hexagonal packing, there are two different types of configuration for  $N$  and  $M$  in all feature operators, one for orientations within the set  $\{0, 60, \dots, 330\}$  and another for orientations within the set  $\{30, 90, \dots, 300\}$  as can be seen in Figure 4.3. We select the

$\theta$  that maximises Equations (4.1) and then rotate the 19 receptive fields about the central field by  $-\theta$ .

### 4.3.3 Principal Component Analysis (PCA)

PCA [Jac91, Jol86] is a multivariate technique in which a number of related variables are transformed into a set of uncorrelated variables. These variables are called the *eigenvectors* and the coefficients which are used to reconstruct the original data are called the *eigenvalues*. These *eigenvectors* correspond to the directions of the principal components of the original data and their statistical significance is determined by the corresponding *eigenvalues*. Given a  $m \times n$  matrix  $X$  containing  $m$  observations of  $n$  variables, PCA entails finding matrices  $V$  and  $D$  so that they satisfy the following equation:

$$C V = V D \quad (4.2)$$

where  $C = X^T X$  is the covariance matrix,  $V$  is a  $n \times n$  matrix containing the *eigenvectors* of  $C$  and  $D$  is a diagonal  $n \times n$  matrix containing the *eigenvalues* of  $C$ . We assume that matrix  $X$  is normalised by subtracting, column by column, the mean value of the variables from each of its elements.

The determination of the *eigenvalues* and *eigenvectors* in Equation (4.2) can be performed using any diagonalisation method. Singular Value Decomposition (SVD) is a method widely used in this case because of its numerical stability. The SVD theorem states that given a  $m \times n$  matrix  $X$  (as above), then there are orthogonal matrices  $U$  ( $m \times m$ ) and  $V$  ( $n \times n$ ) such that:

$$X = U \times \Sigma \times V^T \quad (4.3)$$

where  $\Sigma$  is a  $m \times n$  diagonal matrix containing the singular values of  $X$ , which are also the positive square roots of the (non-negative) *eigenvalues* of  $X^T X$ , the columns of matrix  $U$  contain the *eigenvectors* of  $X X^T$ , and the columns of matrix  $V$  contain the *eigenvectors* of  $X^T X$  [Jac91, Jol86].

Our goal is to decompose the training sets into their *eigenvalues* and *eigenvectors* (or as usually called, principal components). One decomposition is obtained for each of the four feature

Orientation	<i>edge</i>		<i>bar</i>		<i>end</i>		
	Oriented features	Operator masks	Oriented features	Operator masks	Oriented features	Operator masks	
0°		$\bigcirc = 1/8$  $\bullet = -1/11$		$\bullet = -1/12$  $\bigcirc = 1/7$		$\bullet = -1/15$  $\bigcirc = 1/4$	
30°		$\bigcirc = 1/7$  $\bullet = -1/12$		$\bullet = -1/14$  $\bigcirc = 1/5$		$\bullet = -1/16$  $\bigcirc = 1/3$	
60°							
90°							
120°							
150°		$\bullet = -1/12$  $\bigcirc = 1/7$		$\bullet = -1/14$  $\bigcirc = 1/5$		$\bullet = -1/16$  $\bigcirc = 1/3$	
180°		$\bullet = -1/11$  $\bigcirc = 1/8$					$\bullet = -1/15$  $\bigcirc = 1/4$
210°							
240°							
270°							
300°							
330°		$\bigcirc = 1/7$  $\bullet = -1/12$					$\bullet = -1/16$  $\bigcirc = 1/3$

Figure 4.3: Symmetry operators used to detect the feature orientations. The arrows indicate the preferred feature orientation which coincides with the symmetry line in the *end* operator and is perpendicular to this line in the *edge* and *bar* operators.



type's training sets (*edge*, *bar*, *blob* and *end*). We don't need to distinguish between positive (+) and negative (−) representations of features, because the removal of the mean values leaves the two forms being the negative of each other. Each receptive field within a 19-receptive field window will be a variable, and each sample in a feature's training set will be an observation. More precisely:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,19} \\ x_{2,1} & x_{2,2} & \dots & x_{2,19} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,19} \end{bmatrix} \quad (4.4)$$

where  $x_{i,j}$  is the  $j^{\text{th}}$  receptive field of the  $i^{\text{th}}$  observation / training sample. The principal components are stored in a matrix  $V_i$  of  $19 \times 19$  values for each class of features  $i$ .

An unknown data sample named  $Y$  (in our case, a 19-receptive-field-window obtained from a test image) can be projected onto the set of principal components  $V_i$  (obtained as described above) by a simple matrix multiplication:  $P_i = Y' \times V_i$ . The resulting projection  $P_i$  represents that data sample in terms of the principal components for a given primal sketch feature class  $i$ . If a component of the projection  $P_i$  is large, then it suggests that our data is close to the pattern the eigenvector represents, and, if we have a low valued projection, the conclusion is the other way around. We repeat this projection for all classes  $i \in \{\textit{edge}, \textit{bar}, \textit{blob}, \textit{end}\}$ .

PCA is used in our problem to transform the inputs that the neural network modules will receive. However, instead of using all 4x19 projections from the four feature classes, we combine (concatenate) into a new matrix subsets of principal components which are most representative of the data. Then we use this matrix to calculate the input of all the neural modules. The projection operation works identically as described in the previous paragraph, but now the number of columns of the resulting matrix differs from the originally 19 values because we use a combination of different principal components from each of the four feature classes. The main idea is that whenever a data sample from a particular class is filtered through this operation, the outputs related to that class will have high values whereas all the other outputs will remain low, and so the network will have a set of inputs which is more easily separable.

Table 4.1 shows the first 14 principal components of the four classes resulting from the

application of the singular value decomposition on the normalised orientation training data. The process used to construct these feature sets is detailed in Section 4.4.

**Selecting a Subset of the Principal Components.** There are in the literature several *ad-hoc* techniques to select only a subset of the most important principal components [Jol86].

The simplest technique is to select the principal components associated with the largest *eigenvalues*. The general idea behind this selection is that we will be using the components that most contribute to describing the data. On the other hand, we are not interested in components representing the average intensities of the input patterns (the first components of Table 4.1) because those components would not help the process of spreading out the feature classes in the input space. Moreover, the first component will not approximately sum to zero, and it is thus unsuitable to the reflectance estimation discussed in Section 3.3.2. We also do not need to keep the components for features of both positive and negative contrasts, as they are almost the same in Table 4.1. From these considerations above, we have decided to keep the subsets of principal components contained in Table 4.2. Thus the input to the neural networks is reduced to a vector of 17 elements.

#### 4.3.4 Neural Network Architecture

As the classifiers of our architecture we chose MLP-backpropagation networks minimising a least square error metric, because of its simplicity and reasonable computational power.

**Initial Attempts.** Initially we tried to use a unique neural network module to classify the feature's type, orientation and contrast. However this approach didn't work properly. One of the initial architectures that we investigated was a 3-layer neural network in which the input layer had 19 neurons representing each of the 19-window receptive fields, the second layer had a certain number of neurons, and the output layer had a sequence of neurons, each one associated with exactly one of the combinations *feature*, *orientation* and *contrast* of the training data, plus an additional neuron to represent the unknown class. This monolithic classifier had clear disadvantages. Whenever a new sample of a given feature needed to be learnt, the learning of all of the other trained features was affected. Moreover, the training sets were very large, making the training process hard, and this was eventually the main trouble



































































































Component	edge	+bar	-bar	+blob	-blob	+end	-end
1	 15444.51	 6614.50	 8307.51	 1406.85	 2415.70	 14811.19	 25152.20
2	 8574.82	 3332.06	 2665.63	 156.96	 94.70	 4442.84	 2667.40
3	 238.69	 155.40	 155.47	 3.44	 3.30	 222.40	 221.31
4	 63.58	 69.19	 69.69	 0.17	 0.17	 68.29	 68.73
5	 10.12	 31.89	 31.95	 0.16	 0.15	 51.79	 52.03
6	 2.31	 10.18	 10.07	 0.15	 0.13	 7.35	 7.34
7	 1.99	 2.48	 2.48	 0.13	 0.13	 4.48	 4.59
8	 1.39	 1.72	 1.63	 0.12	 0.11	 2.26	 2.38
9	 1.31	 1.09	 1.04	 0.11	 0.10	 1.67	 1.69
10	 1.21	 0.90	 0.87	 0.10	 0.09	 1.35	 1.31
11	 1.21	 0.81	 0.83	 0.09	 0.10	 1.28	 1.29
12	 0.83	 0.75	 0.73	 0.09	 0.08	 1.13	 1.23
13	 1.12	 0.70	 0.52	 0.08	 0.07	 1.03	 1.13
14	 0.88	 0.46	 0.53	 0.07	 0.07	 0.82	 1.09

Table 4.1: Pictorial representation of the first 14 *eigenvectors* and *eigenvalues* extracted from the training sets for each of the primal sketch feature classes (using rotation normalisation). Under each picture is the corresponding *eigenvalue*. Note that the first 3-6 *eigenvectors* encode most of the variation.

Feature	Components
<i>Edge</i>	{2, 3, 4, 5, 6}
<i>Bar</i>	{2, 3, 4, 5, 6}
<i>Blob</i>	{2, 3}
<i>End</i>	{2, 3, 4, 5, 6}

Table 4.2: Chosen subsets of principal components from Table 4.1.

with this approach. It wasn't possible to train the synthetic data using this architecture as the networks never converged to a small error.

Another attempt, described in [GFH98], was to partition three of the feature classes (*bars*, *blobs* and *ends*) into 6 new feature classes according to their contrast intensity (*positive* or *negative*). Then, seven different neural modules, each one designed for a particular feature, was built. These modules had an input layer composed of 19 neurons, followed by a hidden layer. Finally, there was an output layer formed by neurons associated with each of the 6 or 12 standard orientations and a last neuron associated with a non-feature class. The strength of response of an output neuron was a function of the feature's contrast as explained later on in this section. In the case of the positive and negative *blobs* the network's output layer had only 2 neurons, one coding the *blob* itself and the other coding the *non-blob* class. This solved the convergence problem, but when testing the trained networks over real images there was a lot of misclassifications either caused by a training set with insufficient examples or by poor separation in the input space.

**Final Architecture.** In order to achieve better understanding of the problem, we decided to perform a Principal Component Analysis on the training data. After that analysis we realised how the complexity of the problem could be reduced by using PCA to increase the class separation at the network inputs level. We tackled the lack of exemplars in the training set by using a bootstrapping approach which is discussed in Section 4.4.

The current network architectures are very similar to the last architecture described, differing mainly at the input layer: instead of receiving inputs directly from the receptive field windows, they receive the results of the orientation and PCA pre-processing modules. Another difference is that now only 2 output neurons are required by any of the neural modules. The number of output neurons for the oriented features was reduced by using an operator that normalises the

feature orientation as described in Section 4.3.2.

Figure 4.4 illustrates the final neural network architecture

### 4.3.5 Coding the Contrast Information

The contrast within a retinal window is calculated according to the Equation (4.5) [BGG96], which is a well known contrast definition among psychologists and neurobiologists. The desired output for a neuron representing a particular feature was defined in terms of this contrast.

$$c = \frac{|L_{max} - L_{min}|}{L_{max} + L_{min}} \quad (4.5)$$

$L_{max}$  and  $L_{min}$  are the minimum and maximum intensities found in an image patch (Cartesian domain), respectively.

In practical terms,  $L_{max}$  and  $L_{min}$  are two intensities used to draw the brighter and darker regions, respectively, of the synthetic Cartesian training features. In the case of training features hand picked from real images,  $L_{max}$  and  $L_{min}$  are the average intensities of the brighter and darker regions, respectively, within the selected image patch.

In section 3.3.2, we proposed a method to help estimate the original reflectance information of objects. In the end, the receptive field computation was an approximation for the weighted logarithm of the local reflectance information. In section 4.3.3, we explained how to project a receptive field window into subsets of the principal components of the training data in order to achieve better feature class discrimination. As a result of the PCA pre-processing module, the irradiance falling into a receptive field could also be cancelled out.

Thus, the neural network modules do not actually see the original Cartesian intensities, but only the result of the above transformations, which are clearly non-linear. However, this did not pose a problem to the training since, although the mapping became more complex, it remained a function, which in theory can be learnt by a Multi-Layer Perceptron backpropagation neural network.

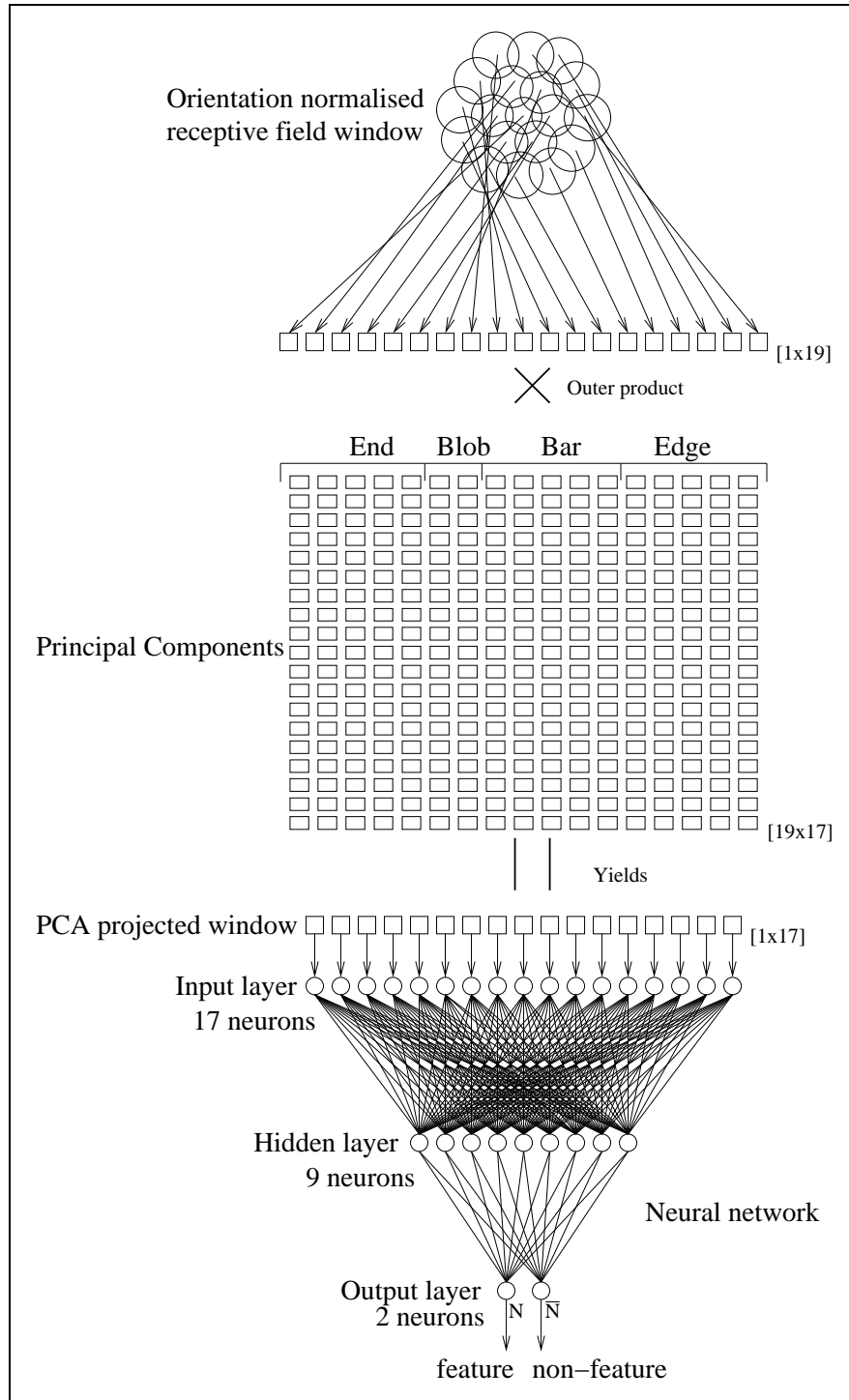


Figure 4.4: Final neural network architecture. At the top of the picture is the PCA pre-processing stage which consists in projecting a normalised receptive field window onto a subset of the training set's principal components. At the bottom is a neural network module with 17 inputs, 9 hidden neurons and 2 output neurons ( $N$  and  $\bar{N}$ ) used to discriminate between feature and non-feature classes. Seven neural network modules like this were trained, one for each of the seven feature classes: *edge*, *+bar*, *-bar*, *+blob*, *-blob*, *+end*, *-end*.

### 4.3.6 Classification Rule

We use a classification rule that takes into account the strategy used during training: whenever a feature that should be recognised by a given neural module is presented then that module's network is trained to output the feature's contrast through neuron  $N$ , and zero through neuron  $\bar{N}$ ; if a counter-example is presented, then neuron  $N$  should now output zero and neuron  $\bar{N}$  one. However, we need to be more tolerant when classifying untrained features, as the network outputs will not necessarily create a sharp separation between feature and non-feature classes. Let's consider a neural module representing a given feature class with its 2 output neurons, the classification rule used in our experiments is presented in the following pseudo-code, which basically states that a module recognises a features whenever its neuron  $N$  produces an output that is above a threshold  $THD$  and also above the output of the other neuron  $\bar{N}$ :

```

if (( $O(N) > O(\bar{N})$ ) and ( $O(N) > THD$ ))
  then  $C = O(N)$ 
  else  $C = 0$ 
endif

```

where neuron  $N$  represents an input belonging to the input's true class, and neuron  $\bar{N}$  represents the non-class input.  $THD$  is the classification threshold,  $O()$  is a function that returns the neuron's numerical output, and  $C$  is the output of the classification rule, which represents the contrast of the detected feature ( $C$  is 0 in the case of no feature being detected). In Section 4.4.3 we show how to obtain the value of  $THD$ .

An unknown input window is fed into all seven neural modules and, provided that the class of that input has been trained before, the corresponding neural module will produce the appropriate contrast while all the other modules will not recognise the input (having neuron  $\bar{N}$  near one and neuron  $N$  below the threshold). What is said above is true in most of the cases due to the construction of the training sets discussed in the following section, but that doesn't completely eliminate the possibility of two or more modules recognising the same input pattern as belonging to their respective classes. For the moment, we do not give any special treatment to this situation, as it might be reasonable to think that some blurred features could be recognised as belonging to two or more classes at the same time. However, there is

no doubt that, if an exclusive classification is required by a final application, a winner-takes-all strategy can be easily implemented based upon the modules' outputs.

## 4.4 Training and Evaluation

It is possible to identify two distinct ways for generating a training set for neural network modules. One possibility would be to extract many 19-receptive field windows from real images and then classify each of these windows in terms of (*feature, orientation, contrast*) either by hand or using the previous 3x3 heuristic operators. However, the amount of data would be very large and there would be no guarantee of accessing the representative points in the feature space even using a large number of real images.

The alternative which we have adopted in this work was to construct an initial training set from synthetically generated features. This seems to be a better approach because a large number of feature variations can be easily generated and, by using the underlying hypothesis that the feature examples are chosen from a more descriptive set, this allows for a smaller training set to be constructed when compared to the previous manual alternative. After some initial experiments, we reached the conclusion that although the synthetic training set was very useful, the manual selection of a small set of features from real images was still required to enhance the final classification results.

It is important to emphasise that the synthetic patterns are drawn in the Cartesian space and therefore this does not contradict our initial assumption that to build a model for the features in the retinal space is a difficult task which has justified the use of a learning based approach.

### 4.4.1 Synthetic Training Data

*Edges* were modelled as a step gradient transition defining a straight line passing through the central receptive field. *Bars* were modelled as having two parallel gradient transition lines passing through the borders of the central receptive field. *Blobs* were modelled as a circular gradient transition between the central receptive field and the outer window of receptive fields. Finally, *ends* were modelled as half *bars*, that is, *bars* that end at the central receptive field.

Contrasts within the set  $\pm\{0.3, 0.4, 0.6, 0.8, 1.0\}$  (using the formula of Equation (4.5)) were



used when drawing *bars*, *blobs* and *ends*, a negative contrast here means that the intensities in the feature background are higher than the ones in the feature itself. *Edges* were drawn using only positive contrasts, i.e. the intensities in the region above the feature orientation line are greater than the intensities in the lower region in order to avoid the generation of the same pattern twice as the feature orientations covers the whole circle. Fifteen different combinations of intensity were used in the generation of the contrasts for all of the features. One of the intensities was taken from the set  $\{85, 170, 255\}$  and the other was derived according to Equation (4.5) to produce the desired contrast.

*Ends* and *edges* were generated at 12 different orientations in the range  $(0^\circ, \dots, 330^\circ)$ , in steps of  $30^\circ$ , while *bars* were generated only at 6 orientations in the range  $(0^\circ, \dots, 150^\circ)$  due to symmetry reasons, and also in steps of  $30^\circ$ . Other sources of variability in the training sets were the “size” of the feature and the use of Gaussian additive noise. *Blobs*, *bars* and *ends* were allowed to vary in size according to 0.6, 0.7 and 0.8 of the central receptive field diameter. Gaussian additive noise was added to each receptive field of the drawn features in order to broaden the training set.

Random *counter examples* of the 19-receptive field windows were generated in order to help the training process. These *counter examples* simulate unstructured input data and data from other low level features not considered in this work that is inevitably present in real images. After being generated, a *counter example* is validated with respect to the feature type they are going to be used with: if the normalised Euclidean distance from the *counter example* to any of the 19-feature vectors of the given type falls below to a given threshold (we have chosen a 0.15 threshold), then that *counter example* is rejected and a new one has to be generated and validated. Counter examples sets were also enriched with exemplars from the other feature classes. Table 4.3 summarises the parameters used in the generation of the synthetic training sets and the total number of features examples generated per class. Table 4.4 shows some of the training features generated in the above way.

A total of seven different training sets was built, one for each neural network module designed to classify *edges*,  $\pm$ *bars*,  $\pm$ *blobs* and  $\pm$ *ends*. Each training set was built from an equal number of feature examples and *counter examples*. The number of examples per feature class was determined previously by the sets of parameters presented on Table 4.3.

Feature	Orientation	Contrast (C)	“Size”	Noise ( $\sigma$ )	Total
<i>Edges</i>	{0, 30, ..., 330°}	+{0.3, 0.4, 0.6, 0.8, 1.0}	not applicable	{0, 1, ..., $n(C)$ }	2700
<i>+Bars</i>	{0, 30, ..., 150°}	+{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	1350
<i>-Bars</i>	{0, 30, ..., 150°}	-{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	1350
<i>+Blobs</i>	not applicable	+{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	225
<i>-Blobs</i>	not applicable	-{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	225
<i>+Ends</i>	{0, 30, ..., 330°}	+{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	2700
<i>-Ends</i>	{0, 30, ..., 330°}	-{0.3, 0.4, 0.6, 0.8, 1.0}	{0.6, 0.7, 0.8}	{0, 1, ..., $n(C)$ }	2700

Table 4.3: Parameters used in generating feature examples for the training sets. The top noise level was defined as a linear function of the feature’s contrast to prevent high noise levels being applied to low contrasting features:  $n(C) = \text{INT}(6 \times |C| + 1)$ . The last column indicates the total numbers of examples generated per feature class.










Image type	<i>Edge</i>	<i>+Bar</i>	<i>-Blob</i>	<i>+End</i>	<i>Counter Example</i>
Cartesian inputs					not applicable
Retinal outputs					

Table 4.4: Some examples of the training features. The differences in contrast between the Cartesian inputs and the retinal outputs are due to the logarithmic receptive field computation.

Half of the *counter examples* was randomly generated and validated as discussed above. The other half were exemplars randomly selected from the other feature classes in the following manner. Let  $CX$  be half of the *counter examples* required by a particular feature class  $F$ ;  $Fc \neq F$  be a particular feature class from which features need to be selected as *counter examples* for  $F$ ;  $\#(Fc)$  be the size of  $Fc$ , as presented on Table 4.3;  $S = \sum_{Fc \neq F} \#(Fc)$  be the sum of the sizes of all feature sets different from  $F$ . Then, the number of features  $NFc$  to be selected from  $Fc$  as *counter examples* of  $F$  is given by Equation (4.6). This is equivalent to saying that a particular feature  $Fc$  contributes as *counter example* in the training set of feature  $F \neq Fc$  in a way that is proportional to the size of  $Fc$  relative to the sum of the sizes of all features going to be used as *counter examples* of  $F$ .

$$NFc = \frac{\#(Fc)}{S} \times CX \quad (4.6)$$

#### 4.4.2 Initial Training

We used the Aspirin/MIGRAINES [Lei92] neural network simulator to perform the training and evaluation of our proposed method. A learning rate of 0.005 and inertia (or momentum) of 0.95 were used with the standard backpropagation algorithm. A neural module was considered trained when all the training patterns passed with a 0.1 maximum error bound. A random presentation order was applied when selecting patterns to be trained. Table 4.5 summarises the class specific architectures and training parameters.

Neural Module	Architecture $I \times H \times O$	# of Epochs $\oplus, \ominus$
<i>Edge</i>	$17 \times 9 \times 2$	80
<i>Bar</i>	$17 \times 9 \times 2$	190, 1360
<i>Blob</i>	$17 \times 9 \times 2$	690, 5540
<i>End</i>	$17 \times 9 \times 2$	480, 1120

Table 4.5: Training parameters.  $I \times H \times O$  are the input, hidden and output units.  $\oplus$  and  $\ominus$  are the number of training epochs corresponding to the sets having positive and negative contrast features, respectively.

### 4.4.3 Evaluating the Initially Trained System's Performance

We tested the trained networks with a set of unknown synthetic features. We employed the same feature generator used to build the initial training sets with the difference that here the features were generated at arbitrary orientations and contrasts.

We generated 80 test exemplars per class by varying the contrast in the range 0.21 to 1.0, with a step of 0.01 (-0.21 to -1.0, with a step of -0.01, for negative features). The intensities used to produce the above contrasts were chosen randomly as well as the remaining parameters specifying orientation, noise level and size. Figure 4.5 shows the target output contrasts (vertical axis) for all the testing sets (horizontal axis shows the pattern number). The first 80 patterns are examples of the feature class in order of increasing contrast, and the next 83 patterns are counter examples (the first 40 were randomly generated and the remaining 43 were randomly chosen from other class examples).

Figure 4.6 shows the actual outputs of the edge neural module produced with the synthetic testing data. As the graphs for the remaining features are similar to the edge one, to save space, they were all accommodated in a single figure (Figure 4.7).

As one would expect, the few errors that appear on the first half of the graphs are associated with low contrast features. The oscillations up and down along the predicted contrasts for the testing sets when compared to the targets outputs (Figure 4.5) are partially explained by the convergence error of 0.1 used during the neural network training. The same applies to the small oscillations in the second half of the pictures (Figures 4.6 and 4.7) corresponding to the networks response to the counter-examples. Reducing the neural network convergence error could reduce the prediction errors during testing, but at the cost of a slower training and risk of *overfitting*, which could incur loss in generalisation. Another cause for the oscillations within the first half of the graphs are the small prediction errors caused by the symmetry operators as explained in the following paragraph. The occurrence of spurious higher amplitude oscillations in the second half of some of the graphs also indicate that more counter examples might be needed in the training sets, but this will be dealt with later on in this section when we explain how to enhance the training sets.

Figure 4.8 shows, along the y-axis, the absolute differences between real and estimated orientations when processing the feature's through the symmetry operators. Blob features were

not considered in the graph as they do not have an orientation. These errors were computed as the absolute difference between the real (ground truth) and estimated (obtained by the corresponding symmetry operator) feature's orientation for the first 80 patterns (x-axis) in the testing sets, which correspond to examples of features. As an indication of good performance, the top errors were all under the operator's resolution (30 degrees).

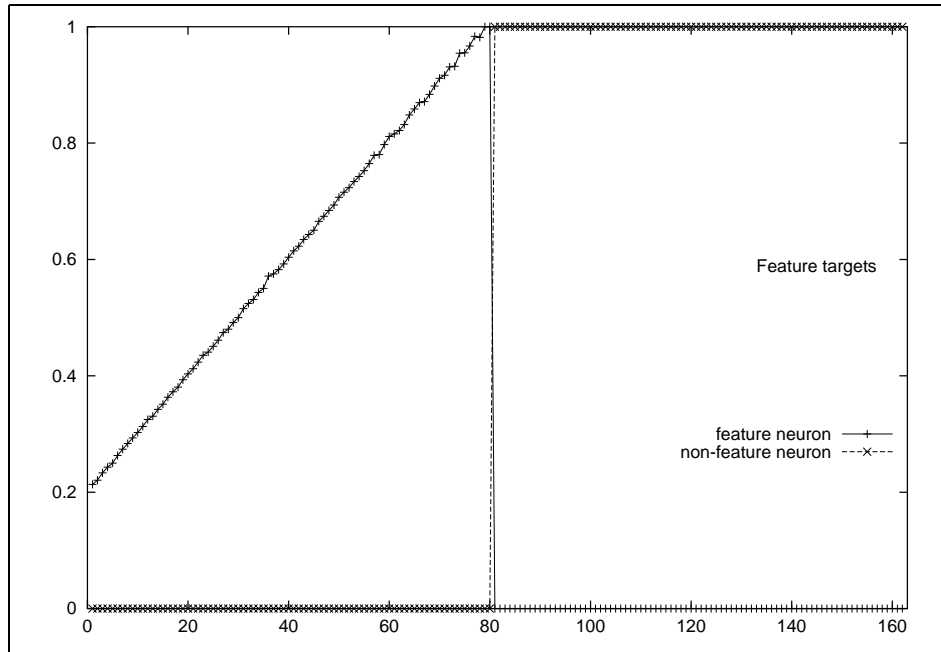


Figure 4.5: Target outputs for the testing sets. The  $x$  axis represents the pattern number and the  $y$  axis represents the desired neuron output. Note that the first 80 feature contrasts are not on a perfectly straight line because they were obtained from 2 integer intensities as specified in Equation (4.5).

Now we need to specify the value of the threshold  $THD$  introduced in Section 4.3.6. Our goal is to choose the highest classification threshold that produces best overall performance. The reason for this is that we are not interested in detecting very low contrast features. From Figures 4.6 and 4.7 it is easy to see that a 0.15 threshold meets this requirement. As an exercise to confirm this observation and to have quantitative results to compare our approach with the previous one discussed in Sections 4.2 and 2.6, we varied the threshold from 0.0 to 1.0, in steps of 0.01, and measured the classification errors. Figure 4.9 shows the results of the threshold experiment for edges. The results for the remaining features are all in Figure 4.10, since the graphs are similar in shape to the edge one. We can see from the figures that any threshold smaller than 0.15 is associated to the overall best performances in all the networks.

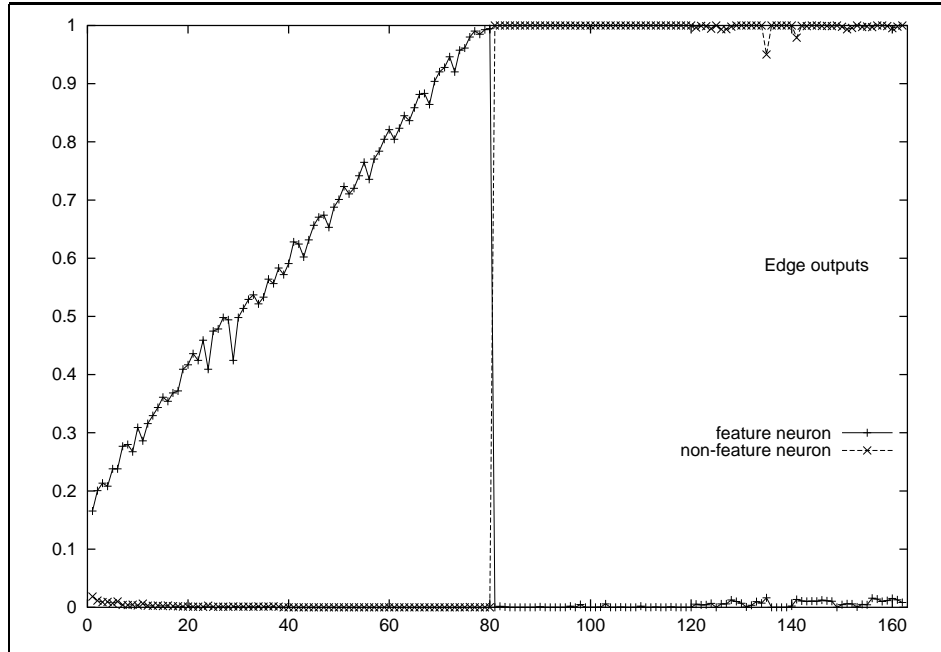


Figure 4.6: Outputs from the trained edge neural module when applied to the testing sets. In order to facilitate the reading of this picture the contrasts of negative features have been converted to their absolute values.

Error type I (false acceptance) occurs whenever the classifier accepts a non-class sample as belonging to the class. Error type II (false rejection) occurs whenever the classifier considers a member of the class as not belonging to the class. The sum of these two errors gives the total error produced by the classifier and, in our specific case, the percentage of correct answers is 100% minus the total error. Equation (4.7) summarises this relation. Sometimes it is also useful to define a rejection criterion, which provides the classifier with the power of answering that a given input cannot be classified, and thus reducing the chances of error, but we decided not to apply this criterion because in the domain of feature extraction, rejecting a feature has the same practical effect as classifying it as not belonging to the class.

$$\text{Correct} = 100\% - (\text{ErrorI} + \text{ErrorII}) \quad (4.7)$$

### Comparing the Proposed Method with Grove and Fisher's Approach

In order to provide a quantitative evaluation of the relative performance of the method proposed here and the previous approach of Grove and Fisher [GF96], which was discussed in Sections

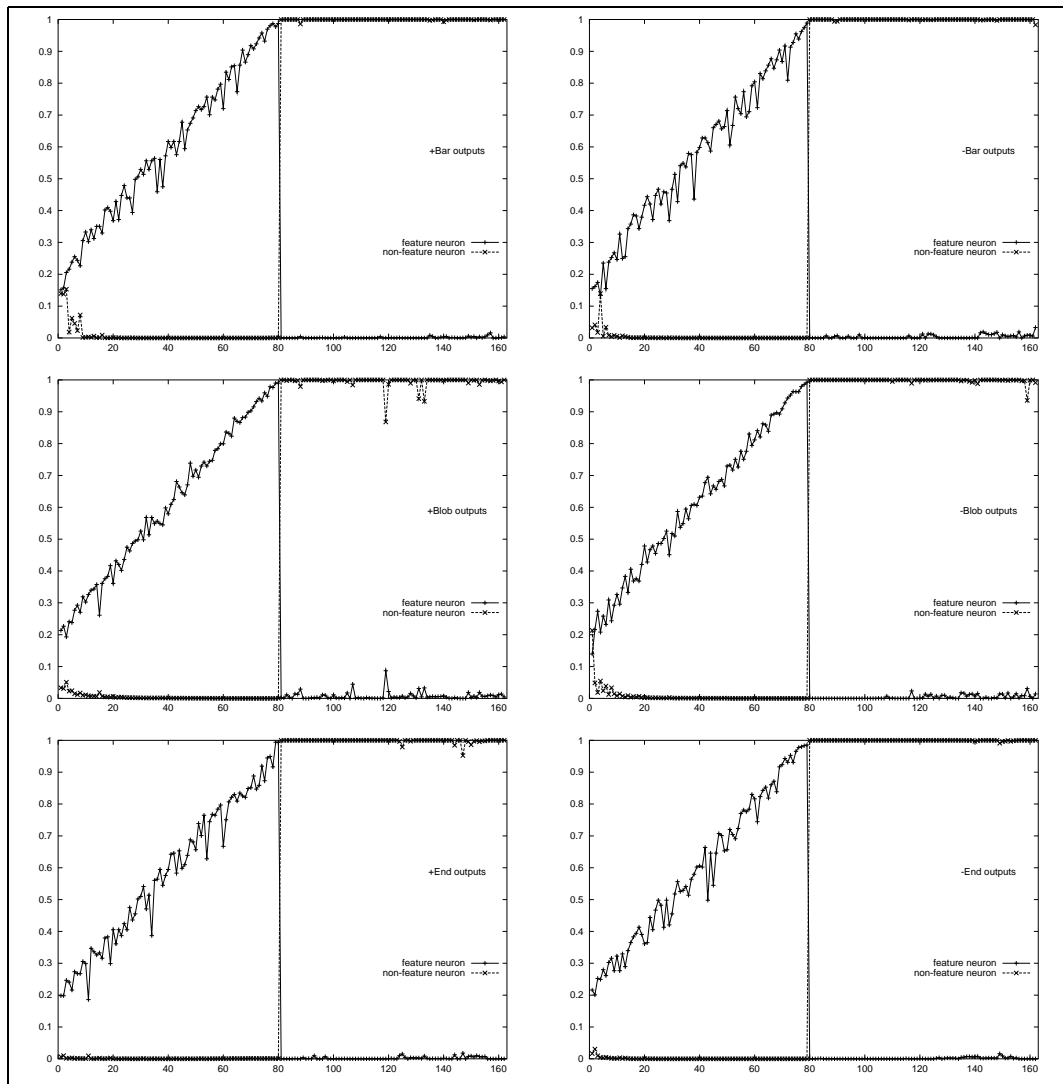


Figure 4.7: Outputs from the remaining trained neural modules when applied to the testing sets. In order to facilitate the reading of this picture the contrasts of negative features have been converted to their absolute values.

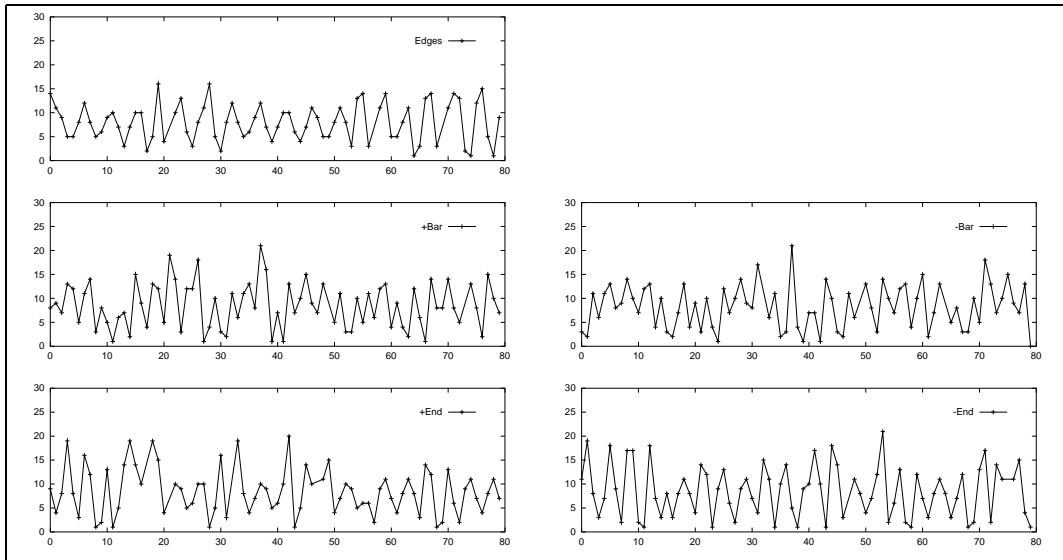


Figure 4.8: Absolute errors of the symmetry operator for the testing sets of the oriented features. Note that these errors do not exceed the value of 20 degrees, which demonstrates a reasonable performance as the highest error is well under the operators' resolution (30 degrees).

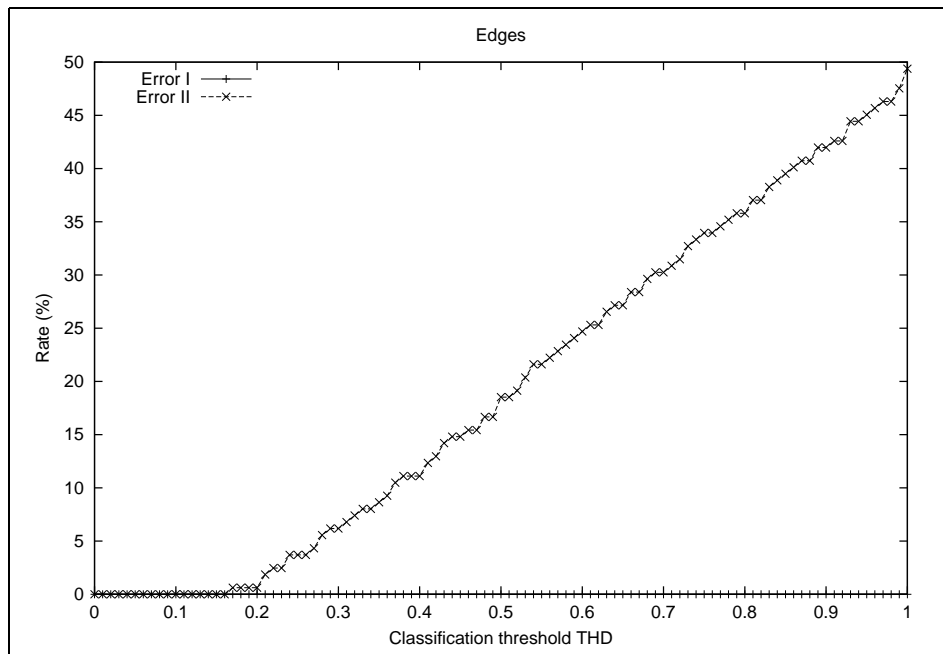


Figure 4.9: Errors type I and II vs. the classification threshold for the trained edge networks. These statistics were calculated from synthetic testing sets. Note here that the type I error curve lies on the horizontal axis.



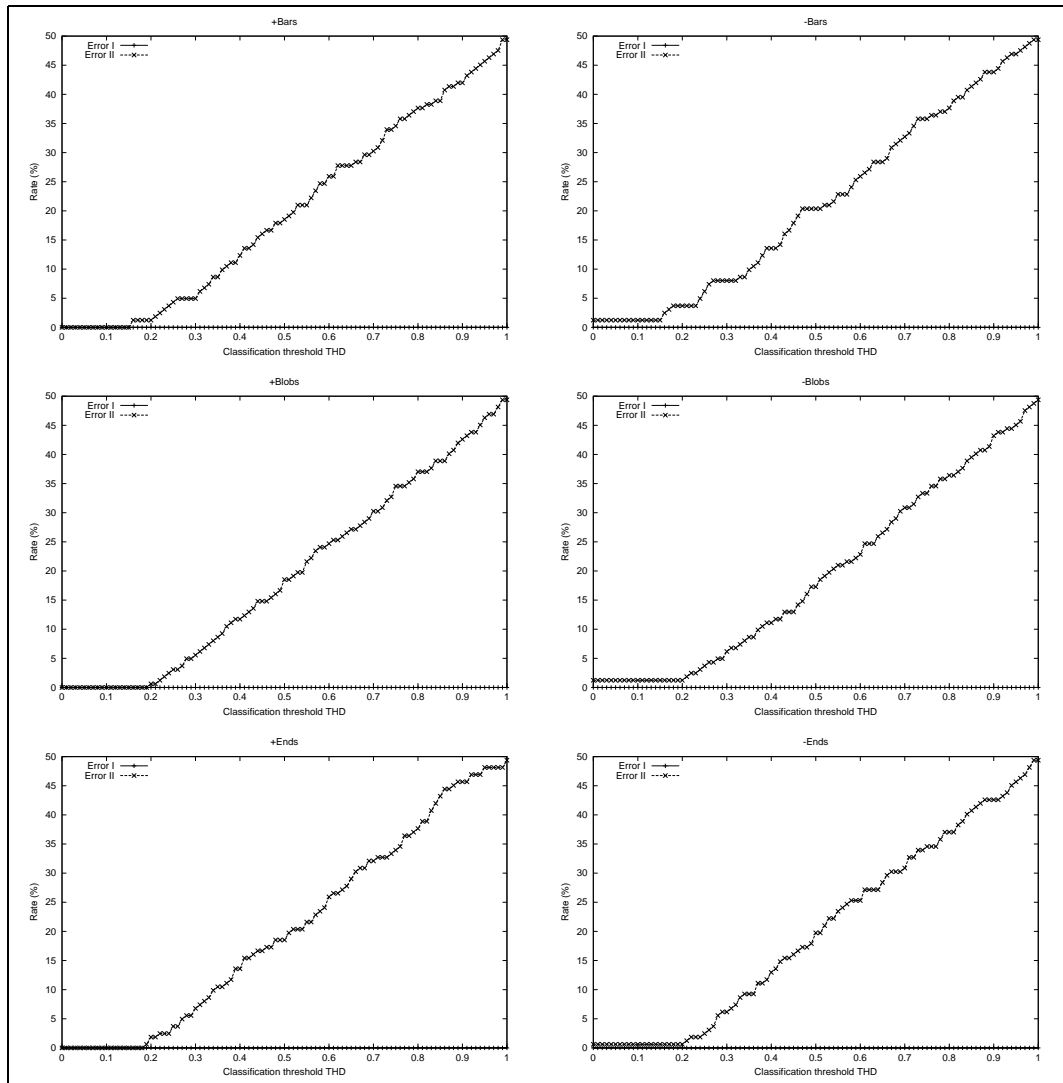


Figure 4.10: Errors type I and II vs. the classification threshold for the trained networks for the remaining features. These statistics were calculated from synthetic testing sets. Note here that the type I error curve lies on the horizontal axis.

4.2 and 2.6, we used the same testing sets described above to generate classification statistics (likewise the ones presented in Figures 4.9 and 4.10) for the previous approach. Since the classification threshold of Grove and Fisher's approach is a function of the intensities found in the images, we decided to vary their threshold from 0 to 100 in steps of 1. This yielded exactly 101 sequential thresholds which is compatible with the range of variation used before to test the neural network approach (thresholds varying from 0 to 1 in steps of 0.01). We later discovered that this was a good choice, since the error curves converged near the higher thresholds to the same levels obtained before. Figure 4.11 shows the errors of the edge operator. The results of the remaining classifiers are presented on Figure 4.12.

By contrasting Figures 4.9 and 4.10 with Figures 4.11 and 4.12, respectively, it is possible to see that all of the new classifiers had no type I errors for all tested thresholds, whereas, when using the previous approach, only *-Ends* had no errors of type I. This excellent type I error performance by the neural network approach may be partially explained by the fact that the classification rule (presented in Section 4.3.6) decides based on the output of two neurons (one responsible for representing features and the other, non-features) that are mutually exclusive. Moreover, there was a reasonable threshold range (typically between 0 and 0.15) for which the new classifier had almost zero errors of type II (optimum performance). No such range could be found in the graphs of the previous approach, which indicates that the previous approach is more unstable and unreliable.

#### 4.4.4 Adding Features from Real Images

The final step of the approach is to enrich the training sets with features extracted from real images. Although using a synthetic training set proved to be very useful to generate a large number of feature variations and also to create a more representative training set, adding features from real images was still needed. The main purpose of these real image features is to correct some small classification errors related to features not generalised from the training sets by the neural network modules.

Initially a set of images is tested using the modules trained with synthetic features. Then a small subset of these images is used as a source of additional training exemplars, which are manually selected. The intensities used to compute the contrast of these new exemplars are estimated from their retinal outputs as if they were composed of uniform intensity patches

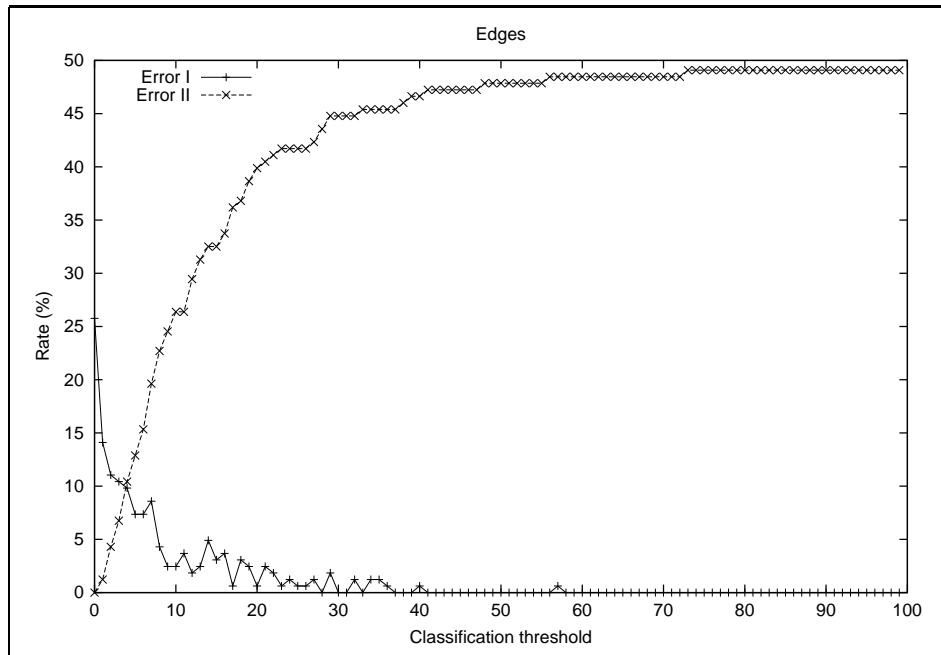


Figure 4.11: Errors type I and II vs. the classification threshold for edges using the previous approach of Grove and Fisher [GF96]. These statistics were calculated from synthetic testing sets.

in the Cartesian domain (similarly to the synthetic features). Finally, the neural modules are re-trained with the improved training sets and tested over the same initial set of images. This process is repeated until the visual output of the extracted features is satisfactory.

Although we re-use the previous network weights as a starting point, we need to re-train the networks with the complete training sets, not only with the recently extracted features. If we do not proceed that way, the network would probably end up forgetting the previously learnt patterns. Another important issue is that when inserting new patterns into the training sets we need to keep balanced the number of examples and counter-examples, otherwise the network would become biased toward either side. This does not mean that the training sets have to have the exact same number of examples and counter examples, but they should be at least at the same order of magnitude.

#### 4.4.5 Experimenting on Synthetic and Real Images

The log-polar mapping described in Section 3.2 is applied to an input Cartesian image, then all the possible 19-receptive field windows are extracted and fed into each of the trained

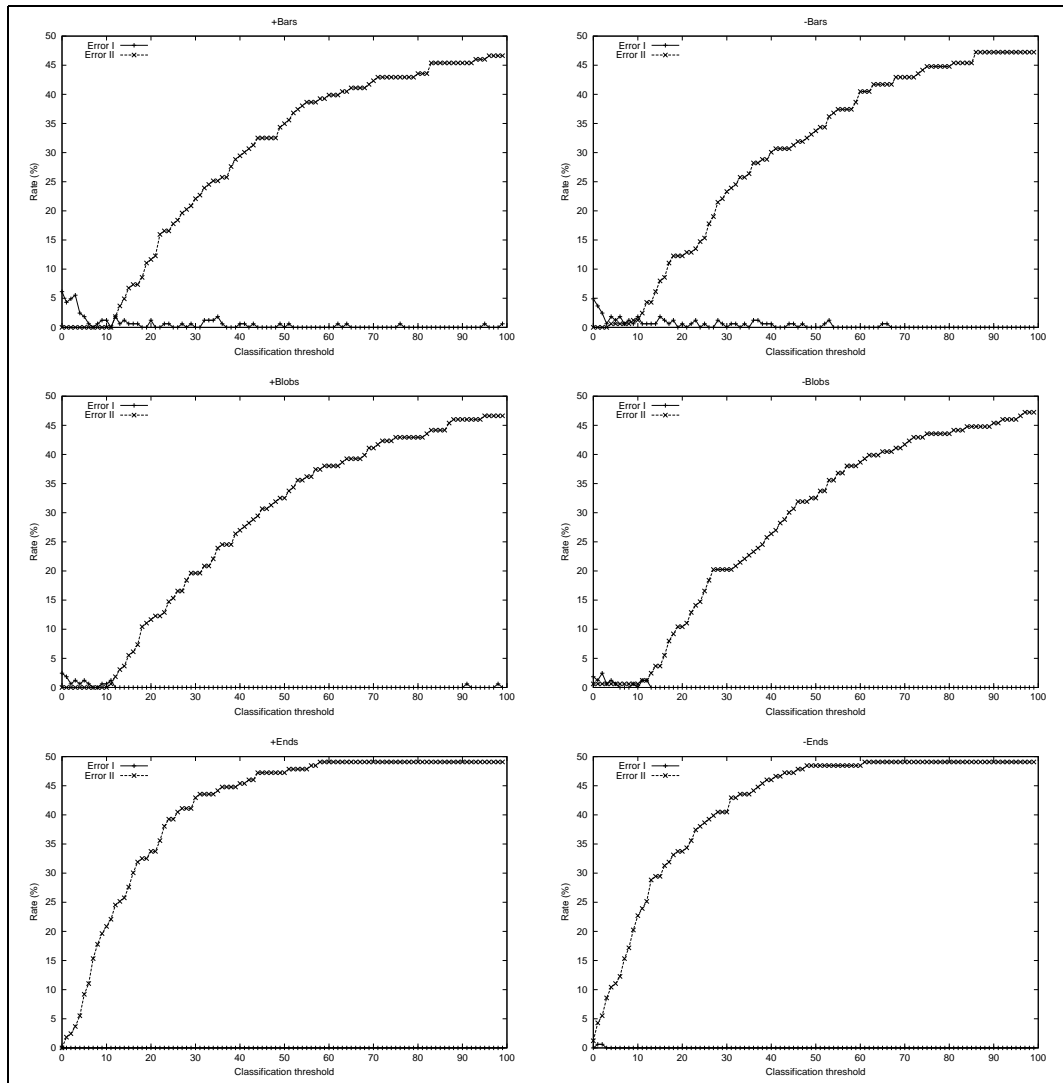


Figure 4.12: Errors type I and II vs. the classification threshold for the remaining features using the previous approach of Grove and Fisher [GF96]. These statistics were calculated from synthetic testing sets. Note that the type I error curve is mainly on the horizontal axis.

neural network modules. Although we had available at this stage several measurements about a feature, like the orientation and contrast, in order to simplify the evaluation process, we decided to deal only with the feature's identity and strength. The classification rule discussed earlier in this section was used to read the results from the neural network outputs.

In order to help the reader to understand the outputs of the feature extraction system described in this chapter and before applying it to any real images, which may contain complicated textures and features not easily spotted by the eye, a number of synthetic images were initially used. Input images containing horizontal, vertical and diagonal edges at varying contrasts were used to test the outputs of the edge classifier. The positive and negative bar classifiers were tested against images containing vertical, horizontal and diagonal bar patterns. The blob classifiers were tested against images containing blob-like features at varying sizes and contrasts. Finally, images containing simple polygons at varying contrasts and orientations were used to test the end classifiers (the vertices of these polygons are the features to be detected).

These examples are shown in the Figures [A.1](#) to [A.5](#) in Appendix A. From a subjective evaluation of these figures, apart from some small uncertainty with regards to the feature location, it is possible to conclude that the neural classifiers are doing a reasonable job at detecting the features they have been designed for. Note that, in some of the pictures, the edge and bar operators eventually fail near the centre or near the periphery of the images, which is expected since the scale of an image feature does not always match the scale of the receptive field window. The last columns of Figures [A.1](#) to [A.5](#) illustrates the results of the previous approach when applied to the same set of synthetic input scenes. Overall, the previous approach had poor contrast sensitivity and a greater number of features not being detected (discontinuities) when compared to the learning based approach proposed in this thesis.

Figure [A.6](#) compares the results of the proposed and previous approaches when applied to the same test image. There we can see that the new approach brought a reasonable improvement to the feature extraction process not only with respect to accuracy (more features correctly extracted) but also to the enhanced quantisation of the strength of the features.

A number test images were classified using the initially trained neural networks. In order to exemplify the role of using training features from real images, one of the test images was

used as a source of a few additional training exemplars which were manually selected from the results of that initial application of the trained networks. By looking at the results on real images (e.g. left part of Figure A.7 in Appendix A), it is possible to see that there are more non-features classified as features than the other way around. As a result, we decided that only receptive field windows of misclassified non-features would be used to form these additional training exemplars in this experiment. Four counter-examples were selected per feature class (Table 4.6). Figure A.7 in Appendix A shows the results of our system over a test image, before and after the addition of manually selected features into the training sets, whereas Figures A.8 and A.9 show, on different test images, only the results of the networks trained with the improved training sets.

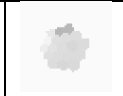

























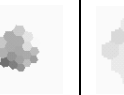

edges	+bars	-bars	+blobs	-blobs	+ends	-ends
						
						
						
						

Table 4.6: New training patterns manually selected from image 1 (left part of Figure A.7). These patterns are all counter-examples. The receptive field sizes vary due to the locations they were found in the image.

## 4.5 Concluding Remarks

In this chapter we have shown how primal sketch features (*edges*, *bars*, *blobs* and *ends*) could be extracted from the retina-like (log-polar) image representation specified in the preceding chapter. These features are part of Marr's hypothesis for the human visual system, and are believed to take part in early visual processes providing a representation that captures invariant primitives from the object's surface and also acting as cues for an attention mechanism. A previous attempt to extract such features [GF96] achieved only partial success. Features were detected using a number of manually defined logical operators within a fixed retinal window,

which, when applied to real images, failed to detect some low contrast features as well as misclassified a number of others (for reference, see the right side of Figure A.6 in Appendix A).

The method presented in this chapter takes advantage of the inherent learning (from examples) and generalisation properties of neural networks. Instead of designing feature extraction operators from scratch, the basic idea was to use a set of exemplars of features and non-features to train a MLP-backpropagation neural network. We initially started with a monolithic network designed to learn all seven types of features (*edges* plus negative and positive contrast versions of *bars*, *blobs* and *ends*) and answer with the feature's type, orientation and contrast, which proved to be impossible to train. The initial architecture received some refinements and ended up having seven independent neural modules (only connected through the training sets, i.e. examples from one module used as counter-examples in the others), one for each of the feature classes considered and receiving inputs from a same PCA pre-processing module. After a specific symmetry operator normalises the features' orientation, the PCA pre-processing module is used to project feature's pixels onto a subset of the principal components of the training data with the purpose of spreading out the classes in the feature's manifold.

In summary, the new approach presented better results with respect to the number of correctly classified features, provided a richer description for the image data with the addition of an estimate for the feature's contrast, and became a more flexible solution to the problem in the sense that whenever a new feature class is required, only its training set needs to be provided.

At least three of the primal sketch features extracted by our approach, blobs, oriented bars and edges, have a closer relation to biologically motivated Machine Vision. Since the pioneering works by Hubel and Wiesel [HW62] reporting the manner in which some neurons (cortical simple cells) in the cat's visual cortex responded to the images of spatially localised bars and edges, there have been many studies trying to model filter shapes to perform an image transformation equivalent to that of simple cells. The two most commonly studied filter shapes are the second differential of a Gaussian ( $\nabla^2 G$ ) [MH80] and the Gabor filter [Dau88]. The first can be approximated by a difference of Gaussians (DOG), and Marr [Mar82] was one of its well known advocates. The third is related to the work of Daugman [Dau88], who showed that the two-dimensional Gabor filter represents the optimal combination of frequency and space information. Daugman proposed that an ensemble of cortical simple cells would be best

modelled as a family of 2-D Gabor wavelets sampling the input domain in a log-polar manner. The idea that simple cells evolved to an optimal design has caused growing interest in the use of Gabor filters by both Neuroscience and Computer Vision communities.

In contrast to the above approaches, in which a single receptive field can be used to detect a given feature, in our image representation, features are detected at a higher level, at the level of a receptive field window. However, both ways to detect a feature still present some similarities. In particular, it is possible to identify some resemblance between the shape of the Gabor wavelets (see [Lee96] for some pictures) and the first PCA components for edges and bars presented in this chapter, which suggests that there might be a relation between the two. An analogous assertion can be formulated relating the DOG or the  $\nabla^2 G$  operators and our blob detector. Perhaps, since we are looking at the same features, the PCA plus neural network arrangement might be learning something very close to the mathematically pure version of the operator, except that our receptive fields are not identical in size. Further investigation on these relations is left as a future work.

The following chapter addresses the problem of how to learn relationships from sets of iconic (2D) object models obtained from a sequence of scenes. These models contains a list of log-polar image descriptions that are close to each other with respect to a similarity measure, and found in the scenes by an existing attention mechanism. Such image descriptions are composed of all the primal sketch feature images, obtained by the modules discussed in this chapter, in addition to the original log-polar *rgb* retinal images.



## Chapter 5

# Learning Iconic Models

Vision systems that *learn and adapt* represent one of the most important future directions in image understanding research. This reflects the overall trend - to make intelligent systems that do not need to be fully and painfully programmed. It is the only way to develop vision systems that are robust and easy to use in many different tasks. *T. Poggio and D. Beymer [PB96]*

### Contents

---

<b>5.1 Learning Primitive Object Feature Models . . . . .</b>	<b>98</b>
<b>5.2 Learning Structured Models via a Representational Graph . . . . .</b>	<b>106</b>
<b>5.3 Exploratory Experiments . . . . .</b>	<b>119</b>
<b>5.4 Structure Learning Case Studies . . . . .</b>	<b>140</b>
<b>5.5 Summary . . . . .</b>	<b>153</b>

---

This chapter addresses the important problem of how to learn relationships from sets of iconic (2-D) object models obtained from a set of images. It assumes a vision system that operates by foveating at interesting regions in a scene, extracting a number of raw primal sketch-like image descriptions, and matching new regions to previously seen ones. An iconic model is defined as a set of regions, or object instances, that are similar to each other, and comprises a list of relative scales, orientations, positions and similarity scores for each pair of image regions. A solution to the structure learning problem is presented in terms of a graph based representation and algorithm. Vertices represent instances of an image neighbourhood found in the scenes. An edge in the graph represents a hypothetical relationship between two neighbourhoods. Intra and inter model relationships are inferred by means of the cliques found in the graph, which leads to rigid geometric models inferred from the image evidence.

## 5.1 Learning Primitive Object Feature Models

This section explains how the process of learning object feature models or primitive geometric models can be implemented. Although this is not the main issue of this chapter, the method for inferring relationships between these object features, which is described in the following section, relies upon certain aspects of the method presented here.

Chapter 4 dealt with primal sketch features, which are essentially low level features based on discontinuities found in the images. These features were learnt and extracted independently of the scene contents and are used by an existing attention mechanism to help the iconic system focusing attention only to the most interesting or promising regions of a larger visual world.

In this section, we are interested in finding out which of these regions, taken from different scenes and represented by their primal sketch features, look similar to each other. Grouping such similar regions can be seen as a way of creating hypothesis for higher level objects in the scenes and that is exactly what we call by learning object feature models. While grouping image regions, we measure and save their relative scale and orientation, and for this reason we can also call these object models by primitive geometric models.

We use the term *feature* in this chapter with a meaning that is slightly different from that used in Chapter 4, which was related to low level discontinuity properties of the images, whereas in this chapter a feature instance means a high level component or part of an object as a whole that falls into the retinal area. The term *model instance* is also used throughout the text with the same meaning.

### 5.1.1 System's Architecture

We assume a vision system architecture as shown in Figure 5.1. The architecture is based on previous research system described in [GF96] and reviewed in Chapter 2 of this thesis. The figure shows only the modules directly related to this chapter.

Module (a) is responsible for converting input pixels into a retina-like image representation (discussed in Chapter 3), which in turn is used by the second module (b) to generate primal sketch features at a number of orientations and contrasts through a neural network (described in Chapter 4 and in [GFH98, GF01, GF02]).

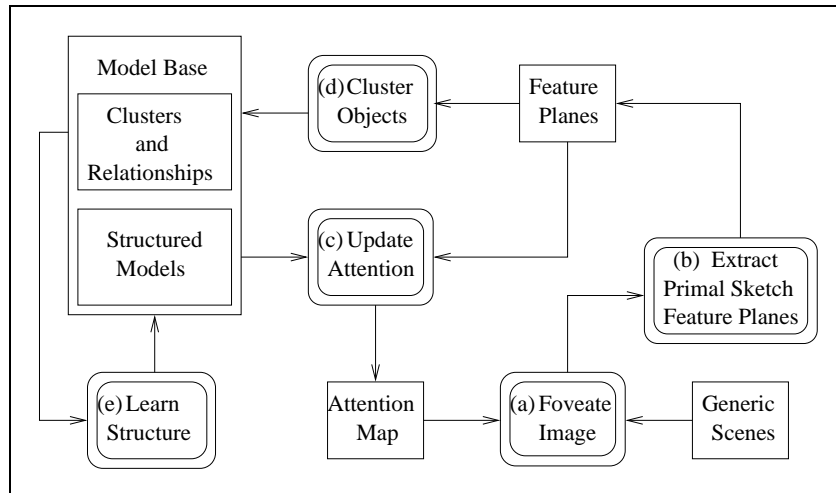


Figure 5.1: System's main modules.

During scene examination, an attention mechanism (c) continuously updates a map which weights points of interest in the scenes based upon the primal sketch features and colour information. The attention mechanism for the moment is the one proposed by Grove and Fisher [GF96], which was not used in our experiments because it did not deliver interest points that were always consistent within a same object, regardless of the scene configuration. Research on an improved attention mechanism is currently under development [Sun01]. The foveation area is smaller than the full scene so that just a smaller section of the input image is analysed at a given time.

Module (d), for which a prototype implementation is proposed in this chapter, clusters primal sketch planes (representing primitive objects) into model classes. It also stores information about the scale, position, orientation and similarity between the clustered objects, which is used by module (e) to build a representation describing possible relationships between primitive models that form larger structured models. Module (e) is the focal point of this chapter. A brief description of modules (d) and (e) can also be found in [GF00].

These structured models can in turn be used to improve the attention and matching processes. In this thesis we are not concerned with the issues of parallelism and the use of subcomponent evidence in the attention mechanism and matching process, which were covered by Marques [Mar96] and MacKirdy [Mac97], respectively.

### 5.1.2 Model Representation

The system's architecture presented in the preceding section assumes that there is a set of scenes or frames to be processed. Each scene  $i$  will be denoted by  $S^i$ . For each scene  $S^i$ , the attention mechanism selects only a subset of the points to be foveated. The  $j^{\text{th}}$  point of scene  $S^i$  will be denoted by  $P^{i,j}$ . The result of foveating at a given point  $P^{i,j}$  is a stack of primal sketch planes at all possible orientations and contrasts (totalling 50 log-polar image planes), plus the original log-polar colour information (separated in red, green and blue channels), which will be all together denoted by vector  $f^{i,j}$ , and represents an iconic model instance. More explicitly:

$$\begin{aligned}
 f^{i,j} = & (Edges@0, Edges@30, \dots, Edges@330, \\
 & \pm Bars@0, \pm Bars@30, \dots, \pm Bars@150, \pm Blobs, \\
 & \pm Ends@0, \pm Ends@30, \dots, \pm Ends@330, red, green, blue) \quad (5.1)
 \end{aligned}$$

It would be possible to take advantage of a multi-resolution image analysis by extracting the image planes above at different retinal scales, like in [GF96] and [FM98], where scales of 1x, 1/2x and 1/4x were used. However, using several scales would incur higher computational costs, so we simplified the representation in our experiments to use just the original scale. The use of different scales could improve matching and the attention mechanism, but, as we mentioned before, these issues are not at the focal point of this thesis.

During the system's operation, model instances are matched against each other. We define an iconic model  $F_t$  of class  $t$  as the set of all model instances that are close to each other with respect to a similarity function  $Sm$ . More precisely, for some clustering threshold  $CTHD$ :

$$\begin{aligned}
 f^{i,j} \in F_t \iff & (\overline{Sm}(f^{i,j}, x_t) > CTHD) \wedge \\
 & (\overline{Sm}(f^{i,j}, x_t) > \overline{Sm}(f^{i,j}, y_{t'})) \quad (5.2)
 \end{aligned}$$

where  $\overline{Sm}(f^{i,j}, x_t)$  is the average similarity for all  $x_t \in F_t$  and  $\overline{Sm}(f^{i,j}, y_{t'})$  is the average similarity for all  $y_{t'} \in F_{t' \neq t}$ .

Any pair of model instances belonging to a given model are related through a similarity value. When determining this value, the matching algorithm (explained in the following section) also

gives an estimate for the relative scale  $rS$  and orientation  $rO$  between these two instances. All these three measurements are stored in a table  $R_t$ , which describes the relationships between any pair of model instances of the same class  $t$ . This can be expressed as:

$$R_t = \{(Sm(f_p, f_q), rS(f_p, f_q), rO(f_p, f_q)) \mid f_p, f_q \in F_t\} \quad (5.3)$$

where  $p, q \in \{1, \dots, sizeof(F_t)\}$  indexes instances within the model.

### 5.1.3 Similarity Function

The matching between two image regions is implemented using the standard statistical cross-correlation coefficient between two signals, as presented by Gonzalez and Woods [GW92] for the case of 2-D images:

$$\rho(f, g) = \frac{1}{\sigma_f \sigma_g} \left( \sum_x \sum_y (f(x, y) - \bar{f}) (g(x, y) - \bar{g}) \right) \quad (5.4)$$

where  $f(x, y)$  and  $g(x, y)$  are the two images to be correlated;  $\bar{f}$  and  $\bar{g}$  are the means calculated over the identical image domains;  $\sigma_f$  and  $\sigma_g$  are the standard deviations defined by:

$$\sigma_f = \sqrt{\sum_x \sum_y (f(x, y) - \bar{f})^2} \quad (5.5)$$

$$\sigma_g = \sqrt{\sum_x \sum_y (g(x, y) - \bar{g})^2} \quad (5.6)$$

The function  $\rho(f, g)$  produces values within the continuous range of  $[-1, +1]$ . When the output is near  $+1$ , then there is good correlation, i.e. whenever one function increases, the other also does in proportion. When the output is near  $0$ , then the functions are uncorrelated, which means that there is no relation between changes in one function and the other. Finally, when the output is near  $-1$ , then the functions are anti-correlated, i.e. whenever one function increases, the other decreases in proportion.

In addition to being able to correlate a pair of image regions, we need also a similarity function  $Sm$  to compare pairs of iconic model instances as defined in Equation (5.1). Theoretical analysis has shown that an effective multivariate cross-correlation coefficient can be simply

the mean of the individual cross-correlation coefficients of the individual channels [FO95] (or image planes in our representation).

However, primal sketch feature planes have a different pixel distribution when compared to the image colour planes: primal sketch planes are usually sparser as they contain only a subset of intensity pixels responsible for representing the detected features (all the remaining pixels are null).

Consequently, a single differing pixel between a pair of feature planes will have a larger effect in decreasing the cross-correlation coefficient than a differing pixel between a pair of colour planes. In other words, the cross-correlation coefficients between pairs of primal sketch planes will tend to be much smaller than the cross-correlation coefficients between corresponding pairs of colour planes.

From the above, we decided to calculate the mean of the cross-correlation coefficients for feature and colour planes separately and then compute  $Sm$  as the average between the two. This is equivalent of assigning a weight of  $\frac{1}{100}$  to each individual feature plane and a weight of  $\frac{1}{6}$  to each colour plane, as there are 50 feature planes and only 3 colour planes. The following equation shows the calculation of  $Sm$ :

$$Sm(f^{i,j}, g^{i,j}) = \frac{1}{2} \left( \frac{1}{W} \sum_{p=1}^W \rho'(f_p^{i,j}, g_p^{i,j}) + \frac{1}{K-W} \sum_{p=W+1}^K \rho'(f_p^{i,j}, g_p^{i,j}) \right) \quad (5.7)$$

where  $f_p^{i,j}$  is the  $p^{\text{th}}$  component of vector  $f^{i,j}$  (similarly for  $g_p^{i,j}$ );  $W$  is the number of primal sketch image planes in  $f^{i,j}$  (and  $g^{i,j}$ ); and  $K$  is the total number of image planes in  $f^{i,j}$  (and  $g^{i,j}$ ),  $K - W$  is the number of colour planes ( $K = 53$  and  $W = 50$ ). As the likelihood of encountering inverse images (anti-correlated) is very small, we only considered the positive (and zero) values of the function  $\rho$  when computing  $Sm$ , i.e.,  $\rho' = \max(\rho, 0)$ .

Ideally, in order to obtain an optimal output from the similarity function, one would have to weight the image planes according to their importance to the matching process. It is reasonable to think that some image planes will be more important than others when matching a particular class of objects. It would be possible to determine these weights by some learning process or by experimentation, however this was left as a future work. An experiment showing the actual outputs of the similarity function  $Sm$  on a number of real images is presented in Section 5.3.1.

### 5.1.4 Clustering Algorithm

The position of the retina, and consequently any underlying local object feature, is obtained by simply looking at the interest map. But, what about the scale and orientation parameters? The log-polar map implemented in our retinal representation can help the process of figuring out the relative scale and orientations of an object component with regard to another previously stored object.

This is possible by translating the log-polar map in both radial and angular directions to inexpensively transform the object features into a number of new orientations and scales, which are then matched against already learnt occurrences of feature instances for all the existing models. The best match will possibly be with the model the feature is represented by. Then the feature together with its position, relative orientation and scale are stored.

The relative position of features is only relevant to features found in a same scene image. Whenever two or more features consistently appear at the same relative distance to each other within a significant set of images, this provides some evidence that they are somehow related to each other, possibly as part of a same rigid object. This sort of evidence is used in the model construction process described in the next section. On the other hand, scales and orientations are measured relative to features found in any arbitrary scene image.

Since the system's architecture operates by analysing unlabelled image regions, we need to perform unsupervised learning of object features (or components). In other words, we need to build clusters of model instances that are similar to each other in order to be able to correctly construct geometric models. Algorithm 5.1 gives a straightforward solution to clustering. Without loss of generality this can be used as a prototype for designing more efficient algorithms.

This algorithm was based on a variation of the nearest neighbour classifier called the minimum distance classifier [DH73], with two main differences. Firstly, unknown patterns are incorporated into their corresponding classes after being classified. Secondly, instead of storing the mean for each class (or prototype) to allow a single distance computation per class, we store all training instances and compute the mean distance between an unknown pattern and all stored instances. This is intended to eliminate the inherent error associated with the mean class prototypes when incorporating noisy, misaligned or corrupted image data.

Algorithm 5.1: Clustering object features.

---

```

for each scene image  $S^i$  do
  for each foveation point  $P^{i,j}$  in the scene do
    obtain the  $j^{th}$  object feature  $f^{i,j}$  at position  $P^{i,j}$  by foveating the input scene and
    extracting primal sketch planes
    if the model base is empty then
      create a new model  $F_t$  and store  $f^{i,j}$  in it
    else
      generate a set of scaled and rotated versions  $f^{l,j}$  of  $f^{i,j}$ 
      find the model  $F_t$  that gives the highest average similarity score  $\bar{C}$  between its internal
      object features  $f^{k,l}$  and one of the  $f^{i,j}$  variations
      if  $\bar{C} > CTHD$  then
        insert  $f^{i,j}$  into  $F_t$ 
        insert into table  $R_t$ 
          the similarity scores  $Sm(f^{i,j}, f^{k,l})$ ,
          the relative scales  $rS(f^{i,j}, f^{k,l})$  and
          the relative orientations  $rO(f^{i,j}, f^{k,l}), \forall f^{k,l} \in F_t$ 
      else
        create a new model  $F_t$  and insert  $f^{i,j}$  into it
      end if
    end if
  end for
end for

```

---

In this algorithm, the process of generating a set of rotated and scaled versions of object components or features is accomplished by shifting the log-polar map in two directions. In Section 3.2.4 we showed that a rotation of  $k$  sectors in the retinal space is equivalent to a translation of  $k$  units in the polar axis (1 unit denotes the angle between any two adjacent receptive field sectors), whereas a change in scale of  $B^p$  in the retinal space is equivalent to a translation of  $p$  units in the log axis (1 unit denotes  $\log(B)$ , with  $B \cong 1.1$ ).

Given a log-polar image representing a retina with 48 rings and 60 sectors, it would be very expensive to compute all the translational possibilities within this space. Moreover, not all scales would have sufficient data to allow discrimination. Therefore, we used only a subset of the possible variations in scale and orientation. We varied the first parameter from -8 to +8 rings in steps of 2, which means the detected relative scales will be within this set of discrete values:  $\{0.48, 0.58, 0.70, 0.83, 1, 1.20, 1.44, 1.72, 2.07\}$  (here, the actual  $B$  value of 1.095 was used). We varied the second parameter from 0 to 60 sectors in steps of 5, which yields to angles within this set of values:  $\{0, 30, \dots, 330\}$ .



To measure the similarity between any pair of features we use the similarity function  $S_m$  as defined in Equation (5.7). The clustering threshold  $CTHD$  is experimentally determined in Section 5.3.1.

Given there are  $I$  interest points to process, Algorithm 5.1 requires  $I(I - 1)/2$  full matchings to build clusters from these points, which means that the time complexity is quadratic in the number of interest points. We consider a full matching as a matching involving all the rotated and scaled versions of a given object feature. However, the space complexity is less critical since it is linear in the number of interest points.

A simple matching between two model instances takes about 3 seconds to compute using a 700MHz CPU. Considering that we do this at 8 different scales and 12 different orientations, it takes about 288 seconds to have a full matching score between any pair of model instances. So, if we have 10 images with an average number of 10 interest points per image to match pairwise, this gives 4950 combinations, which multiplied by 288 seconds would take a staggering 16.5 days to compute.

Thus, there is clearly a lot of room for optimisation with respect to matching and clustering. The utilisation of a faster CPU, parallel hardware, or the design of a distributed version of the algorithm would be good starting points for speeding up the computation time. Nevertheless, Algorithm 5.1 is still feasible and useful as a research tool in its current stage if we considered no more than half a dozen scene images with a few interest points (2-6) per image, as the processing time would be within the range of a few hours up to 2 days.

The models  $F_i$  created by Algorithm 5.1 and the corresponding  $R_i$  tables storing the feature relationships (relative scales, orientations and similarity scores) are used as input to the next and most important step of our approach to iconic model learning: inferring structure from the image evidence. Next section explains in detail how this can be accomplished using a graph-based representation that stores hypotheses about feature correspondences and rigid body constraints between sets of features.

## 5.2 Learning Structured Models via a Representational Graph

In this section we explain how to infer geometric relationships between object feature models learnt according to the process described in the previous section. A general principle adopted in this work is that the recognition of consistent geometric relations allows the inference of larger structural object models. For the purposes of this thesis, we assume that objects can only have 2-D similarity transformation: rotation, scaling and translation in one object component generates an (approximately) equal transformation in all the other components.

The approach adopted to solve the structure learning problem was to build a graph-based representation. Figure 5.2 gives an overview of the entire process which is an expansion of module (e) shown in Figure 5.1. The general aspects of the approach are discussed in this section with reference to the processes (e.1)-(e.7) shown in Figure 5.2, whereas the more detailed information is given in subsequent sections.

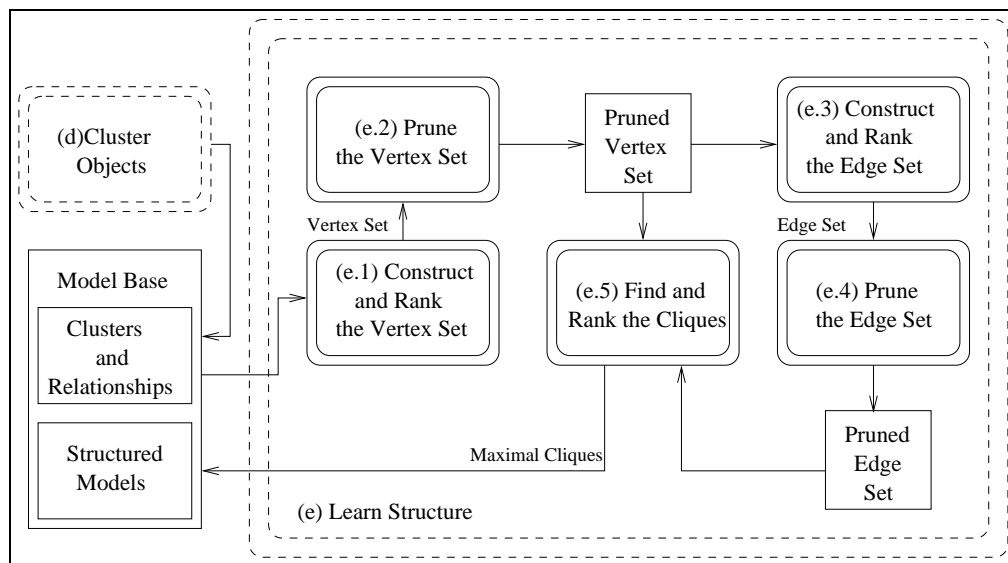


Figure 5.2: Processes (rounded boxes) and data (rectangular boxes) involved in building a relational graph from an iconic model database.

Vertices (e.1) are derived from the Cartesian product of the sets of object features of a given type found in each of the scenes and are ranked according to a function that uses the similarities between their internal component features. They represent hypotheses for the correspondences between object features of a same type across a set of scenes. Vertices representing poor hypothesis are eliminated (e.2). There is a mechanism embedded in the vertex creation that

allows for the possibility of missing features.

An edge (e.3) in the graph represents a hypothesis about the relationship between the features within the two connecting vertices and is ranked according to the compatibility between these two vertices. This compatibility is evaluated as a function of four main factors: (1) the relative scales and (2) the relative orientations within pairs of features of the connecting vertices; (3) the angles and (4) the lengths of the vectors defined by a pair of corresponding instance coordinates in a same image taken from the two connecting vertices.

Edges with low rankings or connecting inconsistent vertices are removed from the graph (e.4). The problem is then reduced to finding the maximal cliques of the graph (e.5). Details of the process are given below.

### 5.2.1 Vertices

Each object feature model  $F_t$  created by the algorithm described in Section 5.1 contains a set of instances  $f^{i,j}$  found in each scene image  $S^i$ . An initial problem found when designing an algorithm to learn relationships from these sets was to deal with the occurrence of multiple unrelated model instances of a same class in the scenes. In other words, how to account for the combinations of instances that appear at consistent positions and orientations and separate them from those who don't? We solve this problem by taking the complete combinatorial set of instances. Each N-tuple from this set will be a vertex in our graph. The set of all vertices  $V_t$  of a given feature type is the Cartesian product between the sets of instances  $F_t^i$  of type  $t$  found in scene  $i$ :

$$V_t = F_t^1 \times F_t^2 \times \dots \times F_t^N \quad (5.8)$$

where  $N$  is the total number of scene images analysed.

Equation (5.8) can be expanded as  $V_t = \{v_1, v_2, \dots, v_M\}$  where  $M$  is the total number of vertices. Each vertex  $v_r$  of type  $t$  can be expanded as  $v_r = (f_t^{1,j_1}, \dots, f_t^{N,j_N})$ , where  $j_k \in \{1, \dots, size(F_t^k)\}$ . As we will not need the positions  $k$  of the instance within an image, for the sake of simplicity, we are going to remove these indexes throughout this section, unless otherwise stated. Also, we are going to drop the type  $t$  for the feature model class, as from now on they will be distinguished by the letter representing the instance itself.

In order to cope with the possibility of model instances being missing in some images, as the attention mechanism might fail searching at some locations, occlusion might have happened and so on, we introduce a \* (wild-card) model instance, which is added to the sets of instances  $F^i$  found in scene  $i$  before computing the combinatorial sets that define the vertices.

### Vertex Ranking

If the similarity scores between pairs of vertex elements  $Sm(f^i, f^j)$  are thought as probabilities of those elements belonging to a same feature class, then a natural way of defining the rank of a vertex is by multiplying all the similarity scores. As a result, for a vertex to be strong all of its elements have to be very similar to each other. However, as the number of images analysed increases, the final rank tend to decrease exponentially toward zero since typical similarities will be within the open interval  $]0, 1[$ . One way to get around this problem is to use an exponential normalisation wrapper which is a function of the number of vertex combinations. This function just answers the simple question: if all similarities between vertices had the same value, what would that value be so that the final product would yield the actual vertex rank? Equation (5.9) shows the vertex ranking function used:

$$Rank(v_r) = \left( \prod_{i=1}^{N-1} \prod_{j=i+1}^N Sm(f^i, f^j) \right)^{\frac{2}{(N-W)(N-W-1)}}, \quad f^i, f^j \neq * \quad (5.9)$$

where  $N$  is the number of feature instances in the vertex ( $N$  is also the number of images analysed),  $W$  is the number of wild-cards in the vertex,  $Sm$  is the similarity function between two features  $f^i, f^j$  (values for this function are obtained via Algorithm 5.1, and are always within the range  $[0, 1]$ ). We do not consider any combination  $(f^i, f^j)$  that contains at least one wild-card.

It is possible to use many other approaches to obtain the rank of a vertex, like, for instance, averaging all the similarity scores  $Sm(f^i, f^j)$  between pairs of constituent instances. One clear difference between this approach and the previous one is that a low similarity pair of vertices would be less important to reducing the final vertex rank than when using the product approach. As the vertex ranks play a minor role in actually finding the structural models (they act mainly as a weight to the final clique scores) the investigation of different functions will be left as future work.

### Vertex Pruning

One side effect of the addition of the wild-card instances is that there will be now a number of vertices with many \*'s when compared to the number of real object features, which can cause relationships being learnt between loose features, or vertices that do not represent any plausible real objects. If there are  $M_i$  instances of feature type  $t$  in image  $i$ , then the number of vertices produced is  $\prod_{i=1}^N (M_i + 1)$ , where 0, 1 or 2 are typical values for  $M_i$ . To reduce the number of vertices, we allow only  $K$  \*'s per vertex (typically  $K=1$  or  $2$ ) during the node creation process.

Ideally, we want to satisfy the constraint  $K \ll N$  ( $N$  is the total number of images) to prevent the creation of vertices that do not have enough image evidence to support the presence / absence of features. If the number of images  $N$  is too small, e.g.  $N=2$ , allowing 2 or 1 wild-cards would create vertices that are all wild-cards or have only one non-wild-card feature. These vertices are inappropriate, since we cannot compute whether the feature represents a genuine relationship or is only coincidental during the edge creation process, which is described later on in this section. The same is true if  $N=3$  and we allow 3 or 2 wild-cards.

Thus, it is clear that there has to be an upper bound to the value of  $K$ . We define the maximum  $K$  as a function of the number of images analysed  $N$  in such a way that the number of wild-cards present in a vertex is never bigger than the number of real features, see Equation (5.10), except when the number of images is too small, i.e.  $N \leq 2$ , in which case we do not allow any wild-cards at all.

An obvious consequence of the above is that if an object feature is missing in more than  $K_{max}$  images, then the vertex creation process cannot represent that feature. Limiting  $K$  also reduces the combinatorial explosion of vertices. We do not threshold the vertex rankings as thresholding has already been applied when clustering features with Algorithm 5.1.

$$K_{max}(N) = \begin{cases} INT\left(\frac{N}{2}\right) & \text{if } N > 2 \\ 0 & \text{otherwise} \end{cases} \quad (5.10)$$

where function  $INT()$  truncates its argument downward to the nearest integer.

### 5.2.2 Edges

An edge  $e = (a, b)$  connects two compatible vertices  $a$  and  $b$  in the graph. Vertices  $a = (a^1, \dots, a^N)$  and  $b = (b^1, \dots, b^N)$  are compatible if for each pair of feature instances in different images  $(a^i, a^j)$  taken from the first vertex, which are related by a given scale and orientation  $W(a^i, a^j) = (rS(a^i, a^j), rO(a^i, a^j))$ , the corresponding pair  $(b^i, b^j)$  in the second vertex has its components related through an approximately equal relative scale and orientation  $W(b^i, b^j) = (rS(b^i, b^j), rO(b^i, b^j))$ . Moreover, each pair of feature instance coordinates  $(P_{a^i}, P_{b^i})$  and  $(P_{a^j}, P_{b^j})$  taken from the same vertex positions will define a pair of vectors  $Q(a^i, b^i) = (A(a^i, b^i), D(a^i, b^i))$  and  $Q(a^j, b^j) = (A(a^j, b^j), D(a^j, b^j))$ , which will be approximately the same when taking into account the feature's relative scales and orientations. Function  $A$  computes the vector angle and  $D$ , its length. The above concepts are illustrated in Figure 5.3.

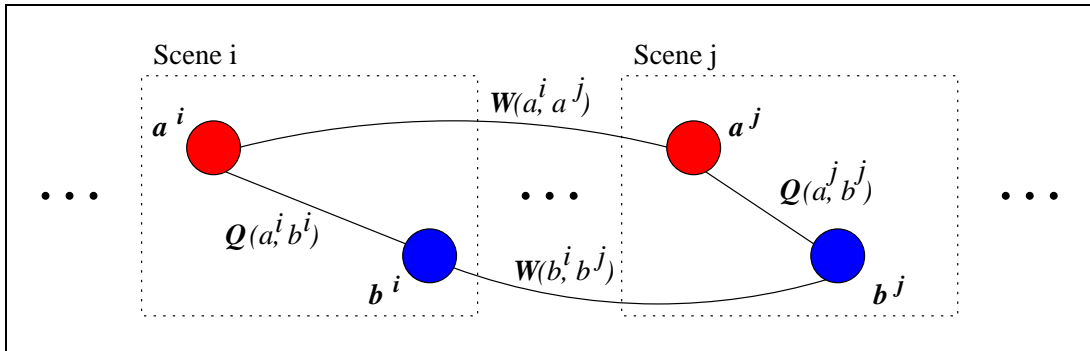


Figure 5.3: Relations between vertex components to form an edge.  $a = (a^1, \dots, a^i, \dots, a^j, \dots, a^N)$ ,  $b = (b^1, \dots, b^i, \dots, b^j, \dots, b^N)$  are vertices connected through an edge  $e(a, b)$ .  $W$  contains the relative scale and orientation between the two component features.  $W$  relates these components across images, whereas  $Q$  contains the angle and length of a vector linking two components from distinct features in the same scene image. Feature pairings  $(a^i, b^i)$ ,  $(a^i, a^j)$ ,  $(b^i, b^j)$  and  $(a^j, b^j)$  are the basis for computing the rank of an edge.

### Edge Ranking

From the previous paragraph is possible to conclude that the rank of an edge is defined as a function of four main quantities: (a) the relative scales and (b) the relative orientations within pairs of features of the connecting vertices; (c) the angles and (d) the lengths of the vectors defined by a pair of corresponding instance coordinates in a same image taken from the two connecting vertices. One of the simplest, yet powerful, ways of comparing these quantities is by using normalised absolute differences. Equation (5.11) shows how to compare the relative

scales of corresponding features  $(a^i, a^j), (b^i, b^j)$  found in two vertices  $a$  and  $b$ .

$$\Delta S^{i,j} = 1 - \frac{\text{abs}(rS(b^i, b^j) - rS(a^i, a^j))}{rS(b^i, b^j) + rS(a^i, a^j)} \quad (5.11)$$

Equation (5.12) shows the same for relative orientations, except that a normalisation function  $\hat{O}$  (Equation (5.13)) is now required to take into account the fact that orientations are measured in a closed circle (e.g. the angles  $359^\circ$  and  $1^\circ$  are actually very close to each other, but their absolute difference is high).

$$\Delta O^{i,j} = 1 - \frac{\hat{O}(\text{abs}(rO(b^i, b^j) - rO(a^i, a^j)))}{180} \quad (5.12)$$

$$\hat{O}(x) = \begin{cases} 360 - x, & \text{if } x > 180 \\ x, & \text{otherwise} \end{cases} \quad (5.13)$$

The angle  $A(a^i, b^j)$  of the vector defined by a pair of corresponding instance coordinates (from the same scene  $j$ ), taken from the two connecting vertices, is expected to be the same angle found in any other pair of instance coordinates (at another scene), apart from the rotation that each of the feature pairs might have suffered from one scene to another. Here we have to decide which feature pair gives the best estimate for the angle in the second scene, so we compute two differences (Equation (5.14)) and take the minimum between these differences, see Equation (5.15). Although normally both values are the same, we take the minimum because of the possibility of imprecise calculations at the earlier stages, for example due to noise or due to an attention mechanism failure to pick up a particular object feature. Note that the normalisation function  $\hat{O}$  has to be used again as orientations are compared..

$$\begin{aligned} dA_a^{i,j} &= \text{abs}(A(a^j, b^j) - (rO(a^i, a^j) + A(a^i, b^i))) \\ dA_b^{i,j} &= \text{abs}(A(a^j, b^j) - (rO(b^i, b^j) + A(a^i, b^i))) \end{aligned} \quad (5.14)$$

$$\Delta A^{i,j} = 1 - \frac{\text{MIN}(\hat{O}(dA_a^{i,j}), \hat{O}(dA_b^{i,j}))}{180} \quad (5.15)$$

Similarly to the angle comparison, the length of the vector connecting two features in an image should be preserved in any other image, apart from the change in scale that each of the feature

pairs might have suffered from one scene to another. Again, the feature pair that gives the best estimate for the scale on the second scene (Equation (5.16)) has to be chosen (Equation (5.17)). Note that this time, the normalisation factor (called here  $q^{i,j}$ ) depends on the minimum value that is chosen (Equation (5.18)).

$$\begin{aligned} dD_a^{i,j} &= \text{abs}(D(a^j, b^j) - (rS(a^i, a^j) \times D(a^i, b^i))) \\ dD_b^{i,j} &= \text{abs}(D(a^j, b^j) - (rS(b^i, b^j) \times D(a^i, b^i))) \end{aligned} \quad (5.16)$$

$$\Delta D^{i,j} = 1 - \frac{\text{MIN}(dD_a^{i,j}, dD_b^{i,j})}{q^{i,j}} \quad (5.17)$$

$$q^{i,j} = \begin{cases} D(a^j, b^j) + (rS(a^i, a^j) \times D(a^i, b^i)), & \text{if } dD_a^{i,j} < dD_b^{i,j} \\ D(a^j, b^j) + (rS(b^i, b^j) \times D(a^i, b^i)), & \text{otherwise} \end{cases} \quad (5.18)$$

As with the vertex ranking function, it is also reasonable to think the rank of an edge as the product of the intermediate ranks (or evidences) calculated above. The difference here is that we have four separate variables to combine, which give an estimate for the compatibility of the connected vertices in terms of scale, orientation and vector angle/length.

One could multiply the products of these variables all together to get a final rank, but the presence of a single null value among all combinations would cause the whole estimate to be zero. A more conservative approach would be to average these products, but the risk here would be to accept features that are completely out of place, with the wrong scale or orientation. Since we want our models to be as precise as possible and outliers may be interpreted as a missing features, which are already dealt with by the wild-card features, we decided to take the first of the above approaches, see Equation (5.19). As with the vertex function, a normalising wrapper is also used here to compensate for the fast rank decay when the number of feature combinations increases.

$$\text{Rank}(e) = \left( \prod_{i=1}^{N-1} \prod_{j=i+1}^N \Delta S^{i,j} \Delta O^{i,j} \Delta A^{i,j} \Delta D^{i,j} \right)^{\frac{2}{N(N-1) - nW(a,b)(2N - nW(a,b) - 1)}}, \quad a^i, a^j, b^i, b^j \neq * \quad (5.19)$$

where  $N$  is the number of scene images analysed;  $nW(a, b)$  is the number of wild-cards found in vertices  $a$  or  $b$  (Equation (5.21)) so that the exponent relates to the number of feature pairings



not containing wild-cards (more details about calculating this number are given in the edge pruning section below);  $\Delta S^{i,j}$ ,  $\Delta O^{i,j}$ ,  $\Delta A^{i,j}$  and  $\Delta D^{i,j}$  are calculated according to Equations (5.11), (5.12), (5.15) and (5.17), respectively. We simply discard any feature pairings that have at least one wild-card, since they cannot be computed based on real image evidence.

### Edge Pruning

The number of edges that can potentially be created from a set of vertices is quadratic in the number of vertices. We have already taken measures to keep the vertex set as small as possible, as discussed earlier. Moreover, the number of cliques, described in the following section, is exponential in the number of vertices and edges. Therefore, one must take care to prevent having an edge set that is too large.

We used four mechanisms to prune the edge space. The initial three act during the edge creation process. The first one eliminates edges that link pairs of vertices containing at least one common instance at the same feature position within the vertex list, as they cannot correspond to any real feature relationships. More formally, remove edge  $e = (a, b)$ , where  $a = (a^1, \dots, a^N)$  and  $b = (b^1, \dots, b^N) \iff \exists! i \cdot a^i = b^i$ .

The second mechanism ignores edges in which no pairing involving real features from the connecting vertices is possible (Figure 5.3 shows the role of feature pairings in creating edges). For instance, vertices  $(a^1, *, a^3)$  and  $(b^1, b^2, *)$  would not form an edge since all pairings of features used to compute the edge rank contain wild-cards, but vertices  $(a^1, a^2, *)$  and  $(b^1, b^2, *)$  would form an edge, since the pairings involving real features  $a^1, a^2, b^1, b^2$  are possible. Note that the vertex limitation on the number of wild-cards has not been violated in both cases above, however only the second pair of vertices would be suitable to form an edge.

This mechanism can be read as: if there is no real image evidence to support the relationship between two vertices, then do not create an edge involving those vertices. In mathematical terms, we define the logical predicate  $noRealPairing(a, b)$  that is true whenever no pairing involving real features from vertices  $a$  and  $b$  is possible:

$$noRealPairing(a, b) \iff N(N-1) = nW(a, b)(2N - nW(a, b) - 1) \quad (5.20)$$

where  $N$  is defined as before, the number of scenes analysed,  $a=(a^1, \dots, a^N)$  and  $b=(b^1, \dots, b^N)$  are the two vertices, and  $nW$  is a function defined by Equation (5.21):

$$nW(a, b) = \sum_{i=1}^N hasW(a^i, b^i) \quad (5.21)$$

where function  $hasW(x, y)$  returns 1 whenever feature  $x$  or feature  $y$  is a wild-card and 0 otherwise, see Equation (5.22):

$$hasW(x, y) = \begin{cases} 1 & \text{if } (x = *) \vee (y = *) \\ 0 & \text{otherwise} \end{cases} \quad (5.22)$$

Function  $nW$  counts the number of wild-cards in  $a$  or  $b$ , so that the number of pairings involving wild-cards can be computed as  $(nW(a, b)(2N - nW(a, b) - 1))/2$ . The total number of possible pairings between the vertices is  $N(N - 1)/2$ . Thus, if these two numbers are the same, no real feature pairing between the two vertices is possible, which demonstrates how Equation (5.20) was formulated.

The third mechanism used during edge creation, thresholding, discards edges ranked below a given threshold, so that edges representing poor or wrong hypotheses are discarded. This threshold plays an important role in deciding the level of tolerance to outliers the system will have: choose a threshold that is too small and lots of outliers may get into the clique finding process causing an excessive number of models to be created; choose a threshold that is too high and good hypotheses may be thrown away.

The last mechanism is more sophisticated than the previous ones, as it deals with the equivalence of edges while taking into account the possibility of wild-card features. This mechanism is used only after all edges have been created and evaluated. Algorithm 5.2 shows how this last edge pruning mechanism is implemented.

The main idea behind the algorithm is to remove an edge whenever two equivalent edges occur. If the number of wild-cards in both edges are the same, then remove the edge with the smallest rank. If the number of wild cards differs, then remove the edge with more wild-cards. A final garbage collection step (not shown here) simply creates a new set of pruned edges by adding only the edges which were not marked for removal.

Algorithm 5.2: Pruning the set of edges  $E$ .

---

```

for  $i = 1$  to  $size(E) - 1$  do {Edge  $i$  to be compared with all remaining edges}
   $ei \leftarrow E(i)$ 
  for  $j = i + 1$  to  $size(E)$ ,  $ei \neq \text{NULL}$  do {See the remaining edges given  $ei$  is not marked for
  removal}
     $ej \leftarrow E(j)$ 
    if  $ej \neq \text{NULL}$  then { $ej$  is not yet marked for removal}
       $nWi \leftarrow nW(ei)$ 
       $nWj \leftarrow nW(ej)$ 
       $ri \leftarrow Rank(ei)$ 
       $rj \leftarrow Rank(ej)$ 
      if  $equiv(ei, ej)$  then {Check whether  $ei$  and  $ej$  are equivalent}
        if  $((nWi = nWj) \text{ and } (ri < rj))$  or  $(nWi > nWj)$  then {If wild-card numbers are
        the same, mark the lower ranking edge for removal, if different, mark the one with more
        wild-cards}
           $ei \leftarrow \text{NULL}$  {Mark edge  $ei$  for removal}
           $E(i) \leftarrow \text{NULL}$ 
        else
           $ej \leftarrow \text{NULL}$  {Mark edge  $ej$  for removal}
           $E(j) \leftarrow \text{NULL}$ 
        end if
      end if
    end if
  end for
end for

```

---

Two vertices  $v = (v^1, \dots, v^N)$ ,  $w = (w^1, \dots, w^N)$  are equivalent if they have the same or equivalent pairwise features:

$$equiv(v, w) \iff \forall i = 1, \dots, N \cdot (v^i = w^i) \vee (v^i = *) \vee (w^i = *) \quad (5.23)$$

Since the graph we build is undirected, we consider two edges  $e1 = (a1, b1)$ ,  $e2 = (a2, b2)$  as equivalent if at least one pairwise combination of their vertices is equivalent. More formally:

$$equiv(e1, e2) \iff (equiv(a1, a2) \wedge equiv(b1, b2)) \vee (equiv(a1, b2) \wedge equiv(b1, a2)) \quad (5.24)$$

### 5.2.3 Cliques

The processes described above constructs a graph  $G$  consisting of a set  $V$  of vertices, which represent hypotheses for the correspondences between object features of a same type across a set of scenes, and a set  $E$  of edges connecting pairs of unordered vertices, which represent hypotheses for the relationships between these features.

Given a graph  $G = (V, E)$ , a clique  $C$  in the graph  $G$  is a subset of  $V$  in which all vertices are pairwise adjacent, i.e.,  $\forall v \in C \cdot \forall (w \neq v) \in C \cdot (v, w) \in E$ . A *maximal* clique in  $G$  is a clique that is not a proper subset of any other clique, i.e., given  $CLIQUES$  is the set of all possible cliques in  $G$ :

$$isMaximal(C \in CLIQUES) \iff \forall (X \neq C) \in CLIQUES \cdot C \not\subseteq X \quad (5.25)$$

A *maximum* clique is a *maximal* clique of largest size (number of vertices). If we think about rigid objects, a maximal clique would therefore represent all possible geometric relationships between the features of an object and would be supported by the multiple occurrences of these features in a set of scenes. Thus, maximal cliques can be seen as geometric models for the rigid objects found in the scenes. It would be redundant to keep cliques that are not maximal since these are already included in the maximal cliques. The maximum clique(s) are not of special interest to us because they represent nothing more than the largest structured objects found (the ones with more features or components). Considering that we can potentially deal with an arbitrary set of distinct objects, the solution to the structure learning problem is not a single maximal clique, but a collection of them.

Although the clique finding problem has been shown to be *NP*-complete, there are many algorithms that use a variety of heuristics to achieve reasonable performance on arbitrary graphs (see [MPW98] and [Woo97] for some examples). As the performance of clique finding algorithms is not a central issue to this thesis, we decided to use one of the simplest approaches, which we adapted from the algorithm for graph matching by clique detection found in [Mes95]. This algorithm was designed to find only maximum cliques and has been adapted here to also find cliques of any size greater than 1. The original algorithm is closely related to the largest independent set detection given in [Chr75]. We perform the maximal clique detection in a separate process, which is a straightforward implementation of Equation (5.25).

Algorithm 5.3 finds cliques of a given size  $i$  by expanding the cliques of size  $i - 1$ . It takes as input a graph  $G = (V, E)$  and returns in the superset  $CLIQUES$  all the cliques found. In the first step,  $C$ ,  $CLIQUES$  and  $L(k) \forall k = 1, \dots, size(V)$  are initialised as empty. Then, a vertex  $v_a \in V$  is selected such that  $v_a$  has not been used before. In other words, if  $v_a \notin L(k)$  for any  $k \leq i$  then  $v_a$  is added to the current list  $L(i)$  so that it will not be used again for the same clique. In the following step, it is then checked whether  $v_a$  is directly connected to each of the vertices found in  $C$  such that  $C \cup \{v_a\}$  forms a clique. If so,  $C \cup \{v_a\}$  is added to the set  $CLIQUES$ . The search for cliques continues until for some size  $i$  no new vertex  $v_a$  can be added to  $C$ , in which case, the algorithm backtracks to the size  $i - 1$  by removing the  $i - 1$ -th vertex from  $C$ , emptying  $L(i)$  and decrementing  $i$  by one. As long as  $i$  is larger than one, it is possible to find new cliques and as soon as  $i$  becomes zero we know that all cliques have been successfully detected.

---

Algorithm 5.3: Finding all cliques in a graph  $G = (V, E)$ .

---

```

i ← 1 {Current search clique size}
C ← ∅ {Current clique found}
CLIQUES ← ∅ {Set of all cliques found}
L(k) ← ∅, ∀ k = 1, ..., size(V) {Set of vertices used for a given size}
while i > 0 do {Search for cliques of size i}
  if ∃ va ∈ V with va ∉ L(k), ∀ k ≤ i then
    L(i) ← L(i) ∪ {va} {Mark va as used for size i}
    if ∃ ea = (va, wa) ∈ E, ∀ wa ∈ C then {If va connects with all vertices in C}
      C ← C ∪ {va} {Expands C with va}
      if size(C) ≥ 2 then {Only cliques of size 2 or more are considered}
        CLIQUES ← CLIQUES ∪ {C} {Add C to the set of cliques found}
      end if
    i ← i + 1 {Search for cliques of the next size}
  else {Backtracks to size i - 1}
    remove the i - 1th vertex from C
    L(i) ← ∅
    i ← i - 1
  end if
end if
end while

```

---

After all cliques have been found using Algorithm 5.3, maximal cliques are extracted by simple cross-checking. Algorithm 5.4 details this process.

Algorithm 5.4: Extracting the maximal cliques.

---

```

MAXCLIQUES  $\leftarrow$   $\emptyset$  {Set of extracted maximal cliques}
for  $i = 1$  to  $size(CLIQUES)$  do {For all cliques found}
   $C \leftarrow CLIQUES(i)$ 
   $isMaximal \leftarrow TRUE$  {Maximality flag}
  for  $j = 1$  to  $size(CLIQUES)$ ,  $isMaximal = TRUE$  do {Select cliques  $T$  for testing with  $C$ 
  until all cliques have been examined or maximality requirement is broken}
    if  $j \neq i$  then {Prevent testing  $C$  with itself}
       $T \leftarrow CLIQUES(j)$  {Select clique  $T$  for testing}
      if  $(T \neq NULL)$  and  $(C \subseteq T)$  then { $T$  is maximal or not yet tested with  $C$ , and  $C$  is a proper
      subset of  $T$ }
         $isMaximal \leftarrow FALSE$  {Requirement for  $C$  being a maximal clique broken}
      end if
    end if
  end for
  if  $isMaximal = TRUE$  then { $C$  is not a proper subset of any other clique in  $CLIQUES$ }
     $MAXCLIQUES \leftarrow MAXCLIQUES \cup \{C\}$  {Insert  $C$  into the set of maximal cliques}
  else
     $CLIQUES(i) \leftarrow NULL$  {Discard  $C$  as it is not maximal}
  end if
end for

```

---

### Clique Ranking

The next stage is to rank the cliques. The rank of a clique is defined as the product of the mean vertex and edge ranks. In Equation (5.26) we used the same normalisation approach applied to the vertex and edge ranking functions, which consisted in raising the result to a power that is inversely proportional to the number of elements in the product.

$$Rank(CLIQUE) = (\bar{V} \times \bar{E})^{\frac{1}{2}} \quad (5.26)$$

where  $\bar{V}$  and  $\bar{E}$  are the mean vertex and edge ranks, detailed in Equations (5.27) and (5.28), respectively.

$$\bar{V} = \frac{1}{size(V)} \sum_{v \in V} Rank(v) \quad (5.27)$$

$$\bar{E} = \frac{1}{size(E)} \sum_{e \in E} Rank(e) \quad (5.28)$$

There is no indication that we need to prune the set of maximal cliques since the weak or

incorrect hypotheses will be discarded at the earlier edge pruning stage. The use of a clique threshold could be helpful to prune some unusual or unpredicted results, however, these have not been found in the experiments developed in this thesis.

### 5.3 Exploratory Experiments

The objective of this section is to show how the clustering and the structure learning algorithms presented in this chapter behave under a number of experimental conditions. The first question that we would like to answer is how much the matching scores of the clustering algorithm (given in Section 5.1.4) degrade with the misplacement of the fixation points, which are provided by an attention mechanism. It would also be interesting to know how / if the nature of the objects (highly structured / non-structured) and the kind of background (textured / non-textured) influence the results. The answer to the above questions is important as it shows what would be the tolerance of the clustering algorithm to a real attention mechanism.

Another important question is related to the sensitivity of the structure learning method to errors or deviations in the earlier processes, e.g., how the results degrade with misplaced fixation points or imprecisely detected scales and orientations? Moreover, it would be equally important to add the possibility of some of the object features being missed, in order to test the wild-card behaviour. Finally, one may want to know what is the sensitivity of the system to the size of the problem, in other words, what are the upper and lower bounds to which the system works satisfactorily?

#### 5.3.1 Misplacement of the Fixation Points

The first and simplest experiment that we performed was to evaluate how the similarity function behaves under the situation when a pair of model instances is not perfectly aligned. In other words, we wanted to know how far a fixation point could be from the actual model instance coordinates while maintaining a good matching score. The importance of this experiment is threefold: firstly, it provides an indication on how accurate an attention mechanism attached to the system has to be; secondly, it gives a rough idea about the threshold to be used when deciding whether a given model instance should be added to an existing iconic model or be used to create a new one; and, thirdly, it also indicates how stable the process is

going to be.

Since it would be an impossible task to perform this experiment with all imaginable kinds of images, we had to narrow down the input space by choosing two high level properties of the scenes that would be relevant for causing sensible changes to the output of the similarity function. We chose the complexity of the object under the focus of attention and the complexity of the background. A complex object is defined as an object that contains many structural components, like textures, edges, colours etc, which would make it more distinguishable from the others, whereas a simple object is the other way around. We distinguish between two kinds of background: plain, when it is essentially formed by an homogeneous colour or pattern, having little information to disturb the matching of a central object; and cluttered, when there is structure that eventually could trouble matching.

The experiment consisted of initially choosing a fixation point on the object and extracting a model instance there. Then, all possible fixation points inside a circle centred on the first point were used to generate neighbouring model instances, which were in turn compared to the first model instance according to the similarity metric. All the resulting similarities are represented along the y-axis of a 3-D graph, with the x-z axes representing local image coordinates inside the circle. Plots of the average similarities versus the radii of these circles are also presented.

Appendix B illustrates the four possible combinations between the above scene properties. Figure B.1 shows the experiment on a simple object, plain background. Figure B.2 shows the experiment on a simple object, cluttered background. Figure B.3 shows the experiment on a complex object, plain background. Finally, Figures B.4 and B.5 show the experiment on complex objects and cluttered backgrounds.

It is possible to draw two main conclusions from these figures. The first conclusion is that regardless the scene properties, on average the similarity scores went down to near 0.5 at about 3 pixels away from the central feature, which gives us some indication of the threshold *CTHD* to use in Algorithm 5.1. For instance, with a threshold of 0.5 and not considering the variations objects may suffer from one scene to another, we will still be able to correctly classify model features within a radius of 3 pixels away from the central feature. The second conclusion is that, point-wise, the similarity score is much more robust to the misplacement of the fixation points along the symmetry axis of the objects. For example, in Figures B.3 and



B.4, high scores (above 0.5) were obtained along the symmetry axis as far as 16 pixels away from the central feature.

### 5.3.2 Experiments with Synthetic Scenes

In order to study the behaviour of the structure learning approach in a way that is independent of the matching algorithm used, we synthetically generated a sort of “canonical” iconic model database derived from a set of four hypothetical scenes. Figure 5.4 illustrates these scenes.

We considered a class of objects formed by a triangular arrangement of three feature types or models,  $a$ ,  $b$  and  $c$ , represented in the figure by a hexagon (red), a square (blue) and a triangle (green), respectively. Scene 1 contains an object instance at its natural scale and orientation, whereas Scene 2 contains an object instance which was rotated  $180^\circ$  and scaled down by a factor of 0.5. Scene 3 contains two object instances, the first one rotated  $90^\circ$  anticlockwise and scaled up by a factor of 2, and the second one rotated  $270^\circ$  anticlockwise and scaled down by a factor of 0.5. Finally, Scene 4 is a version of Scene 1 which was rotated  $90^\circ$  clockwise and scaled up by a factor of 2.

Table 5.1 shows the iconic model database as if it was generated from the scenes in Figure 5.4. Each column contains a particular feature instance, represented by its label and location, plus the relative scales, orientations and similarity scores ( $rS, rO, Sm$ ) between the feature and all of its predecessors of a same type. The main diagonals of the sub-table are not shown as the edge construction process does not include self-connected vertices. The lower half diagonals are also not shown because they are symmetric to the upper half (we build a graph that is undirected).

#### How Structure Learning Works in Practice

Before running the entire system on real complex scenes, we would like first to demonstrate how the proposed structure learning approach works on a simpler case, so that the algorithms and representations can be more easily understood.

The edge threshold was initially set to an arbitrary small value (0.01). When presenting the vertex, edge and clique results, a sequential number (labelled “#”) is used to identify individual components in the table. Although the graph components will appear sorted by their ranks, the

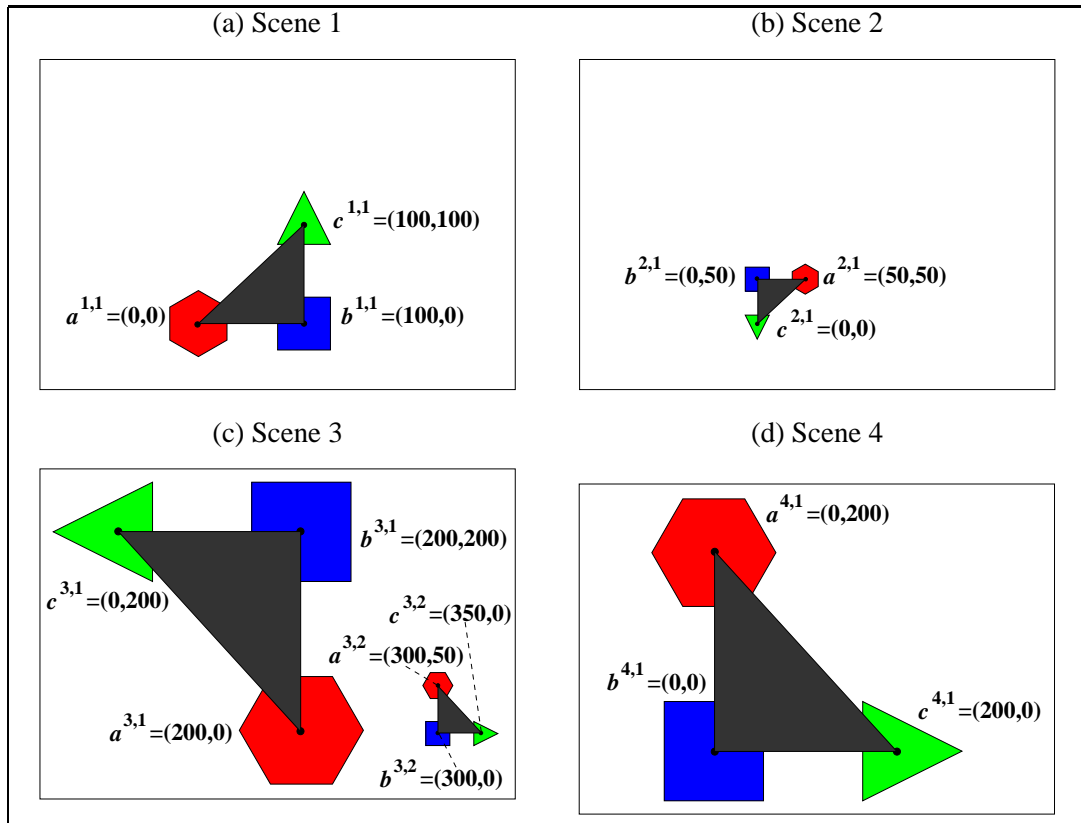


Figure 5.4: Four hypothetical scenes used to synthesise an iconic model database.

sole purpose of this sorting is to help analyse the results.

In an initial experiment, we simply ran the structure learning approach using the data from Table 5.1 as input. Regardless of the number of wild-cards allowed per vertex (we tested with 0, 1 and 2), the final output (Table 5.2) was always the same: the correct cliques representing the rigid objects present in the scenes.

Table 5.3 summarises the numbers of vertices, edges, pruned edges, raw cliques and maximal cliques involved when building the graph for each number of allowed wild-cards. The table also shows how many passes were required by the clique finding algorithm.

Note that, although the number of vertices and edges increases when the number of allowed wild-cards goes up, the number of pruned edges and vertices remains the same. This is a good indication that the edge pruning strategy is doing a good job at removing redundant edges, since the synthetic scenes do not have any missing features. There is also some additional evidence in support of that from the fact that the final number of maximal cliques did not

(a)	$a^{2,1}_{(50,50)}$	$a^{3,1}_{(200,0)}$	$a^{3,2}_{(300,50)}$	$a^{4,1}_{(0,200)}$
$a^{1,1}_{(0,0)}$	0.50,180,1.00	2.00, 90,1.00	0.50,270,1.00	2.00,270,1.00
$a^{2,1}_{(50,50)}$		4.00,270,1.00	1.00, 90,1.00	4.00, 90,1.00
$a^{3,1}_{(200,0)}$			0.25,180,1.00	1.00,180,1.00
$a^{3,2}_{(300,50)}$				4.00, 0,1.00

(b)	$b^{2,1}_{(0,50)}$	$b^{3,1}_{(200,200)}$	$b^{3,2}_{(300,0)}$	$b^{4,1}_{(0,0)}$
$b^{1,1}_{(100,0)}$	0.50,180,1.00	2.00, 90,1.00	0.50,270,1.00	2.00,270,1.00
$b^{2,1}_{(0,50)}$		4.00,270,1.00	1.00, 90,1.00	4.00, 90,1.00
$b^{3,1}_{(200,200)}$			0.25,180,1.00	1.00,180,1.00
$b^{3,2}_{(300,0)}$				4.00, 0,1.00

(c)	$c^{2,1}_{(0,0)}$	$c^{3,1}_{(0,200)}$	$c^{3,2}_{(350,0)}$	$c^{4,1}_{(200,0)}$
$c^{1,1}_{(100,100)}$	0.50,180,1.00	2.00, 90,1.00	0.50,270,1.00	2.00,270,1.00
$c^{2,1}_{(0,0)}$		4.00,270,1.00	1.00, 90,1.00	4.00, 90,1.00
$c^{3,1}_{(0,200)}$			0.25,180,1.00	1.00,180,1.00
$c^{3,2}_{(350,0)}$				4.00, 0,1.00

Table 5.1: Synthetic model database derived from Figure 5.4. Sub-tables (a), (b) and (c) show the internal ( $rS, rO, Sm$ ) relationships for feature instances of type  $a$ ,  $b$  and  $c$ , respectively. Beside each feature instance label is the position  $(x, y)$  where it has been found.

#	Clique	Rank
1	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.2: Cliques found in the first experiment on synthetic scenes.

Max *'s per Vertex	Vertices	Edges	Pruned Edges	Pruned Vertices	Clique Finding Iterations	Raw Cliques	Maximal Cliques
0	6	6	6	6	44	8	2
1	27	147	6	6	44	8	2
2	54	372	6	6	44	8	2

Table 5.3: Summary of first experiment on synthetic scenes, showing the sizes of individual graph components and iterations of the clique finding algorithm under a varying number of wild-card features per vertex.

change with the increase in wild-cards.

In a second experiment, we illustrate the wild-card role in representing missing features. Features  $a^{1,1}$ ,  $a^{2,1}$  and  $b^{2,1}$  were removed from the synthetic scenes of Figure 5.4 and from Table 5.1 as well. The resulting model base is represented in Table 5.4.

(a)	$a^{3,2}$ (300,50)	$a^{4,1}$ (0,200)	
$a^{3,1}$ (200,0)	0.25,180,1.00	1.00,180,1.00	
$a^{3,2}$ (300,50)		4.00, 0,1.00	

(b)	$b^{3,1}$ (200,200)	$b^{3,2}$ (300,0)	$b^{4,1}$ (0,0)
$b^{1,1}$ (100,0)	2.00, 90,1.00	0.50,270,1.00	2.00,270,1.00
$b^{3,1}$ (200,200)		0.25,180,1.00	1.00,180,1.00
$b^{3,2}$ (300,0)			4.00, 0,1.00

(c)	$c^{2,1}$ (0,0)	$c^{3,1}$ (0,200)	$c^{3,2}$ (350,0)	$c^{4,1}$ (200,0)
$c^{1,1}$ (100,100)	0.50,180,1.00	2.00, 90,1.00	0.50,270,1.00	2.00,270,1.00
$c^{2,1}$ (0,0)		4.00,270,1.00	1.00, 90,1.00	4.00, 90,1.00
$c^{3,1}$ (0,200)			0.25,180,1.00	1.00,180,1.00
$c^{3,2}$ (350,0)				4.00, 0,1.00

Table 5.4: Synthetic model database derived from Figure 5.4 after removing features  $a^{1,1}$ ,  $a^{2,1}$  and  $b^{2,1}$ .

Again, we varied the number of allowed wild-cards per vertex. In this experiment we will also confirm some of the hypothesis formulated in the first experiment by actually looking at the intermediate structures created. Table 5.5 summarises the experiment statistics.

Max *'s per Vertex	Vertices	Edges	Pruned Edges	Pruned Vertices	Clique Finding Iterations	Raw Cliques	Maximal Cliques
0	2	0	0	0	1	0	0
1	11	10	2	4	18	2	2
2	27	50	6	6	44	8	2

Table 5.5: Summary of the second experiment on synthetic scenes. Missing features  $a^{1,1}$ ,  $a^{2,1}$  and  $b^{2,1}$  under a varying number of wild-card features per vertex.

With zero wild-cards per vertex, only two vertices could be created: ( $c^{1,1}$ ,  $c^{2,1}$ ,  $c^{3,2}$ ,  $c^{4,1}$ ) and ( $c^{1,1}$ ,  $c^{2,1}$ ,  $c^{3,1}$ ,  $c^{4,1}$ ), both with scores of 1.00. This was already expected as there were no features of type  $a$  in scenes 1 and 2, and no features of type  $b$  in scene 2 to allow vertex formation with these feature types. Moreover, an edge between the two vertices found could not be created as they have common features.

The results for the case where one wild-card feature was allowed per vertex are detailed in Tables 5.6 through 5.9. Since object feature  $a$  requires two wild-cards to be properly represented in a vertex and we allowed no more than one wild-card per vertex, there is no vertex involving  $a$  in Table 5.6.

#	Vertex	Rank
1	$(b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00
2	$(b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
3	$(c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
4	$(c^{1,1}, c^{2,1}, c^{3,2}, *)$	1.00
5	$(c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
6	$(c^{1,1}, c^{2,1}, c^{3,1}, *)$	1.00

#	Vertex	Rank
7	$(c^{1,1}, c^{2,1}, *, c^{4,1})$	1.00
8	$(c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
9	$(c^{1,1}, *, c^{3,1}, c^{4,1})$	1.00
10	$(*, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
11	$(*, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.6: Vertices found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex. There is no vertex involving feature type  $a$  since this would require two wild-card features.

The principle of edge equivalency has worked properly when moving from Table 5.7 to 5.8. For instance, the edges pairs (1,2), (1,3), (1,4) and (1,5) in Table 5.7 are equivalent, therefore, since they all have the same rank, only the one with less wild-cards has been kept (edge 1). The same holds for edges (6,7), (6,8), (6,9) and (6,10), where edge 6 was the one chosen to be kept. Since the new set of pruned edges does not relate all of the initial vertices and we are not interested in cliques of size 1, a later vertex pruning process takes place to remove the orphan vertices. Therefore only the four vertices seen in Table 5.8 are kept. Only two cliques could be found, and these were already maximal (see Table 5.9).

#	Edge	Rank
1	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, *)$	1.00
3	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, *, c^{4,1})$	1.00
4	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
5	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (*, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
6	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
7	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, *)$	1.00
8	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, *, c^{4,1})$	1.00
9	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, *, c^{3,1}, c^{4,1})$	1.00
10	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (*, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.7: Edges found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex.

Tables 5.10 through 5.15 detail the results for the case when two wild-card features were

#	Edge	Rank
1	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.8: Edges after equivalency pruning in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex.

#	Clique	Rank
1	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.9: Cliques found in the second experiment on synthetic scenes when allowing no more than one wild-card feature per vertex.

allowed per vertex. Now we can see that the two vertices needed for representing feature type  $a$  appear in the set (vertices 1 and 2 in Table 5.10). The number of edges is much larger than the previous experiments because the number of vertices also increased.

Among the edges found, which are partially shown in Table 5.11, it is possible to identify the ideal candidate hypothesis to represent the object's structure: edges 1, 3, 7, 9, 13 and 25. Edge pruning indeed kept exactly those ideal candidates (Table 5.12). Note that only edges with real image feature pairings were created in Table 5.11.

Since in this experiment all edges have the same ranking (no noise has been added yet), edge 1 from Table 5.11 was not removed as it had less wild-cards than the only edge it was equivalent to (edge 2). Edge 3 had also less wild-cards than any other edge it was equivalent to, specifically edges 4, 5 and 6. The same can be said about the other four edges that won the equivalency test, edges 7, 9, 13 and 25. The six edges that remained after pruning are presented in Table 5.12 and the corresponding set of pruned vertices is given in Table 5.13.

The set of all cliques found is in Table 5.14. It is easy to see that cliques 1 and 2 are supersets of all the remaining cliques, which implies they are maximal, as indeed Algorithm 5.4 has detected (Table 5.15). The missing occurrences of feature  $a$  (scenes 1 and 2) and  $b$  (scene 2) are correctly represented in these two cliques. Moreover, there are two cliques because scene 3 contains 2 object instances and therefore the model can be explained in two different ways.

#	Vertex	Rank
1	$(*, *, a^{3,2}, a^{4,1})$	1.00
2	$(*, *, a^{3,1}, a^{4,1})$	1.00
3	$(b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00
4	$(b^{1,1}, *, b^{3,2}, *)$	1.00
5	$(b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
6	$(b^{1,1}, *, b^{3,1}, *)$	1.00
7	$(b^{1,1}, *, *, b^{4,1})$	1.00
8	$(*, *, b^{3,2}, b^{4,1})$	1.00
9	$(*, *, b^{3,1}, b^{4,1})$	1.00
10	$(c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
11	$(c^{1,1}, c^{2,1}, c^{3,2}, *)$	1.00
12	$(c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
13	$(c^{1,1}, c^{2,1}, c^{3,1}, *)$	1.00
14	$(c^{1,1}, c^{2,1}, *, c^{4,1})$	1.00
15	$(c^{1,1}, c^{2,1}, *, *)$	1.00
16	$(c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
17	$(c^{1,1}, *, c^{3,2}, *)$	1.00
18	$(c^{1,1}, *, c^{3,1}, c^{4,1})$	1.00
19	$(c^{1,1}, *, c^{3,1}, *)$	1.00
20	$(c^{1,1}, *, *, c^{4,1})$	1.00
21	$(*, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
22	$(*, c^{2,1}, c^{3,2}, *)$	1.00
23	$(*, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
24	$(*, c^{2,1}, c^{3,1}, *)$	1.00
25	$(*, c^{2,1}, *, c^{4,1})$	1.00
26	$(*, *, c^{3,2}, c^{4,1})$	1.00
27	$(*, *, c^{3,1}, c^{4,1})$	1.00

Table 5.10: Vertices found in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex. Now it is possible to see two vertices involving feature type  $a$  (vertices 1 and 2).

### Perturbation Model

We defined a perturbation model to alter the measurements of Tables 5.1 and 5.4 in order to evaluate the tolerance of the structure learning approach to deviations in the previous stages of the system. We discuss here the range of variations the system should tolerate (or to be robust to) in the light of the parameters chosen for the model. Finally we apply this model to the tables and repeatedly run the structure learning algorithms to evaluate the system's behaviour.

The perturbation model  $\mathcal{P}$  consists in essentially applying random noise  $R(\mu, \sigma)$ , uniformly distributed between  $[\mu - \sigma, \mu + \sigma]$ , to the four measurements produced by Algorithm 5.1: (a) the feature coordinates, (b) the relative scales and (c) orientations, and (d) the similarities relative to the other features of a same type in the model base. Since in all cases the parameter  $\mu$  will be zero or the measurement itself, the model parameters will be only the four  $\sigma$ 's associated with the above measurements, formally:  $\mathcal{P}(\sigma^{(a)}, \sigma^{(b)}, \sigma^{(c)}, \sigma^{(d)})$ .

In Section 5.3.1, we discovered that the similarity metric only starts to present noticeable degradation (values under 0.5) when features are misplaced 3 or more pixels away from the central feature coordinates. Therefore, it would be sensible to expect the structure learning algorithms to be tolerant to that level of variation. For this reason, noise with  $\sigma = 3$  and  $\mu = 0$  was added to the feature coordinates. If  $P(x, y)$  is a feature coordinate, then we have

#	Edge	Rank
1	$(*, *, a^{3,2}, a^{4,1}) (b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00
2	$(*, *, a^{3,2}, a^{4,1}) (*, *, b^{3,2}, b^{4,1})$	1.00
3	$(*, *, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
4	$(*, *, a^{3,2}, a^{4,1}) (c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
5	$(*, *, a^{3,2}, a^{4,1}) (*, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
6	$(*, *, a^{3,2}, a^{4,1}) (*, *, c^{3,2}, c^{4,1})$	1.00
7	$(*, *, a^{3,1}, a^{4,1}) (b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
8	$(*, *, a^{3,1}, a^{4,1}) (*, *, b^{3,1}, b^{4,1})$	1.00
9	$(*, *, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
10	$(*, *, a^{3,1}, a^{4,1}) (c^{1,1}, *, c^{3,1}, c^{4,1})$	1.00
11	$(*, *, a^{3,1}, a^{4,1}) (*, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
12	$(*, *, a^{3,1}, a^{4,1}) (*, *, c^{3,1}, c^{4,1})$	1.00
13	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
14	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, *)$	1.00
15	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, *, c^{4,1})$	1.00
16	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
17	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, *, c^{3,2}, *)$	1.00
18	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, *, *, c^{4,1})$	1.00
19	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (*, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
20	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (*, *, c^{3,2}, c^{4,1})$	1.00
21	$(b^{1,1}, *, b^{3,2}, *) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
22	$(b^{1,1}, *, b^{3,2}, *) (c^{1,1}, c^{2,1}, c^{3,2}, *)$	1.00
23	$(b^{1,1}, *, b^{3,2}, *) (c^{1,1}, *, c^{3,2}, c^{4,1})$	1.00
24	$(b^{1,1}, *, b^{3,2}, *) (c^{1,1}, *, c^{3,2}, *)$	1.00
25	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
⋮	⋮	⋮
48	$(*, *, b^{3,1}, b^{4,1}) (c^{1,1}, *, c^{3,1}, c^{4,1})$	1.00
49	$(*, *, b^{3,1}, b^{4,1}) (*, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
50	$(*, *, b^{3,1}, b^{4,1}) (*, *, c^{3,1}, c^{4,1})$	1.00

Table 5.11: Edges found in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex.

#	Edge	Rank
1	$(*, *, a^{3,2}, a^{4,1}) (b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00
2	$(*, *, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
3	$(*, *, a^{3,1}, a^{4,1}) (b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
4	$(*, *, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
5	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
6	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.12: Edges after equivalency pruning in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex.



#	Vertex	Rank
1	$(*, *, a^{3,2}, a^{4,1})$	1.00
2	$(*, *, a^{3,1}, a^{4,1})$	1.00
3	$(b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00

#	Vertex	Rank
4	$(b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
5	$(c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
6	$(c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.13: Vertices kept after edge pruning in the second experiment on synthetic scenes when allowing no more than two wild-card features per vertex.

#	Clique	Rank
1	$(*, *, a^{3,2}, a^{4,1}) (b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(*, *, a^{3,1}, a^{4,1}) (b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
3	$(*, *, a^{3,2}, a^{4,1}) (b^{1,1}, *, b^{3,2}, b^{4,1})$	1.00
4	$(*, *, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
5	$(*, *, a^{3,1}, a^{4,1}) (b^{1,1}, *, b^{3,1}, b^{4,1})$	1.00
6	$(*, *, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00
7	$(b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
8	$(b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.14: Cliques found in the second experiment on synthetic scenes when allowing no more than two wild-card feature per vertex.

#	Clique	Rank
1	$(*, *, a^{3,2}, a^{4,1}) (b^{1,1}, *, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	1.00
2	$(*, *, a^{3,1}, a^{4,1}) (b^{1,1}, *, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	1.00

Table 5.15: Maximal cliques from the second experiment on synthetic scenes when allowing no more than two wild-card feature per vertex. These two cliques represent exactly the expected models for the data.

$$P(x, y) \Leftarrow P(R(x, 3), R(y, 3)).$$

As emphasised in Section 5.1.4, relative scales are estimated by shifting the ring axis of the log-polar map to generate a sequence of candidate matchings. A  $p$  ring translation in the log-polar space represents a change in scale of  $B^p$  in the retinal (Cartesian) space. If we expect a uncertainty of  $\pm 2$  rings and given that  $B \cong 1.1$ , then we will have to accept at most 21% error on the actual Cartesian scale estimate, which seems reasonable. Thus, we change a relative scale estimate  $rS$  in the following way:  $rS \Leftarrow rS \times B^{R(0,2)}$ . A more realistic simulation could use discrete ring numbers chosen from a distribution instead of continuously varying random ring numbers, since the ring number are always discrete. We preferred the second form to maximise data variability within the uncertainty range.

Relative orientations are also estimated via translations in the log-polar space: a translation of  $k$  units in the polar axis corresponds to a rotation of  $k$  sectors in the retinal space. A variation of 1 sector corresponds to the characteristic angle between any two adjacent sectors, which is exactly 6 degrees if considering the retinal parameters used in this thesis. If we allow a  $\pm 5$  sectors variation, this will correspond to a maximum of  $\pm 30$  degrees absolute error in the relative orientation estimation  $rO$ , which seems acceptable when comparing to the angular resolution of the primal sketch feature detectors (also 30 degrees). Thus, we have:  $rO = \hat{O}(R(rO, 30))$ , where function  $\hat{O}$  is used to keep orientations within a closed circle (see Equation (5.13) for further details). A more realistic simulation could use discrete orientations that were multiple of 6, instead of continuously varying random orientations. We preferred again the second form to maximise data variability.

Finally, we explain how to alter the similarities between pairs of features. Given a similarity  $Sm$ , one could simply assign a new similarity in the form  $R(Sm, \sigma)$ , but similarities have to be restricted to the interval  $[0, 1]$ . To deal with this problem, we use a generate-and-test approach: iteratively generate outputs to the function  $R(Sm, \sigma)$  until we obtain a value within the range  $[0, 1]$ . The value of  $\sigma$  is a function of the matching error we want the system to be tolerant to. As we are accepting a 3 pixels top error in the feature's location and, according to Appendix B, the worst similarities will be around 0.5 under this level of misplacement, which represents 50% of the top matching score, we chose  $\sigma = 0.5 \times Sm$ .

From the above discussion, we reached the conclusion that the model should have the following

parameters:  $\mathcal{P}(3,2,30,0.5)$ . The results of one application of this set of parameters to Table 5.1 are presented in Table 5.16.

(a)	$a^{2,1}$ (51,48)	$a^{3,1}$ (197,-2)	$a^{3,2}$ (298,51)	$a^{4,1}$ (-1,200)
$a^{1,1}$ (1,-2)	0.43,199,0.86	2.40, 75,1.00	0.54,292,0.99	1.89,270,0.89
$a^{2,1}$ (51,48)		3.36,269,0.77	0.94, 91,0.93	4.02, 89,0.61
$a^{3,1}$ (197,-2)			0.28,193,0.59	1.10,196,0.60
$a^{3,2}$ (298,51)				4.14, 28,0.53

(b)	$b^{2,1}$ (0,48)	$b^{3,1}$ (202,198)	$b^{3,2}$ (299,-2)	$b^{4,1}$ (1,0)
$b^{1,1}$ (102,-1)	0.57,190,0.80	2.35,106,0.89	0.47,294,0.70	1.65,246,0.70
$b^{2,1}$ (0,48)		3.66,287,0.77	1.02,118,0.91	4.21, 71,0.83
$b^{3,1}$ (202,198)			0.22,191,0.79	0.88,181,0.86
$b^{3,2}$ (299,-2)				3.81, 0,0.61

(c)	$c^{2,1}$ (-2,2)	$c^{3,1}$ (-1,199)	$c^{3,2}$ (350,1)	$c^{4,1}$ (198,2)
$c^{1,1}$ (102,97)	0.59,207,0.68	2.29, 90,0.89	0.50,252,0.71	1.93,257,0.53
$c^{2,1}$ (-2,2)		4.62,286,0.72	1.14,105,0.83	3.63, 90,0.70
$c^{3,1}$ (-1,199)			0.29,152,0.51	1.13,179,0.64
$c^{3,2}$ (350,1)				4.79,358,0.80

Table 5.16: Results of one application of  $\mathcal{P}(3,2,30,0.5)$  to Table 5.1.

### Finding an Edge Threshold

In the previous experiments, we dealt with highest similarity, perfectly oriented and aligned object features. For this reason, the vertex, edge and clique ranks were all ranked the same (1.0). Thus, there was no need for using an edge threshold to prune the edge space.

Now, we will show that in more realistic situations, the edge threshold plays an important role in securing the correctness of the final cliques found. We also show through experimentation what would be a good edge threshold to use.

After feeding Table 5.16 into our structure learning algorithms, and using a null edge threshold, we discovered that there were unexpected maximal cliques among the cliques found due to the fact that some low ranking edges have survived through to the final stages of the approach.

Tables 5.17 and 5.18 show the actual edges and vertices after pruning and Table 5.19 shows the maximal cliques. Clearly, the low ranking edges 7 through 12 in Table 5.17 do not represent correct hypotheses for the features' relationships, and therefore should be removed. These edges are the cause of the unexpected maximal cliques 3 through 8 in Table 5.19. Note that,

although the edges associated with these cliques have low ranks, the final clique scores are still reasonably high. This is because the rank of a clique (Equation (5.26)) depends also on the vertex ranks, which in this case are high (Table 5.18).

#	Edge	Rank
1	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.80
2	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.79
3	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1})$	0.79
4	$(b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.78
5	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1})$	0.78
6	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.77
7	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.18
8	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1})$	0.14
9	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1})$	0.13
10	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.09
11	$(b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.08
12	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.08

Table 5.17: Edges created from perturbed model base in Table 5.16 when using a null edge threshold.

#	Vertex	Rank
1	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1})$	0.81
2	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1})$	0.78
3	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1})$	0.78

#	Vertex	Rank
4	$(b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1})$	0.75
5	$(c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.70
6	$(c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.69

Table 5.18: Vertices created from perturbed model base in Table 5.16 when using a null edge threshold.

Any edge threshold within the interval  $]0.18, 0.77]$  would have removed the undesired edges and consequently the incorrect maximal cliques as well. But what would be a good edge threshold to use in any situation? We believe there is no definite answer to this question since it depends on the level of the variation one may want the system to be tolerant or robust to.

Given the noise model discussed previously together with the noise parameters we regarded as acceptable, it is possible get some clues of an ideal edge threshold by actually looking at the boundary between the highest incorrect edge rank and the lowest correct edge rank in a number of runs of the structure learning approach on several perturbed model bases.

In a first experiment, we considered there was no missing feature in the model base (Table 5.1), applied the noise model  $\mathcal{P}(3,2,30,0.5)$  10 different times, and then ran the structure learning approach, under a varying number  $K$  of allowed wild-cards per vertex, on each of the 10

#	Clique	Rank
1	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.77
2	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.76
3	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.53
4	$(a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.51
5	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.50
6	$(a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (b^{1,1}, b^{2,1}, b^{3,2}, b^{4,1}) (c^{1,1}, c^{2,1}, c^{3,1}, c^{4,1})$	0.50
7	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,2}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.50
8	$(b^{1,1}, b^{2,1}, b^{3,1}, b^{4,1}) (a^{1,1}, a^{2,1}, a^{3,1}, a^{4,1}) (c^{1,1}, c^{2,1}, c^{3,2}, c^{4,1})$	0.49

Table 5.19: Maximal cliques created from model base in Table 5.16 when using a null edge threshold.

modified databases. Figure 5.5 plots the rank of the highest incorrect edges together with the rank of the lowest correct edges found in each of the 10 different runs. There is a separate plot for each value of  $K$ . In a second experiment, we considered the model base with 2 missing features of type  $a$  and one missing feature of type  $b$ , as shown in Table 5.4. We reproduced all the steps of the previous experiment, except that we used only  $K = 2$  wild-cards, since  $K = 0, 1$  would not properly represent the missing features. Figure 5.6 shows the graph resulting from the second edge threshold selection experiment.

By looking at the graphs it is possible to see that, among all runs, the safe interval for an edge threshold would be  $]0.2, 0.6]$ . Selecting a particular threshold within this interval is highly related to the error-reject trade off. Choosing a number that is too close to the lower bound may cause the system to be more tolerant to noise at the cost of increasing the chance of getting wrong edges, and consequently wrong cliques, into the system, whilst choosing a number that is too close to the upper bound may increase the accuracy but cause the rejection of good edge hypotheses. We chose a threshold of 0.5, which is slightly shifted toward the upper bound, as we prefer higher accuracy at the risk of some cliques not being detected in the end. In all the remaining experiments of this chapter, involving synthetic and real model bases, this threshold has proved to work satisfactorily.

### Tolerance to Noise

We applied  $\mathcal{P}(3,2,30,0.5)$  to Table 5.4 100 different times, producing new (perturbed) model databases which were then fed into the structure learning algorithms (allowing no more than  $K = 2$  wild-cards per vertex). We used the same edge threshold of 0.5 discussed above. Apart

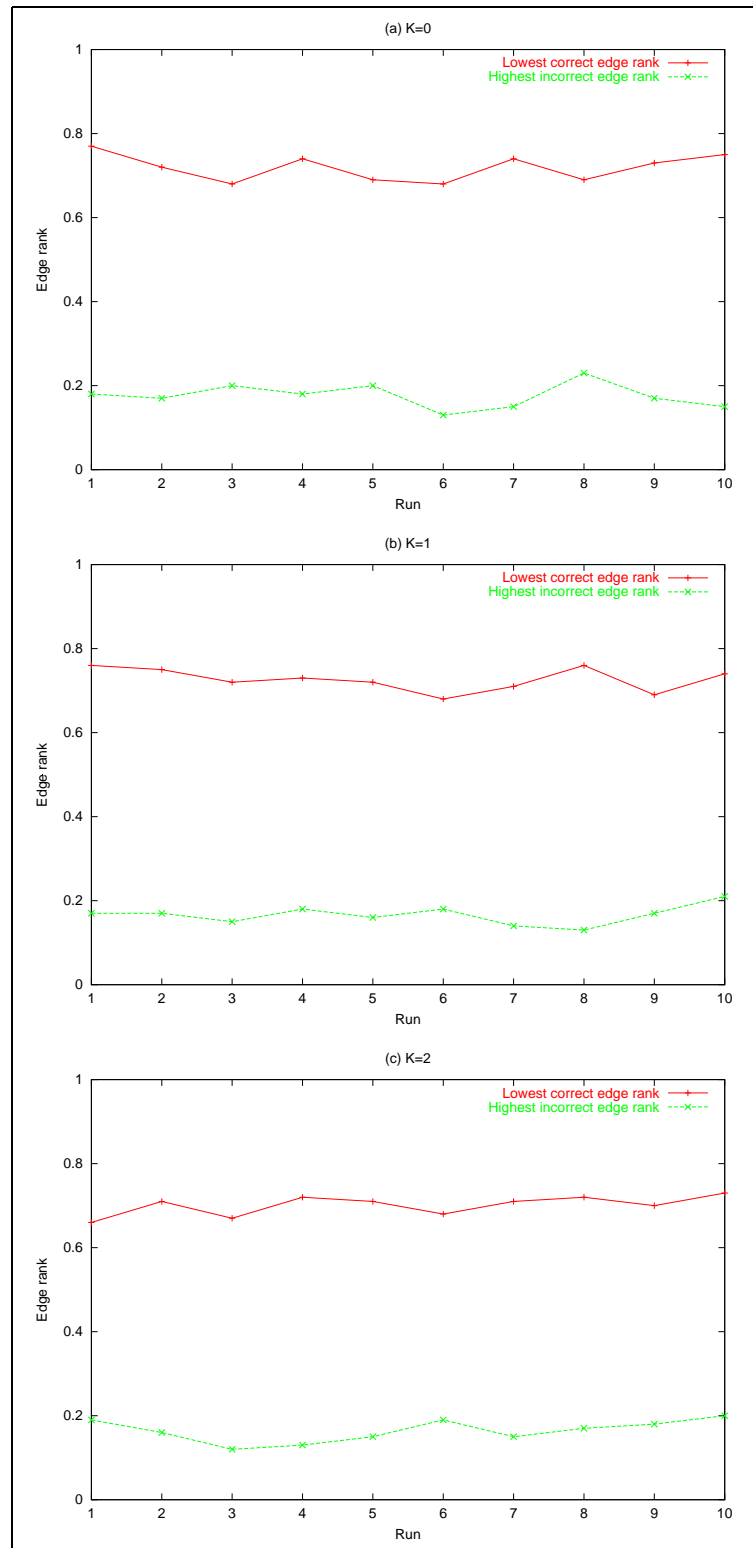


Figure 5.5: Finding an edge threshold with no missing feature. Plots of the ranks (y-axis) of the highest incorrect edges and lowest correct edges from 10 modified databases (x-axis) derived from Table 5.1. Different values of  $K$  were used: (a)  $K = 0$ , (b)  $K = 1$  and (c)  $K = 2$ .

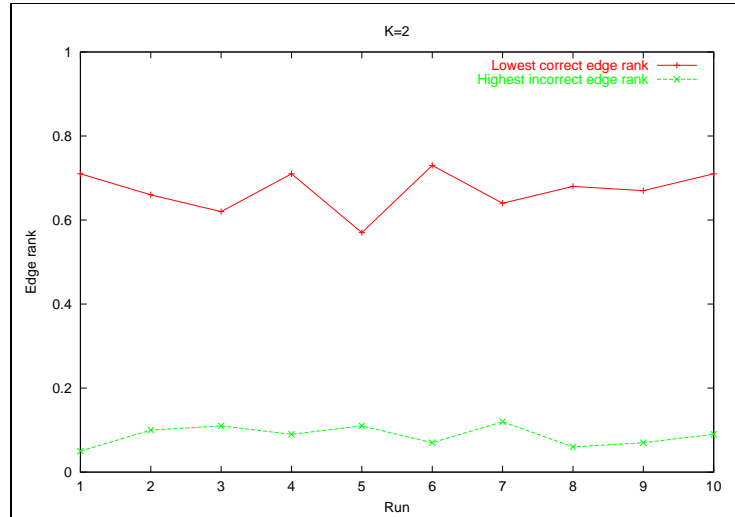


Figure 5.6: Finding an edge threshold with 2 missing features. Plots of the ranks (y-axis) of the highest incorrect edges and lowest correct edges from 10 modified databases (x-axis) derived from Table 5.1. A maximum of  $K = 2$  wild-cards was allowed per vertex.

from small variations in the ranks (no larger than 0.1), the resulting cliques were consistently correct in all of the runs, which showed a reasonable tolerance of the system to errors or imprecisions at the earlier stages.

### How far can we go?

The last important questions that we would like to answer in this section are related to the limitations of the proposed approach: What is the minimum number of images and features per image to allow proper learning of object relations? If we keep increasing the number of image features, at which point will combinatorial failure happen?

To answer the first question is straightforward and we do not need to perform any experiment. The edge construction process, by definition, requires at least 2 images with features in order to compute a non empty set of edges. Due to the constraint in Equation (5.10), no wild-card is allowed in this 2 image setup. One or more wild-cards are allowed only when using 3 or more images. There has to be at least two different feature types occurring in pairs in both images for an edge to be created. Although, cliques can be found when the number of images is small, they will be of little or no significance since there is limited image evidence to be considered by the vertex and edge ranking functions.

We give an answer to the second question which is based on experimentation. The theoretical complexity related to building a graph from a set of scenes and to processing this graph in order to find structured models has already been briefly discussed in the previous sections. If there are  $M_i$  instances of a feature per scene  $i$  of  $N$ , then the number of vertices that can be produced for that feature is  $\prod_{i=1}^N (M_i + 1)$ , where 0, 1 or 2 are typical values for  $M_i$ . Since we limit the number  $K$  of allowed wild-cards per vertex, the  $+1$  term above will be present only  $K$  times in the product. However, if  $M_i$  is greater than 1 in most scenes (due to multiple occurrences of a same feature), then there will be an explosion in the number of vertices. The number of edges that can potentially be created from a set of vertices is quadratic in the number of vertices and the number of cliques is exponential in the number of vertices and edges.

We designed and ran an experiment to analyse the combinatorics of our approach. We considered our initial scene setup as shown in Figure 5.4(a) and generated  $N - 1$  new scene configurations by transforming feature  $a^{1,1}$  with random translations, rotations and scales, and deriving the remaining features' location and properties in such a way that a rigid body relation between the features  $a$ ,  $b$  and  $c$  would hold.

Since we were only interested in the model base to be used by the structure learning approach, we did not actually generate images, just the associated object feature locations and properties. Next, we altered this model base according to our noise model  $\mathcal{P}(3, 2, 30, 0.5)$ , as explained before. Finally, we removed  $L = 0, 1, 2$  features from each feature type (randomly chosen among all images) while allowing  $K = 0, 1, 2$  wild-cards per vertex for each value of  $L$ .

If one prefers to think about the probability  $P_{missing}$  of a particular feature being missing instead of the actual number  $L$  of removed features, Equation (5.29) shows how to calculate this probability as a function of  $L$ . Thus, from Equation (5.29) we can deduce that for  $L = 0, 1, 2$ , we have  $P_{missing} = 0, 1/N, (2N - 1)/(N(N - 1))$ , respectively, where  $N$  is the number of scenes analysed. For example,  $N = 10$  yields  $P_{missing} = 0\%, 10\%, 21.1\%$ , respectively.

$$P_{missing}(L) = \sum_{i=0}^{L-1} \frac{1}{N-i} \quad (5.29)$$

We ran the structure learning approach 200 times for each set of parameters and computed the average statistics, which were then plotted in a set of graphs together with the number of scenes  $N$ , the number of removed features  $L$  and the number of allowed  $K$  wild-cards per



vertex. Besides the importance of getting statistically representative results, we needed to repeat the experiments a sufficient number of times to be able to get time measurements well above the operating system's clock resolution (around  $10^{-6}$  of a second), otherwise we would end up with null or unreliable measurements.

Appendix C contains all the graphs created. The graphs are distributed in 8 different figures, each plotting a variable against the number of images  $N$ . Each of the figures contains 3 graphs, (a), (b) and (c), which are associated with the number of allowed wild-cards per vertex  $K = 0, 1, 2$ , respectively. We used a special strategy to vary  $N$  when using a particular value of  $L$ : keep incrementing  $N$  until the total CPU (processor) time exceeds a given target (we used a fixed target of 3 seconds). Therefore, in most graphs, the value of  $N$  is different among the  $L$  curves. This was done to assure that all  $L$  curves would be plotted within the same time scale.

Figure C.1 shows the average CPU time in seconds spent by the entire approach (from vertex creation up to clique pruning) versus the number of images  $N$ . We ran on a 700MHz Athlon processor with 256MB of RAM under the Linux operating system. This time was calculated in such a way that it did not include the synthetic model base creation process, nor the noise generation process, nor the operating system's background processes running in time-sharing with our program.

In all three graphs shown, for a given  $N$ , the time is higher when no feature instance is removed from the scenes ( $L = 0$ ), followed by the cases when 1 and 2 feature instances are removed, in that order. This was expected since a missing feature either will not be represented (due to insufficient wild-cards) or will take a single wild-card; whereas a non-missing feature will cause two possible representations inside a vertex (itself and a wild-card), thus increasing the number of vertex combinations and consequently the number of edge combinations as well.

Figures C.4 and C.5, which show the average number of vertices and edges created, confirm the above observation. When  $K < L$ , there are not enough wild-cards in a vertex to represent the  $L$  missing features and therefore the average number of vertices shown in Figure C.4(a) (or edges in Figure C.5(a)) for the curves  $L = 1$  and  $L = 2$  is always zero, the same is true in Figure C.4(b) (or edges in Figure C.5(b)) for the curve  $L = 2$ .

The overall shape of the curves in Figure C.1 suggests the time complexity is exponential in the number of images. Indeed, if we look at Figure C.2, which shows the natural logarithm of

the CPU time, it is possible to see that as  $N$  increases the  $L$  curves converge to pure parallel lines, which is a good evidence to our hypothesis.

In order to know which parts of the approach were most contributing to the overall computation time, we separated the edge and vertex creation processes from the clique finding one. We discovered that most of the time was concentrated in the vertex and edge processes (including pruning), the clique finding process was taking a time so irrelevant to the entire process that, even after the 200 repetitions, this time was beyond the system's clock resolution. Among vertex and edge processes, the former contributed with just a tiny fraction of the total time. It is possible to see from the above that the exponential time increase with regards to the number of scenes is caused mainly by the required feature pairings when computing the edge rank.

We refrained from showing graphs with the average CPU time for edge and vertex creation since they are practically identical in shape (and in magnitude, for edges) to the ones in Figure C.1. Instead, we preferred to show the actual number of iterations of the clique finding algorithm, in order to demonstrate that their computation time had to be small, see Figure C.3. Firstly, the number of iterations proved to be invariant to the number of scenes, which indicates that the previous vertex and edge pruning processes did a good job at reducing the graph complexity. Moreover, regardless the values assumed by  $K$  and  $L$  the top number of iterations was never above 15.

The average number of raw cliques found was small for all values of  $K$  and  $L$  (see Figure C.8), which confirms the low number of iterations by the clique finding algorithm. In fact, given the structure of our 3-featured object used to create the synthetic model base, we were expecting to have 4 raw cliques when  $K \geq L$  (three cliques of size 2 and one clique of size 3), and the graphs in Figure C.8 show exactly that.

We carefully examined the two exceptional cases ( $N = 4, 5$ ) for the curve  $L = 2$  in Figure C.8(c), where the average number of cliques was smaller than 4. It turned out that this happened because  $L$  was close to  $N$ , which increased the probability of creating vertices that would not form an edge due to alternate occurrences of wild-cards within pairs of vertices. For instance, although vertices  $(a_1, *, a_3, *, a_5)$  and  $(*, b_2, *, b_4, b_5)$  have both a sufficient number of non-missing features, they cannot form an edge since no plausible edge rank can be calculated (see Equation (5.20) on page 113 and the referring text for more details about this issue).

The model base was built from exactly one structured object randomly translated and rotated across a number of scenes, therefore we expected to find exactly one maximal clique for all sets of parameters. Figure C.9 shows the average number of maximal cliques found. Apart from the exceptional cases above and the cases where  $K < L$ , this number was indeed one.

Since we were applying a noise model to every single synthetic model base generated, it is reasonable to accept that eventually the structure learning approach would not form the correct graph structure. In fact this can be verified by looking again at Figures C.4 through C.9: note that the average numbers of pruned vertices and edges, raw and pruned cliques suffer tiny variations, which is an evidence that in some cases the correct numbers have actually not been found.

It should be pointed out that finding the expected numbers of vertices, edges and cliques most of the time does not necessarily imply that these are the correct solution to the problem. Since the purpose of these experiments was mainly to investigate the combinatorial limitations of our approach, we did not implement any automatic method for checking the correctness of the structures created, instead, we simply inspected by hand a few tens of the individual experiments under different values of  $N$ ,  $L$  and  $K$  and confirmed that the expected structures and the correct solution were there.

A final remark is that the changes in the overall numbers of vertices and edges (Figures C.4 and C.5), when moving from graphs (a) toward (c), are nearly linear in relation to  $N$ . The only exception is for the curve  $L = 0$  when  $K = 2$  (Figures C.4(c) and C.5(c)), which looks more like exponential. Indeed, this difference is reflected in the  $L = 0$  CPU time plot of Figures C.1(c) and C.2(c).

In conclusion, if we suppose there will be some features missing from the images and we allow a sufficient number of wild-cards per vertex to represent them, the number of vertices, edges, and the clique finding process seem not to be a major limitation to the approach. However the same cannot be said about the computation time. In order to demonstrate what would be this time in the case of 2 missing features per vertex and 2 allowed wild-cards (Figure C.1(c), curve  $L = 2$ ), we fitted an exponential function to the original data and extrapolated this function to 30 images. The results of this experiment are presented in Figure C.10. The extrapolation says that the structure learning approach would spend approximately 15 hours (180 times slower

than before) to produce a result with just an addition of 12 extra scenes.

## 5.4 Structure Learning Case Studies

The best way to demonstrate how our approach works with real world scenes is by developing a number of case studies. In order to focus on the structure learning process, and to keep away from other aspects of the problem (like attention, lighting invariance, dealing with clutter and so on) which are not the main issue of this chapter, we tried to make the first real case study as simple as possible.

### 5.4.1 Telephone Unit

Three scene images (see Figure 5.7) were created from two top view pictures of a telephone handset and its base unit, respectively, taken against a black background using a digital camera. The camera was attached to a moving arm, perpendicular to the ground, with an incandescent point light source adjacent to the camera, so that images could be acquired at different distances from the objects under practically the same illumination conditions.

The two pictures were digitally placed inside a large black image under varying scales and orientations. In the first scene  $S^1$ , the handset and base were placed parallel to each other. In the second scene  $S^2$ , the handset was translated, rotated by  $90^\circ$  (counter-clockwise) and scaled down by a factor of 70% with respect to its first occurrence. Finally, in scene  $S^3$ , the base unit was scaled down by a factor of 60% of its original size and the handset was rotated by  $300^\circ$  (counter-clockwise) with respect to its occurrence in the first image.

A set of interest points was manually selected and passed to the system. These points consisted of: three pairs of central microphone/speaker positions within the telephone handsets  $\{a^{i,1}, a^{i,2}\}$ , three consistent 'led' positions  $\{b^{i,1}\}$ , three identifiable dark spots  $\{c^{i,1}\}$  and two corners  $\{e^{i,1}\}$  (missing when  $i = 3$ ) within the base unit. Two distraction points  $\{d^{1,1}, f^{2,1}\}$  not belonging to any distinguishable feature have also been defined across the first two scenes.

From this configuration of scenes, what we want our system to learn is that the handset and base units are each structured models, but because of the way we created our scenes (handset and base not obeying a rigid body transformation) they should not jointly form another structured

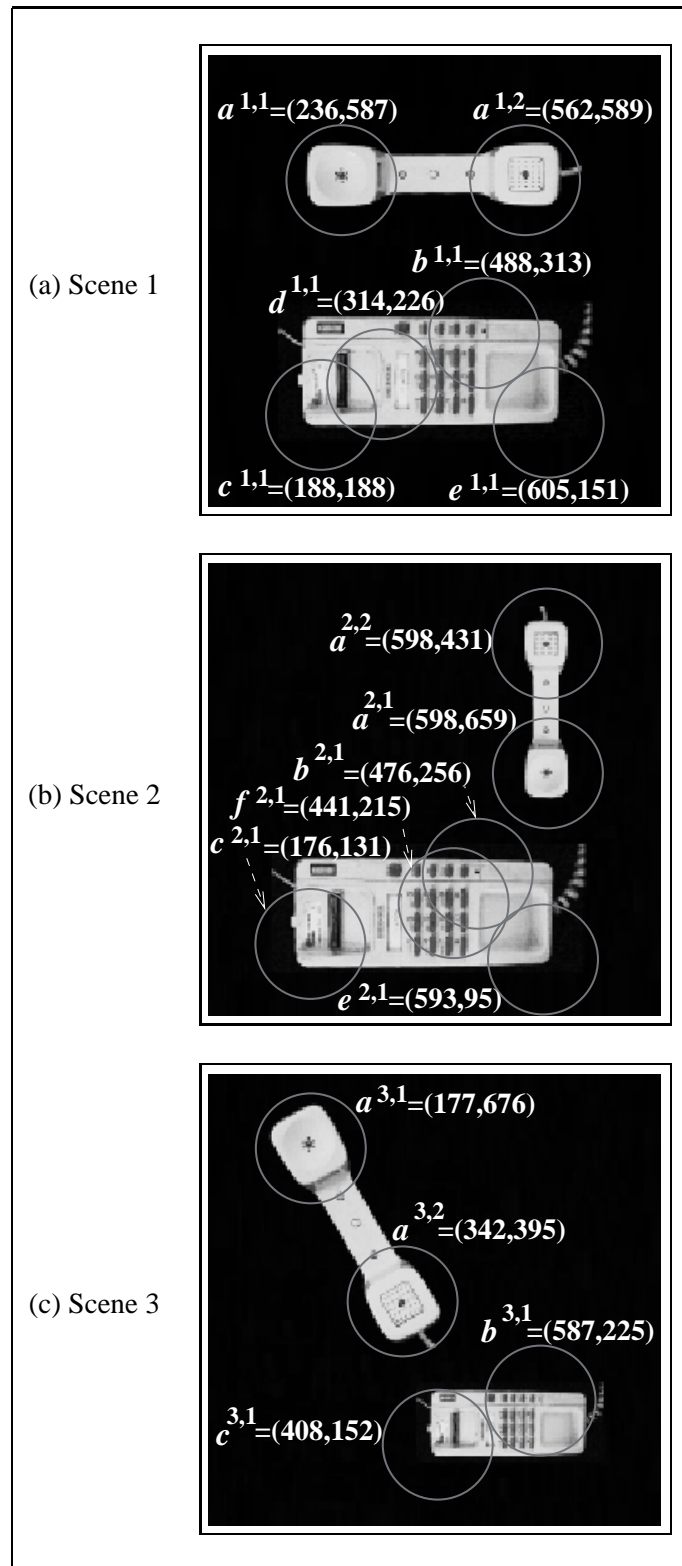


Figure 5.7: Scene images used for the telephone case study. Circles in the figure denote the retinal areas whose centres are at the interest points given to the system. The picture also shows the feature types  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ ,  $f$  obtained by Algorithm 5.1.

model. Additionally, the distraction features should not be taken into account and the missing occurrence of feature  $e$  in scene 3 has to be represented by a wild card in the vertex.

The first step was to apply the Algorithm 5.1, described in Section 5.1, to the set of interest points given. Note that the similarity scores were calculated from real foveations (producing colour and extracted primal sketch planes) of the actual objects in the scenes.

A clustering threshold of 0.5 has been used. The results are summarised in Table 5.20. Ground truth values for the  $(rS, rO, Sm)$  measurements are also presented in the table underneath each entry. Apart from the similarity scores  $Sm$ , all the other measurements were very close, if not identical, to the expected values. As one might expect, six different feature types have been automatically identified as a result of the algorithm.

(a)	$a^{1,2}$ (562,589)	$a^{2,1}$ (598,431)	$a^{2,2}$ (598,659)	$a^{3,1}$ (177,676)	$a^{3,2}$ (342,395)
$a^{1,1}$ (236,587)	1.00,180,0.66 <small>1.00,180,1.00</small>	0.70, 90,0.76 <small>0.70,90,1.00</small>	0.70,270,0.50 <small>0.70,270,1.00</small>	1.00,300,0.80 <small>1.00,300,1.00</small>	1.00,120,0.70 <small>1.00,120,1.00</small>
$a^{1,2}$ (562,589)		0.70,270,0.67 <small>0.70,270,1.00</small>	0.70, 90,0.73 <small>0.70,90,1.00</small>	1.00,120,0.68 <small>1.00,120,1.00</small>	1.00,300,0.83 <small>1.00,300,1.00</small>
$a^{2,1}$ (598,431)			1.00,180,0.51 <small>1.00,180,1.00</small>	1.44,210,0.80 <small>1.43,210,1.00</small>	1.44, 30,0.70 <small>1.43,30,1.00</small>
$a^{2,2}$ (598,659)				1.44, 30,0.62 <small>1.43,30,1.00</small>	1.44,210,0.63 <small>1.43,210,1.00</small>
$a^{3,1}$ (177,676)					1.00,180,0.72 <small>1.00,180,1.00</small>

(b)	$b^{2,1}$ (476,256)	$b^{3,1}$ (587,225)
$b^{1,1}$ (488,313)	1.00, 0,0.95 <small>1.00,0,1.00</small>	0.58, 0,0.83 <small>0.60,0,1.00</small>
$b^{2,1}$ (476,256)		0.58, 0,0.85 <small>0.60,0,1.00</small>

(c)	$c^{2,1}$ (176,131)	$c^{3,1}$ (408,152)
$c^{1,1}$ (188,188)	1.00, 0,0.88 <small>1.00,0,1.00</small>	0.70, 0,0.74 <small>0.70,0,1.00</small>
$c^{2,1}$ (176,131)		0.58, 0,0.73 <small>0.60,0,1.00</small>

(e)	$e^{2,1}$ (593,95)
$e^{1,1}$ (605,151)	1.00, 0,0.72 <small>1.00,0,1.00</small>

Table 5.20: Results of the Algorithm 5.1 on the telephone scenes. Sub-tables (a), (b), (c) and (e) present the relationship triplets  $(rS, rO, Sm)$  for feature instances of type  $a$ ,  $b$ ,  $c$  and  $e$ , respectively. The lower diagonals are not shown because they are symmetric. The ground truth relationship values are presented, using a small font, underneath each computed triplet. The feature types  $d$  and  $f$  have only one instance each and therefore were not included in the table. Their coordinates are as follows:  $d^{1,1}=(314,226)$ ,  $f^{2,1}=(441,215)$ .

We initially used the object feature models and relations from Table 5.20 to build a graph according to what is described in Section 5.2. By using an edge threshold of 0.5, which is consistent with the threshold chosen during the synthetic experiments in the previous section, it was possible to obtain five maximal cliques, which are listed in Table 5.21. No more than

$K = 1$  wild-cards were allowed per vertex. We did not test with  $K = 2$  as this would create vertices with more wild-cards than real image features (Section 5.2.1, page 109, discussed this issue).

#	Clique	Rank
1	$(b^{1,1}, b^{2,1}, b^{3,1}) (c^{1,1}, c^{2,1}, c^{3,1}) (e^{1,1}, e^{2,1}, *)$	0.88
2	$(a^{1,1}, a^{2,1}, a^{3,1}) (a^{1,2}, a^{2,2}, a^{3,2})$	0.87
3	$(a^{1,1}, a^{2,1}, a^{3,2}) (a^{1,2}, a^{2,2}, a^{3,1})$	0.83
4	$(a^{1,2}, a^{2,1}, a^{3,2}) (a^{1,1}, a^{2,2}, a^{3,1})$	0.82
5	$(a^{1,2}, a^{2,1}, a^{3,1}) (a^{1,1}, a^{2,2}, a^{3,2})$	0.81

Table 5.21: Maximal cliques found for the telephone scenes when adding no noise to the models.

Cliques 2-5, involving features of type  $a$ , indicate that the telephone handset features define a rigid geometric model governed by the relationships between the clique vertex components. The reason why there are four cliques describing the same geometric relation is because the handset features were classified as being of the same type, so they can be interchanged within a vertex without breaking the geometric constraint.

Note that the highest similarity score among these four cliques is for the correct case, which probably arises due to sight variations in the data, including the small segment of the telephone cord. The remaining clique (1) corresponds to structural model for the telephone base unit, which, as expected, contains a vertex representing, with a wild-card, the missing occurrence of feature  $e$  in the third image.

The next step was to add noise to the models created. We used the same noise level of  $\mathcal{P}(3,2,30,0.5)$  to disturb Table 5.20. The resulting models are presented in Table 5.22. After running the algorithms on the data of Table 5.22, and using a 0.5 edge threshold, the same five maximal cliques could be found, but this time with lower scores due to the noise addition (Table 5.23).

### 5.4.2 PDA and CD Player

In this section we explore a more complex case study. Instead of creating scenes by digitally pasting on a black image a number of previously acquired and segmented object images, we now acquire the entire scenes from real object configurations. In this case study, illumination and camera position were not controlled.

(a)	$a^{1,2}$ (562,587)	$a^{2,1}$ (600,431)	$a^{2,2}$ (595,657)	$a^{3,1}$ (179,676)	$a^{3,2}$ (341,395)
$a^{1,1}$ (233,586)	1.06,203,0.61	0.73, 67,0.43	0.62,259,0.47	0.86,271,0.50	1.01,138,0.55
$a^{1,2}$ (562,587)		0.65,293,0.64	0.73, 99,0.56	1.14,137,0.84	1.09,287,0.92
$a^{2,1}$ (600,431)			1.09,167,0.74	1.69,190,0.76	1.21, 18,0.76
$a^{2,2}$ (595,657)				1.28, 44,0.90	1.38,223,0.70
$a^{3,1}$ (179,676)					0.87,197,0.97

(b)	$b^{2,1}$ (477,256)	$b^{3,1}$ (586,225)
$b^{1,1}$ (485,310)	0.98,351,0.87	0.65,349,0.70
$b^{2,1}$ (477,256)		0.55,353,0.73

(c)	$c^{2,1}$ (176,129)	$c^{3,1}$ (409,149)
$c^{1,1}$ (185,185)	1.17,350,0.54	0.80, 0,0.94
$c^{2,1}$ (176,129)		0.52, 1,0.61

(e)	$e^{2,1}$ (591,97)
$e^{1,1}$ (606,149)	1.14, 15,0.80

Table 5.22: Results of perturbing Table 5.20 with noise parameters  $\mathcal{P}(3,2,30,0.5)$ .

#	Clique	Rank
1	$(e^{1,1}, e^{2,1}, *) (b^{1,1}, b^{2,1}, b^{3,1}) (c^{1,1}, c^{2,1}, c^{3,1})$	0.77
2	$(a^{1,2}, a^{2,2}, a^{3,1}) (a^{1,1}, a^{2,1}, a^{3,2})$	0.71
3	$(a^{1,2}, a^{2,1}, a^{3,2}) (a^{1,1}, a^{2,2}, a^{3,1})$	0.70
4	$(a^{1,2}, a^{2,1}, a^{3,1}) (a^{1,1}, a^{2,2}, a^{3,2})$	0.69
5	$(a^{1,2}, a^{2,2}, a^{3,2}) (a^{1,1}, a^{2,1}, a^{3,1})$	0.67

Table 5.23: Maximal cliques found for the telephone scenes when using a noise modified model database.

Two objects, a portable computer (PDA) and a CD player, were placed on top of a large blue canvas at different positions in a room indirectly illuminated by fluorescent lamps in the ceiling.

This time, we used a hand held digital camera to manually get an approximate top view of the scenes, which allowed a small (but noticeable) degree of perspective distortion, motion blur, shadows, specular reflections and focusing problems. Figure 5.8 shows the three scenes acquired under these experimental conditions.

A total of five interest points per scene was selected as follows: in the CD player, one point centred on the open-lid button  $a^{i,1}$ , another  $b^{i,1}$  on a black spot at lid centre and a final point  $c^{i,1}$  on a logo at the back of the lid; in the PDA, one point centred on the activity light  $d^{i,1}$  and another on the on-off button  $e^{i,1}$  ( $i$  is the scene number).

From scene 1 to 2 the CD player was rotated by approximately 90 degrees, while the PDA suffered no change in orientation. The distance from the camera to both objects did not suffer



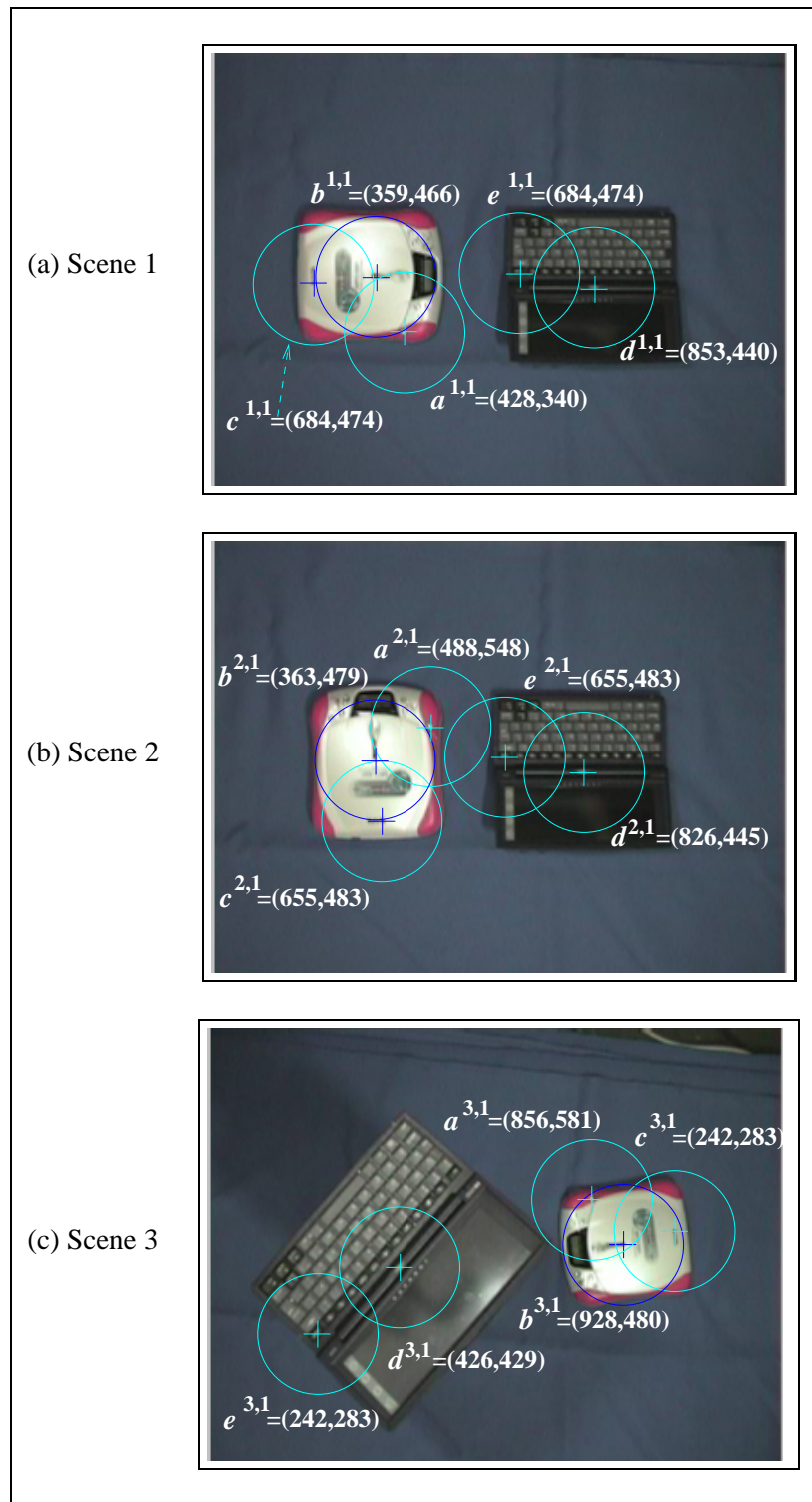


Figure 5.8: Scene images used for the 2<sup>nd</sup> case study. Circles in the figure denote the retinal areas whose centres are at the interest points given to the system. The picture also shows the feature types  $a$ ,  $b$ ,  $c$  and  $d$  obtained by Algorithm 5.1.

any noticeable change as well. But in scene 3, the camera was positioned further away from the objects so that the CD player appeared as having about 85% of its original size. The PDA, on the contrary, was elevated from the ground in such a way that it appeared as having about 144% of its previous size. The PDA was rotated about 60 degrees and the CD player was rotated about 185 degrees with respect to its occurrence in the first scene.

After running Algorithm 5.1 on the above interest points, we obtained the model database presented in Table 5.24, which is an approximation to the correct clusters of object features. The approximate ground truth values are presented underneath each cell using a small font size.

Note that a low clustering threshold, equal to 0.2, had to be used this time so that the algorithm could cope with the lower matching scores due to the arbitrary lighting conditions and free camera positioning. This threshold was actually easy to determine, since the similarities for features not expected to match were sufficiently low (0.05, on average).

The overall estimated relative scales and orientations were not so close to the ground truth for the scenes when comparing with the first experiment (Table 5.20). Regardless of these imperfections created by the clustering algorithm under more realistic scene conditions, it will be possible to see below that the structure learning approach can still infer the correct objects' structure from the data.

(a)	$a^{2,1}$ (488,548)	$a^{3,1}$ (856,581)	(b)	$b^{2,1}$ (363,479)	$b^{3,1}$ (928,480)
$a^{1,1}$ (428,340)	1.00, 90, 0.56 <small>1.00, 90, 1.00</small>	0.83, 180, 0.33 <small>0.85, 185, 1.00</small>	$b^{1,1}$ (359,466)	1.00, 90, 0.74 <small>1.00, 90, 1.00</small>	0.83, 180, 0.21 <small>0.85, 185, 1.00</small>
$a^{2,1}$ (488,548)		0.83, 90, 0.32 <small>0.85, 90, 1.00</small>	$b^{2,1}$ (363,479)		0.83, 90, 0.22 <small>0.85, 90, 1.00</small>
(c)	$c^{2,1}$ (381,337)	$c^{3,1}$ (1041,509)	(d)	$d^{2,1}$ (826,445)	$d^{3,1}$ (426,429)
$c^{1,1}$ (226,450)	1.00, 90, 0.44 <small>1.00, 90, 1.00</small>	0.83, 180, 0.43 <small>0.85, 185, 1.00</small>	$d^{1,1}$ (853,440)	1.00, 0, 0.84 <small>1.00, 0, 1.00</small>	1.20, 60, 0.24 <small>1.44, 60, 1.00</small>
$c^{2,1}$ (381,337)		0.83, 90, 0.38 <small>0.85, 90, 0.38</small>	$d^{2,1}$ (826,445)		1.20, 60, 0.25 <small>1.44, 60, 1.00</small>
(e)	$e^{2,1}$ (655,483)	$e^{3,1}$ (242,283)			
$e^{1,1}$ (684,474)	1.00, 0, 0.37 <small>1.00, 0, 1.00</small>	1.20, 60, 0.26 <small>1.44, 60, 1.00</small>			
$e^{2,1}$ (655,483)		1.44, 60, 0.19 <small>1.44, 60, 1.00</small>			

Table 5.24: Model database for the 2<sup>nd</sup> experiment involving a PDA and a CD player. Ground truth relationship values are presented, using a small font, underneath each computed triplet ( $rS, rO, Sm$ ).

This time we applied a modified set of noise parameters to the model database, since there was already enough variation in some of the estimated data, mainly the similarity scores and the relative scales. Thus, the new set of parameters used was  $\mathcal{P}(3,1,30,0.1)$ . Table 5.25 contains the new model database perturbed by these parameters.

(a)	$a^{2,1}$ (490,549)	$a^{3,1}$ (858,580)	(b)	$b^{2,1}$ (364,479)	$b^{3,1}$ (925,477)
$a^{1,1}$ (430,337)	0.98, 61,0.60	0.81,188,0.33	$b^{1,1}$ (356,465)	0.98,106,0.71	0.78,185,0.20
$a^{2,1}$ (490,549)		0.80, 90,0.34	$b^{2,1}$ (364,479)		0.86,114,0.24
(c)	$c^{2,1}$ (382,337)	$c^{3,1}$ (1041,508)	(d)	$d^{2,1}$ (825,442)	$d^{3,1}$ (426,429)
$c^{1,1}$ (227,451)	0.99,113,0.48	0.80,205,0.43	$d^{1,1}$ (853,441)	0.95,357,0.78	1.15, 75,0.23
$c^{2,1}$ (382,337)		0.81, 90,0.35	$d^{2,1}$ (825,442)		1.16, 81,0.27
(e)	$e^{2,1}$ (652,485)	$e^{3,1}$ (240,285)			
$e^{1,1}$ (686,473)	1.03, 28,0.39	1.13, 81,0.26			
$e^{2,1}$ (652,485)		1.44, 39,0.19			

Table 5.25: Results of perturbing Table 5.24 with noise parameters  $\mathcal{P}(3,1,30,0.1)$ .

After running the structure learning approach on the perturbed models (Table 5.25), allowing 0 or 1 wild-card per vertex, the two expected maximal cliques were found:  $(c^{1,1}, c^{2,1}, c^{3,1})$   $(a^{1,1}, a^{2,1}, a^{3,1})$   $(b^{1,1}, b^{2,1}, b^{3,1})$ , with a rank of 0.56 ; and  $(d^{1,1}, d^{2,1}, d^{3,1})$   $(e^{1,1}, e^{2,1}, e^{3,1})$ , with a rank of 0.47.

The following step was to remove two of the features,  $b^{3,1}$  and  $d^{1,1}$  from the model database in Table 5.24 and apply the noise parameters again, in order to inspect wild-card behaviour. When allowing 1 wild-card per vertex, it was possible to get the expected maximal cliques using the previous 0.5 edge threshold, see Table 5.26.

#	Clique	Rank
1	$(b^{1,1}, b^{2,1}, *)$ $(c^{1,1}, c^{2,1}, c^{3,1})$ $(a^{1,1}, a^{2,1}, a^{3,1})$	0.67
2	$(e^{1,1}, e^{2,1}, e^{3,1})$ $(*, d^{2,1}, d^{3,1})$	0.42

Table 5.26: Maximal cliques found in the PDA / CD player experiment with features  $b^{3,1}$  and  $d^{1,1}$  missing.

### 5.4.3 Limitation of the Object Matching

One of our initial thoughts about autonomous learning of structured iconic models was that a space variant sensor could help matching features under a cluttered or complex background,

i.e., if the high resolution centre of the sensor was occupied mostly by the object component then the low resolution differing background would have a small impact in the matching score.

However, in practice, not all object features will fall into the sensor in such a way to occupy most of the high resolution units. Elongated objects, like a screw driver for instance, do not have this property, but a round face with a fixation point at the nose, eye or mouth would probably have.

Moreover, most attentional strategies rely upon the extraction of several low level features including corners, edges, blobs etc, and these features are not necessarily located at the objects centre of mass.

Thus, it is likely that consistently located fixation points will be somewhere in the object boundaries, corners, bright or dark spots. We chose interest points to this chapter's experiments having in mind the above considerations.

The main question now is: can we actually demonstrate the above limitation with an example? The answer is affirmative. Figure 5.9 shows the object CD player under different backgrounds. We selected 2 object features per scene: one centrally located ( $a^{i,1}$ ) and another peripherally located ( $b^{i,1}$ ), where  $i = 1, 2, 3$  is the scene number.

Table 5.27 presents the matching scores for two separate clustering experiments (*CTHD* was set to 0) involving the two kinds of interest points mentioned above.

(a)	$a^{2,1}$ (335,260)	$a^{3,1}$ (329,220)	(b)	$b^{2,1}$ (315,196)	$b^{3,1}$ (371,196)
$a^{1,1}$ (307,208)	0.70, 180, 0.54 0.68, 180, 1.00	0.48, 270, 0.26 0.49, 260, 1.00	$b^{1,1}$ (336,297)	0.70, 180, 0.21 0.68, 180, 1.00	0.70, 270, 0.05 0.49, 260, 1.00
$a^{2,1}$ (335,260)		0.70, 90, 0.31 0.72, 80, 1.00	$b^{2,1}$ (315,196)		0.58, 90, 0.13 0.72, 80, 1.00

Table 5.27: Matching scores for (a) central,  $a^{i,1}$ , and (b) peripheral,  $b^{i,1}$  interest points. Truth values are presented using a small font, underneath each table cell.

From the matching scores and relative scales in the table, it is possible to see that central interest points ( $a^{i,1}$ ) are much more reliable than peripheral ones ( $b^{i,1}$ ), i.e., the matching scores are higher and the estimated relative scales are closer to the scenes' true relative scales.

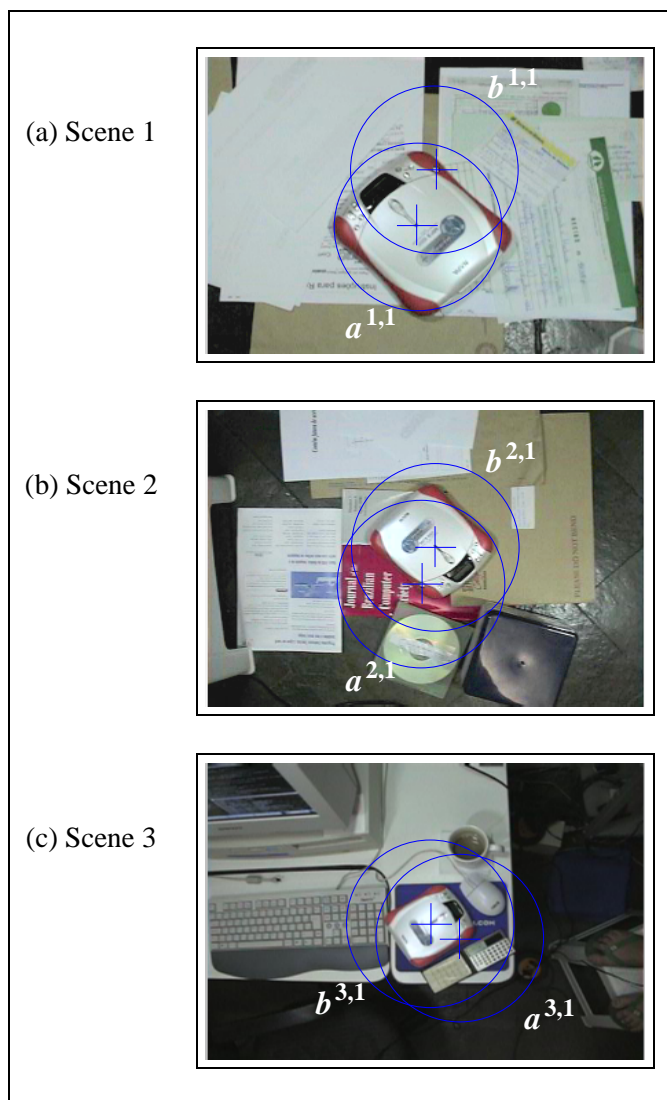


Figure 5.9: Images used to test the matching scores of (a) central,  $a^{i,1}$ , and (b) peripheral,  $a^{i,2}$ , interest points.

#### 5.4.4 Doll

In the fourth and final experiment we consider an object with an arbitrary background, lighting conditions and camera position. As we could see from the previous discussion regarding object matching with a space variant sensor, some objects like faces can help the matching metric to be less affected by any differing peripheral background information. We are not considering here differences that may be noticeable in the main area of a face, e.g., due to glasses, a moustache, hair style and so on.

Figure 5.10 shows three scenes where the doll object is present. In the first scene, the object was pictured horizontally. In order to create a second doll instance, we rotated the original image 180 degrees and appended the result to the scene, since we did not have a second doll. In the second scene, the doll now appears standing up under a different background and at a size that is about 63% of its size in the first scene. Finally, in the third scene, the doll is about 137% bigger than its first occurrence and again appears standing up under a different background.

Table 5.28 details the results of the clustering algorithm. Underneath each table cell are approximate truth values for the features' relationships. The matching scores were reasonably high, even though different backgrounds have been used.

It should be noted that a different model was created to describe each of the two eyes. This happened because the retinal mask is larger than the object features, so, when centred at one eye, it also included parts of the hair, mouth, nose and the other eye, making left and right eyes almost a mirror image of each other, therefore not producing a good match.

Table 5.29 shows the result of one application of  $\mathcal{P}(3,2,30,0.5)$  to Table 5.28.

The structure learning results for the perturbed model base are in Table 5.30. If we remove features  $c^{3,1}$  and  $b^{2,1}$  from Table 5.29 in order to test wild-card usage, then we have the results shown in Table 5.31. Both tables contain exactly the correct maximal cliques for the structure found in the scenes.

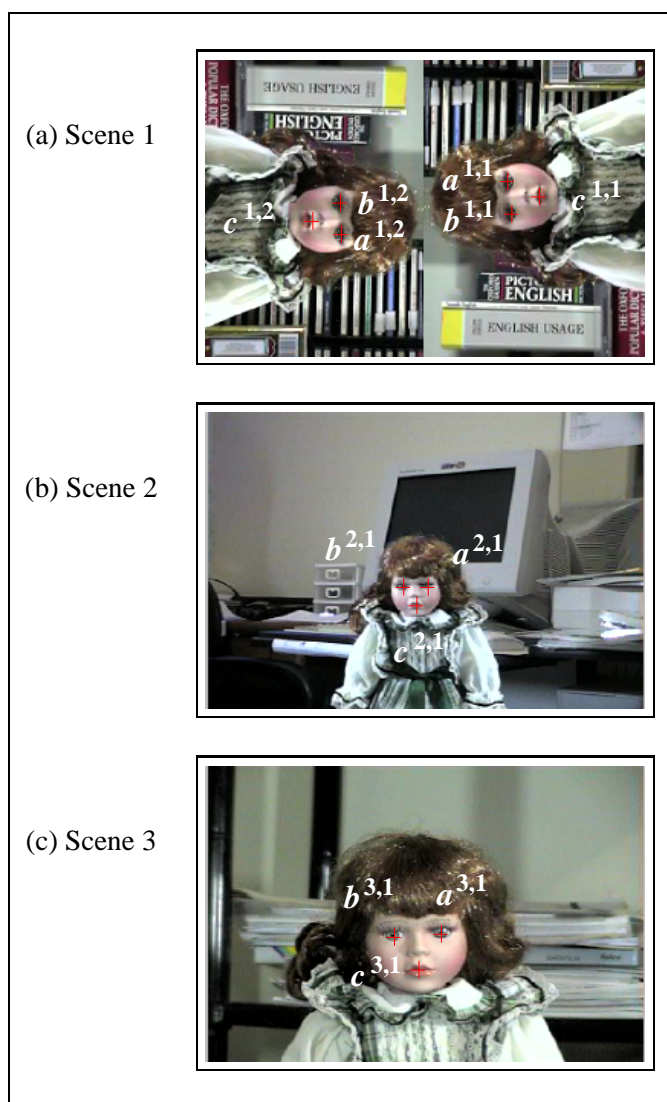


Figure 5.10: Scenes with the doll object for structure learning. In order to make the figure not too crowded, instead of drawing a circle where the retinal masks are, we just show the fixation points with small crosses. Features were automatically labelled after running Algorithm 5.1.

(a)	$a^{1,2}$ (173,223)	$a^{2,1}$ (260,177)	$a^{3,1}$ (245,184)
$a^{1,1}$ (402,209)	1.00, 180, 0.54 1.00, 180, 1.00	0.70, 270, 0.68 0.63, 270, 1.00	1.44, 270, 0.44 1.37, 270, 1.00
$a^{1,2}$ (173,223)		0.70, 90, 0.48 0.63, 90, 1.00	1.44, 90, 0.31 1.37, 90, 1.00
$a^{2,1}$ (260,177)			2.07, 0, 0.45 2.17, 0, 1.00

(b)	$b^{1,2}$ (176,179)	$b^{2,1}$ (290,178)	$b^{3,1}$ (310,189)
$b^{1,1}$ (398,254)	1.00, 180, 0.68 1.00, 180, 1.00	0.70, 270, 0.57 0.63, 270, 1.00	1.20, 270, 0.30 1.37, 270, 1.00
$b^{1,2}$ (176,179)		0.70, 90, 0.24 0.63, 90, 1.00	1.20, 90, 0.22 1.37, 90, 1.00
$b^{2,1}$ (290,178)			2.07, 0, 0.43 2.17, 0, 1.00

(c)	$c^{1,2}$ (139,199)	$c^{2,1}$ (275,155)	$c^{3,1}$ (280,140)
$c^{1,1}$ (434,235)	1.00, 180, 0.91 1.00, 180, 1.00	0.70, 270, 0.61 0.63, 270, 1.00	1.44, 270, 0.61 1.37, 270, 1.00
$c^{1,2}$ (139,199)		0.70, 90, 0.72 0.63, 90, 1.00	1.44, 90, 0.50 1.37, 90, 1.00
$c^{2,1}$ (275,155)			2.07, 0, 0.50 2.17, 0, 1.00

Table 5.28: Model database from the doll experiment. (a) Left eye features. (b) Right eye features. (c) Mouth features. Ground truth relationship values are presented, using a small font, underneath each computed triplet ( $rS, rO, Sm$ ).

(a)	$a^{1,2}$ (170,223)	$a^{2,1}$ (258,175)	$a^{3,1}$ (245,184)
$a^{1,1}$ (402,210)	1.16,199,0.36	0.71,243,0.66	1.58,261,0.63
$a^{1,2}$ (170,223)		0.71, 87,0.41	1.38, 99,0.19
$a^{2,1}$ (258,175)			2.43,350,0.62

(b)	$b^{1,2}$ (177,180)	$b^{2,1}$ (290,175)	$b^{3,1}$ (307,190)
$b^{1,1}$ (399,252)	0.97,153,0.72	0.84,295,0.51	1.15,289,0.16
$b^{1,2}$ (177,180)		0.69, 91,0.13	1.03,101,0.21
$b^{2,1}$ (290,175)			2.02, 12,0.41

(c)	$c^{1,2}$ (138,197)	$c^{2,1}$ (273,153)	$c^{3,1}$ (279,142)
$c^{1,1}$ (436,234)	0.89,154,0.62	0.72,241,0.35	1.30,267,0.33
$c^{1,2}$ (138,197)		0.58, 70,0.89	1.31, 77,0.75
$c^{2,1}$ (273,153)			1.92, 26,0.68

Table 5.29: Results of perturbing Table5.28 with noise parameters  $\mathcal{P}(3,2,30,0.5)$ .



#	Clique	Rank
1	$(c^{1,2}, c^{2,1}, c^{3,1}) (a^{1,2}, a^{2,1}, a^{3,1}) (b^{1,2}, b^{2,1}, b^{3,1})$	0.59
2	$(a^{1,1}, a^{2,1}, a^{3,1}) (c^{1,1}, c^{2,1}, c^{3,1}) (b^{1,1}, b^{2,1}, b^{3,1})$	0.57

Table 5.30: Maximal cliques correctly found for the doll experiment with no missing features.

#	Clique	Rank
1	$(c^{1,2}, c^{2,1}, *) (a^{1,2}, a^{2,1}, a^{3,1}) (b^{1,2}, b^{2,1}, *)$	0.63
2	$(a^{1,1}, a^{2,1}, a^{3,1}) (c^{1,1}, c^{2,1}, *) (b^{1,1}, b^{2,1}, *)$	0.55

Table 5.31: Maximal cliques correctly found for the doll experiment with missing features  $b^{2,1}$  and  $c^{3,1}$ .

## 5.5 Summary

In this chapter we provided an answer to the question of whether or not is possible to learn rigid geometric models from 2-D image evidence (iconic object models) acquired from a set of scenes. We found that structured models can indeed be learnt in such a context by using a graph-based representation and algorithms.

Iconic models and relations are obtained via a clustering algorithm that works by evaluating the similarities between new and existing object features represented in terms of a set of retina-like image planes. We assume that the locations of the fixation points of object features in a scene are provided by an external attention mechanism (not investigated in this thesis), which uses bottom-up (based on low-level features) and top-down (based on high level models) strategies to consistently indicate regions of interest in the objects. The image representation itself was presented in Chapter 3 and the raw primal-sketch features that form the image planes were discussed in Chapter 4.

The sensitivity of the similarity function to the misplacement of the fixation points was analysed in terms of four possible combinations of object and background type. Results showed that, regardless of image contents, errors above 3 pixels in both directions away from the ideal fixation point would cause the function to produce values below half of its maximum output. The only exception is when the points are on the symmetry axis of an object (if present), in which case a larger misplacement would be tolerated.

A number of experiments involving synthetic scenes helped not only to understand the mechanisms behind the graph construction but also to elicit the tolerance to noise and the true limitations of the approach. Some case studies also showed how the approach works with real images.

There are some issues related to the algorithms described in this chapter that require further research. For instance, there are clearly other ways of defining the rank of a vertex, edge or clique. The vertex creation process is not yet the optimal solution to the problem as it suffers from a scalability problem: the size of the resulting combinatorial set grows exponentially with the number of images. However it is still a reasonable solution for a few tens of images. One way to reduce the number of combinations would be to pre-group multiple instances of the same model class as if it were a new type of object. Finding a more computationally attractive vertex definition is also left as future work.

Although the structure learning approach presented in this chapter required a computational time exponential in the number of input scene images (see Figure C.10 in Appendix C for reference), it should be pointed out that this is meant to be an off-line or background learning process. Moreover, the goal is not to learn from a large set of images at once, but to incrementally build structured models from small sets of images, therefore minimising the graph construction complexity. Given that the clustering process keeps classifying image regions while recording their relative scale and orientations, it would not be difficult to combine the outputs of the structure learning approach when applied to small sets of scenes at a time. Although we left this as future work, we shall briefly discuss below how this might be accomplished.

Suppose we had sufficient time to run our structure learning method on a large set of scenes from which a particular structure model could be learnt. Since we are looking only at the image evidence, it is likely that the results of the structure learning method on subsets of this large set of scenes would yield approximately the same model. Obviously, models learnt from smaller subsets would be less reliable due to reduced supporting image evidence.

Thus, the process of integrating the outputs of multiple structure learning runs would roughly consist in: (a) merging the common structures learnt (cliques); then (b) updating the vertex, edge and clique ranks to reflect the improved image evidence, and (c) adding to the system

the possible new structures found. It should be pointed out that the task of determining the common cliques across two or more learning runs is simpler than a graph isomorphism problem (which is found to be NP hard), since here all the vertices in the graph will be consistently labelled.

A slightly simpler, yet similar, approach would involve performing an initial bootstrap learning of a few scenes (typically 5 or 6), then incrementally identifying and removing known image evidence (or object features) for every new scene analysed, thus allowing learning to take place only on new evidence. The process of identifying image evidence that is already explained by the existing models could be accomplished via a simple relational matching algorithm. Obviously, models should be incrementally updated with this evidence. Structure learning would therefore take place only on the features that are not yet part of any existing model.

## *Chapter 6*

# *Conclusion and Further Work*

Model-based vision is computationally intractable without reducing the large set of objects that potentially explain a set of data to a few serious candidates that require more detailed analysis.

*R. B. Fisher [Fis89]*

### **Contents**

---

<b>6.1 Summary of the Thesis and Contributions</b>	<b>157</b>
<b>6.2 Future Work</b>	<b>160</b>

---

This chapter summarises all the research done and contributions as well as gives a list of possible future research to be carried out on each of the main issues addressed in this thesis: primal sketch feature extraction, primitive iconic model learning and structure learning.

## 6.1 Summary of the Thesis and Contributions

We present a summary based on the three main chapters of the thesis:

### 6.1.1 Log-Polar Image Representation

Initially, in Chapter 3, we presented the specification of a biologically inspired image representation, which has interesting properties:

- The input Cartesian image is re-sampled through the use of a set of overlapping receptive fields which produces an image smaller in size but still retaining high resolution in the middle.
- The progressively lower resolution periphery is implemented as a log-polar image, which has the property of converting changes in scale and rotation into translations in the log-polar space.
- If higher resolutions are needed in the periphery, a simple attention mechanism could be used to change the foveation point to any point in the periphery.
- The receptive field computation together with the pre-processing stage uses an estimate for the local surface reflectance of objects in the scene which makes it approximately invariant to local changes in the illumination and scene composition.
- It suits several practical applications such as time-to-contact in active vision, general object recognition and image compression [ST92].

There is no particular novelty to the work presented in this chapter, being related mainly to research published by [RS90], [ST92], [GF96], [LWV97] and [Jur99]. The main purpose of this chapter was to develop an image representation that was suitable for the most relevant components of the research presented in this thesis, which are discussed in Chapters 4 and 5.

### 6.1.2 Primal Sketch Feature Extraction

In the main section of Chapter 4, we concentrated on a new approach to extracting primal sketch features from the log-polar image representation. It is believed that primal sketch

features like (*edges, bars, blobs and ends*) [Mar82] are used by the human visual system as more compact representations for image data and also as cues for an attention mechanism.

A previous approach to detecting these features in a log-polar image [GF96] used heuristically defined operators and did not produce good results when applied to real images. Another work [LWV97] modelled lines, circular arcs and ellipses using equations and a LMS fit was employed to minimise the data to model error. The main novelty of our approach was to learn the transformation that computed the features instead of trying to build an explicit model of them. Neural Network learning of edge features has been attempted before with uniformly sampled images (see, for instance, [PCB94] and [CTR95]), but to date no other work, apart from ours, has extracted, via learning, primal sketch features (not exclusively edges) in a log-polar sensor geometry. Below are the other important aspects of our approach:

- Features are detected at several different orientations and contrasts, which gives a rich description of the detected features.
- Contrasts are coded at the neural network outputs in a way that imitates the human ability to perceive them.

The implementation of our approach was not straightforward, but problems were tackled with some few refinements. The direct coding of orientations at the neuron outputs did not work appropriately. As a result a set of symmetry operations was defined to normalise the feature orientations prior to learning and classification. Also, using receptive fields values as the networks inputs did not provide enough separation at the input space, and thus a PCA pre-processing module was included to reduce the number of inputs and increase separability. Although only using synthetic features in the training sets proved to be satisfactory under a performance evaluation involving synthetic testing features, when applying the initially trained networks to real images the results were not the same. Thus a process of manually selecting a few features from real images was introduced in order to enrich the training sets.

From the results presented in Chapter 4 and Appendix A, we can conclude that our feature extraction approach is successful, but there is still room for some further improvement, which we discuss in Section 6.2.

Thus, this thesis has presented an original method for computing primal sketch features in a

log-polar image, and also a method for learning the parameters of that computation.

### 6.1.3 Model Learning

#### Iconic Models

In a simple algorithm we have shown how to build models of visual objects found in a set of scenes through an iconic vision system. An iconic model is defined as a set of regions, or object instances, that are similar to each other, and comprises a list of relative scales, orientations, positions and similarity scores for each pair of image regions. This is accomplished by using the rotation/scaling properties of the log-polar map and a cross-correlation classifier.

There are several works in the literature dealing with matching, tracking and searching objects in log-polar images (e.g. [Jur99], [SE95], [GF96], [Mac97], [FM98] or [LWV97]), however the model classes are previously known, and are either hardwired into the system or manually selected and segmented to form training sets.

An important aspect of the method we designed was to assume that object components (or features) would be likely to appear under a set of interest points given by an external attention mechanism. To a certain extent, the colour normalisation implemented in Chapter 2 and the low resolution retinal periphery helped to match features under different backgrounds and light conditions.

Therefore, no prior knowledge of the scene structure and objects is required, nor is human assistance, when determining the classes of models. Although computationally unattractive, this algorithm proved to be feasible when the number of images and features per image is not too large, which is acceptable since the next step of the approach (structure learning, discussed below) can operate properly on this density of data.

#### Structure Learning

In this thesis we also answered affirmatively the important question of whether or not it would be possible to learn rigid geometric models from 2-D image evidence (iconic object models) acquired from a set of scenes. In Chapter 5 we found that structured models can indeed be learnt in such a context by using a graph-based representation and algorithm. In a number

of experiments with synthetic and real scenes, we have shown how our approach works in practise.

An important difference between the way we learn models and the existing traditional approaches is that our system is designed to search the visual field for objects in an attentive way, like humans and some other animals do. In this way, the relative position of clustered features can be recorded and, with the help of the features' relative scale and orientation, possible relationships among features can be worked out.

Within the class of geometric, symbolic or structure based object recognition, typical systems normally use surface [Fis89] or volumetric [Bie86] relationships between object components to build models and improve matching. In our work, instead, we use 2-D image (iconic) evidence to build structured models.

In [FM98], subcomponent evidence from model relations was used to improve matching and attention. Thus, our approach to model learning extended that work in providing an autonomous way of building a database of iconic models and relations.

## 6.2 Future Work

### 6.2.1 Primal Sketch Feature Extraction

#### **Integrating all the Classifiers**

When looking at the results of the application of the final architecture on real images, it is easy to see that there is some intersection between the outputs of the classifiers. Although it is acceptable to have a pattern that is in the boundary between two different classes, and thus being classified as a member of both classes, for some problems it would be more appropriate to have the intersection between the outputs of all classifiers be an empty set. This intersection was minimised when we included a subset of features from other classes as counter-examples to a particular feature class. But if the objective is to have a completely disjunct set of features, something else has to be used. One option would be to include a final decision rule to choose the classifier that produced the strongest output and discard all the other classifiers for that particular feature. Another option would be to add to our architecture a final winner-takes-all neural network that would be used to choose a unique feature class. The training set for this



classifier could be built, for instance, from some human decisions about the feature identity.

### **Analysis of Receptive and Projective Fields of Hidden Units**

Once all neural networks were trained and tested, it can be useful to analyse the patterns of weights on connections to and from the hidden units, so that we can have an idea on how the mappings were actually performed. The goal is to determine which parts of the input were particularly important for a given unit by examining which connections contributed mostly to the activation of that unit. This is usually called an analysis of *receptive fields*<sup>1</sup>. There is also an analogous analysis that can be performed on the hidden units, which is called analysis of *projective fields*. The projective field of a hidden unit is composed of those output units that have strong weights on their connections from that hidden unit.

One obvious application of the above analysis is to provide an understanding on how the network solves the problem. For example, Kosslyn [Kos94] analysed the hidden units of a network designed to recognise shapes and give its location on a 5x5 input array. He discovered that some hidden units had strong positive weights from input units that were arranged into horizontal, vertical or diagonal bars in the input array and then concluded that those units were apparently serving as feature detectors.

Another interesting application is to produce an approximation for the outputs of a trained neural network in terms of a set of logical rules or a mathematically defined operator. Under certain conditions, this can be done automatically. Thus, our solution could be dramatically simplified by the use of those logical rules, which may be more transparent than the trained modules. This approach has a clear advantage when compared to a previous approach, which also used some sort of logical rules, in the sense that now the rules would be based on real statistics of the data and not on heuristics.

## **6.2.2 Model Learning**

### **Primitive Iconic Models**

It is clear that a more appropriate (and efficient) classifier can be used here. Instead of actually storing the object instances themselves while building iconic models, a unsupervised Neural

---

<sup>1</sup> Note this is a slightly different use of the term *receptive field* from that used in image feature detection.

Network or a Kernel-Based method could probably do a better job by representing object classes by means of a small number of parameters. Moreover, it would be interesting to investigate how to avoid having to perform sequential access to all object instances in the model base before a match occurs.

A naive, but perhaps effective, algorithm could match unseen object features with just one model instance (a prototype) per model class in the database. This would produce only one set of matching results comprising similarity, and geometric relations of scale and orientation. The unknown results between a unseen object feature and all the remaining instances of the winner class, which are required by the structure learning approach, could be estimated by indirect calculations (or propagation in the database) using the existing computed values. However, the choice of a prototype model instance may be a difficult task since we do not have all real similarity measurements to compare in a first place.

### **Structure Learning**

There are some issues related to the algorithms described in this thesis that require further research. For instance, there are other ways of defining the rank of a vertex, as for example the average of the similarity scores between all the pairs of vertex elements. A study on how the functions used to rank vertices, edges and cliques influence the learning results would be equally important.

The vertex creation process is not yet the optimal solution to the problem as it suffer from a scalability problem: the size of the resulting combinatorial set grows exponentially with the number of images. However it is still a reasonable solution for a few tens of images. One way to reduce the number of combinations would be to pre-group multiple instances of same model class as if it were a new type of object. Finding a more computationally attractive vertex definition is left as future work.

# *Bibliography*

- [BA83] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983. Referred to in page(s) 15, 34.
- [BAR94] J. Ben-Arie and K.R. Rao. Optimal template matching by nonorthogonal image expansion using restoration. *Machine Vision and Applications*, 7:69–81, 1994. Referred to in page(s) 3, 9.
- [BAW97] J. Ben-Arie and Z. Wang. Pictorial recognition using affine-invariant spectral signatures. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 35–39, San Jose, Puerto Rico, 1997. Referred to in page(s) 12.
- [BFG96] J.R. Bach, C. Fuller, and A. Gupta. The virage image search engine: An open framework for image management. In *SPIE Electronic Imaging: Storage and Retrieval for Image and Video Databases IV*, volume 2670, pages 76–87, San Jose, California, February 1996. SPIE. Referred to in page(s) 3, 13.
- [BGG96] V. Bruce, P. Green, and M. Georgeson. *Visual Perception: Physiology, Psychology, and Ecology*. Psychology Press, 3rd edition, 1996. Referred to in page(s) 77.
- [Bie86] I. Biederman. Human image understanding: recent research and a theory. In A. Rosenfeld, editor, *Human and Machine Vision II*, Perspectives in Computing, pages 13–57. Academic Press, 1986. Referred to in page(s) 2, 160.
- [BS99] G. Bonmassar and E.L. Schwartz. Space-variant fourier analysis: the exponential chirp transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1080–1089, October 1999. Referred to in page(s) 11.
- [BSV96] A. Bernadino and J. Santos-Victor. Sensor geometry for dynamic vergence: Characterization and performance analysis. In *Proceedings of Workshop on Performance Characteristics of Vision Algorithms*, Cambridge, UK, April 1996. Referred to in page(s) 62, 62.
- [BWBD86] J. Babaud, A.P. Witkin, M. Baudin, and R.O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:26–33, 1986. Referred to in page(s) 15.
- [Chr75] N. Christofides. *Graph Theory: An Algorithmic Approach*. Academic Press, 1975. Referred to in page(s) 116.

- [CTR95] W.C. Chen, N.A. Thacker, and P.I. Rockett. A neural network for probabilistic edge labelling trained with a step edge model. In *Proceedings of 5th International Conference on Image Processing and its Applications*, pages 618–621, Edinburgh, Scotland, July 1995. Referred to in page(s) 23, 34, 158.
- [Dau88] J.G. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988. Referred to in page(s) 95, 95.
- [DH73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc., 1973. Referred to in page(s) 103.
- [DK95] K. Daniilidis and V. Kruger. Optical flow computation in the log-polar-plane. In *Proceedings of International Conference on Computer Analysis of Images and Patterns*, pages 65–72, 1995. Referred to in page(s) 62.
- [dSKC<sup>+</sup>89] J. Van der Spiegel, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Belluti, and G. Soncini. A foveated retina-like sensor using ccd technology. In C. Mead and M. Ismail, editors, *Analog VLSI and Neural Network Implementations*. Kluwer, Boston, 1989. Referred to in page(s) 11, 57, 63.
- [EC93] C. Enroth-Cugell. Helmerich lecture: The world of retinal ganglion cells. In R. Shapley and D. Man-Kit Lam, editors, *Contrast Sensitivity*, volume 5, chapter 9, pages 149–179. MIT Press, 1993. Referred to in page(s) 55.
- [EM97] J. Edwards and H. Murase. Appearance matching of occluded objects using coarse-to-fine adaptive masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 533–539, San Jose, Puerto Rico, 1997. Referred to in page(s) 9, 10.
- [Eri90] C. H. Eriksen. Attentional search of the visual field. In D. Brogan, editor, *Visual Search*, chapter I.1, pages 3–19. Taylor and Francis, 1990. Referred to in page(s) 5.
- [FCS98] B. Fischl, M.A. Cohen, and E.L. Schwartz. Rapid anisotropic diffusion using space-variant vision. *International Journal of Computer Vision*, 28(3):199–212, July 1998. Referred to in page(s) 59.
- [Fel85] J.A. Feldman. Four frames suffice: a provisional model of vision and space. *Behavioural and Brain Sciences*, 8(2):265–313, 1985. Referred to in page(s) 6, 8, 25, 27, 27, 28.
- [FF95a] C. Fuchs and W. Förstner. Polymorphic grouping for image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 175–182, MIT, Massachusetts, 1995. Referred to in page(s) 3.
- [FF95b] B.V. Funt and G.D. Finalyson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, 1995. Referred to in page(s) 16.
- [FG96] R.B. Fisher and A.P. Gionis. Surface reflectance recovery under point light illumination. In R. B. Fisher and E. Trucco, editors, *Proceedings of British Machine Vision Conference*, pages 263–272, 1996. Referred to in page(s) 16.

- [Fis86] R.B. Fisher. *From Surfaces to Objects: Recognizing Objects Using Surface Information and Object Models*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1986. Referred to in page(s) 17, 19, 32.
- [Fis89] R.B. Fisher. *From Surfaces to Objects*. John Wiley and Sons, 1989. Referred to in page(s) 2, 17, 19, 32, 156, 160.
- [Fle90] M.M. Fleck. Classifying symmetry sets. In *Proceedings of British Machine Vision Conference*, pages 297–302, Oxford, England, 1990. Referred to in page(s) 68.
- [FM98] R.B. Fisher and A. MacKirdy. Integrating iconic and structured matching. In *Proceedings of the European Conference on Computer Vision*, volume II, pages 687–698, Freiburg, Germany, June 1998. Referred to in page(s) 3, 3, 6, 9, 32, 100, 159, 160.
- [FO95] R.B. Fisher and P. Oliver. Multi-variate cross-correlation and image matching. In *Proceedings of British Machine Vision Conference*, pages 623–632, 1995. Referred to in page(s) 102.
- [FSC97] B. Fischl, E.L. Schwartz, and M.A. Cohen. The local structure of space-variant images. *Neural Networks*, 10(5):815–831, 1997. Referred to in page(s) 12, 34, 59.
- [GF96] T.D. Grove and R.B. Fisher. Attention in iconic object matching. In *Proceedings of British Machine Vision Conference*, volume 1, pages 293–302, Edinburgh, Scotland, 1996. Referred to in page(s) 3, 3, 6, 9, 25, 28, 29, 30, 32, 37, 37, 55, 59, 60, 62, 65, 66, 86, 91, 92, 94, 98, 99, 100, 157, 158, 159, 172, 172.
- [GF00] H.M. Gomes and R.B. Fisher. Structural learning from iconic representations. *Lecture Notes in Artificial Intelligence, subseries of Lecture Notes in Computer Science*, 1952:399–408, 2000. Also *Proceedings of International Joint Conference: 7th Ibero-American Conference on AI and 15th Brazilian Symposium on AI*. Referred to in page(s) v, 99.
- [GF01] H.M. Gomes and R.B. Fisher. Learning and extracting primal-sketch features in a log-polar image representation. In *Proceedings of Brazilian Symposium on Computer Graphics and Image Processing*, pages 338–345, Florianópolis, Brazil, October 2001. IEEE Computer Society. Referred to in page(s) v, 5, 98.
- [GF02] H.M. Gomes and R.B. Fisher. An architecture for primal-sketch feature extraction in a log-polar representation. Submitted to *Pattern Recognition Letters*, 2002. Referred to in page(s) 5, 98.
- [GFH98] H.M. Gomes, R.B. Fisher, and J. Hallam. A retina-like image representation of primal sketch features extracted using a neural network approach. In *Proceedings of Noblesse Workshop on Non-Linear Model Based Image Analysis*, pages 251–256, Glasgow, Scotland, July 1998. Springer-Verlag London. Referred to in page(s) v, 5, 76, 98.
- [GJ97] A. Gupta and R. Jain. Visual information retrieval. *Communications of the ACM*, 40(5):70–79, May 1997. Referred to in page(s) 3, 13.

- [Gri90] W.E.L. Grimson. The effect of indexing on the complexity of object recognition. Technical Report 1226, MIT, Artificial Intelligence Lab, April 1990. AI Memo. Referred to in page(s) 17, 19.
- [GW92] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1992. Referred to in page(s) 34, 101.
- [HB89] B.K.P. Horn and M.J. Brooks, editors. *Shape from shading*. MIT Press, Cambridge, Mass. London, 1989. Referred to in page(s) 56.
- [HCK97] C.Y. Huang, O.I. Camps, and T. Kanungo. Object recognition using appearance-based parts and relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 877–883, Puerto Rico, June 1997. IEEE Computer Society Press. Referred to in page(s) 9.
- [Hur92] A.C. Hurlbert. Neural network approaches to color vision. In H. Wechsler, editor, *Neural Networks for Perception*, volume 1(Human and Machine Perception), chapter II.5, pages 265–284. Academic Press, 1992. Referred to in page(s) 55.
- [HVC00] D. Hall, V.C. Verdière, and J.L. Crowley. Object recognition using coloured receptive fields. In *Proceedings of the European Conference on Computer Vision*, pages 164–177, Trinity College, Dublin, Ireland, 2000. Referred to in page(s) 9, 17.
- [HW62] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962. Referred to in page(s) 95.
- [Jac91] J.E. Jackson. *A User's Guide to Principal Components*. Wiley series in probability and mathematical statistics. John Wiley & Sons Inc., 1991. Referred to in page(s) 71, 71.
- [Jen94] J. Jennens. Quasi-invariant iconic object recognition. Unpublished MSc dissertation, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1994. Referred to in page(s) 3, 6, 9, 28.
- [Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer series in statistics. Springer-Verlag Inc., New York, 1986. Referred to in page(s) 71, 71, 74.
- [Jur99] F. Jurie. A new log-polar mapping for space variant imaging. application to face detection and tracking. *Pattern Recognition*, 32:865–875, 1999. Referred to in page(s) 10, 62, 157, 159.
- [Kos94] S.M. Kosslyn. *Image and brain: the resolution of the imagery debate*. MIT Press, London, England, 1994. Referred to in page(s) 161.
- [KvD92] J.J. Koenderink and A.J. van Doorn. Generic neighborhood operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):597–605, 1992. Referred to in page(s) 15.
- [Lee96] T.S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10), October 1996. Referred to in page(s) 96.

- [Lei92] R. R. Leighton. The Aspirin/MIGRAINES Neural Network Software - Release V6.0. Technical Report MP-91W00050, MITRE Corporation, October 1992. User's Manual. Referred to in page(s) 83.
- [Lin94] T. Lindberg. *Scale-space theory in computer vision*. Kluwer Academic, London, 1994. Referred to in page(s) 31.
- [Lin96] T. Lindeberg. Scale-space: A framework for handling image structures at multiple scales. In *Proceedings of CERN School of Computing*, pages 1–12, The Netherlands, September 1996. Referred to in page(s) 15.
- [LWV97] F.L. Lim, G.A.W. West, and S. Venkatesh. Use of log polar space for foveation and feature recognition. *IEE Proceedings: Vision, Image and Signal Processing*, 144(6):323–331, December 1997. Referred to in page(s) 11, 59, 62, 157, 158, 159.
- [Mac97] A. MacKirdy. Full subcomponent evidence and further parallelism in an iconic object recognition system. Unpublished Undergraduate Dissertation, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1997. Referred to in page(s) 3, 3, 6, 9, 32, 33, 99, 159.
- [Mar82] D. Marr. *Vision*. W. H. Freeman and Co., 1982. Referred to in page(s) 5, 6, 8, 8, 25, 25, 26, 28, 31, 34, 64, 64, 65, 95, 158.
- [Mar96] P.J.P. Marques. A parallel connectionist viewpoint quasi-invariant iconic object recognition system. Unpublished final report no. EPCC-SS96-13, Summer Scholarship Programme, Edinburgh Parallel Centre and Department of Artificial Intelligence, Edinburgh University, 1996. Referred to in page(s) 9, 32, 99.
- [Mes95] B.T. Messmer. *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*. PhD thesis, Institute for Applied Mathematics, University of Bern, CH, November 1995. Referred to in page(s) 116.
- [MH80] D. Marr and E. Hildreth. Theory of edge detection. In *R. Soc. Lond. B* 207, pages 187–217, 1980. Referred to in page(s) 95.
- [MK97] K. Messer and J. Kittler. Using feature selection to aid an iconic search through an image database. In *International Conferene on Acoustics, Speech and Signal Processing*, volume 4, Munich, Germany, April 1997. Referred to in page(s) 9.
- [MKK97] K. Messer, J. Kittler, and M. Kraaijveld. Selecting features for neural networks to aid an iconic search through an image database. In *IEE 6th International Conference on Image Processing and Its Applications*, pages 428–432, 1997. Referred to in page(s) 9, 13.
- [MLP<sup>+</sup>96] J. Mundy, A. Liu, N. Pillow, A. Zisserman, S. Abdallah, S. Utcke, S. Nayar, and C. Rothwell. An experimental comparison of appearance and geometric model based recognition. In *International Workshop on Object Representations in Computer Vision*, 1996. In association with ECCV 1996. Referred to in page(s) 9, 21, 22.

- [MN95] H. Murase and S.K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995. Referred to in page(s) 9.
- [MPW98] W. Myrvold, T. Prsa, and N. Walker. A dynamic programming approach for timing and designing clique algorithms. In *Algorithms and Experiments (ALEX'98): Building Bridges Between Theory and Applications*, pages 88–95, 1998. Referred to in page(s) 116.
- [MS99] J. Mundy and T. Saxena. Towards the integration of geometric and appearance-based object recognition. In D. A. Forsyth, J. L. Mundy, V. Gesu, and R. Cipolla, editors, *Lecture Notes In Computer Science 1681: Shape Contour and Grouping in Computer Vision*, pages 234–245. Springer-Verlag, Eidelberg, 1999. Referred to in page(s) 9.
- [MSP96] R. Milanese, D. Squire, and T. Pun. Correspondence analysis and hierarchical indexing for content-based image retrieval. In *Proceedings of the IEEE International Conference on Image Processing*, Lausanne, Switzerland, September 1996. Referred to in page(s) 13.
- [Nel96] R.C. Nelson. Memorization learning for object recognition. In S. K. Nayar and T. Poggio, editors, *Early Visual Learning*, chapter 9, pages 215–236. Oxford University Press, 1996. Referred to in page(s) 19.
- [NNM94] S.A. Nene, S.K. Nayar, and H. Murase. Slam: Software library for appearance matching. In *Proc. of ARPA Image Understanding Workshop*, Monterey, November 1994. Referred to in page(s) 21.
- [ONM00] N. Oshiro, A. Nishikawa, and F. Miyazaki. Robust vision for vision-based control of motion. In M. Vincze and G.D. Hager, editors, *Using foveated vision for robust object tracking: 3-D horopter analysis*, chapter 11, pages 137–143. IEEE Press, 2000. Referred to in page(s) 62, 62.
- [Pae87] B. Paechter. A new look at model invocation with special regard to supertype hierarchies. Unpublished MSc Dissertation, Dept. of Artificial Intelligence, University of Edinburgh, 1987. Referred to in page(s) 19.
- [Par94] F. Pardo. Development of a retinal image sensor based on cmos technology. Technical Report LIRA-TR 6/94, LIRA-Lab, DIST - University of Genova, 1994. Referred to in page(s) 11, 57, 63.
- [PB96] T. Poggio and D. Beymer. Regularization networks for visual learning. In S. K. Nayar and T. Poggio, editors, *Early Visual Learning*, chapter 3, pages 43–66. Oxford University Press, 1996. Referred to in page(s) 97.
- [PBC92] D.T. Pham and E. Bayro-Corrochano. Neural networks for low-level image processing. In *Proceedings of the IEE International Conference on Artificial Neural Networks*, pages 809–812, 1992. Referred to in page(s) 23.
- [PCB94] A.R. Pearce, T. Caelli, and W. Bischof. Rulegraphs for graph matching in pattern recognition. *Pattern Recognition*, 27(9):1231–1247, 1994. Referred to in page(s) 158.



- [PCS95] F. Panerai, C. Capurro, and G. Sandini. Space variant vision for an active camera mount. In *Proceedings of SPIE AeroSense95*, Florida USA, April 1995. Referred to in page(s) 62, 62.
- [PPS96] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook - content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, 1996. Referred to in page(s) 3, 11, 13.
- [PR92] R. Prokop and A. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *Graphical Modules and Image Processing*, 54(5):438–460, September 1992. Referred to in page(s) 68.
- [Rao97] R.P.N. Rao. Dynamic appearance-based recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 540–546, San Jose, Puerto Rico, 1997. Referred to in page(s) 9, 15.
- [RB95] R.P.N. Rao and D.H. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence Journal*, 78:461–505, 1995. Referred to in page(s) 3, 3, 9, 14, 16, 34.
- [Rot95] C. Rothwell. *Object recognition through invariant indexing*. Oxford University Press, 1995. Referred to in page(s) 19, 21, 21.
- [RS90] A.S. Rojer and E.L. Schwartz. Design considerations for a space-variant visual sensor with complex-logarithmic geometry. In *Proceedings of the International Conference on Pattern Recognition*, pages 278–285, 1990. Referred to in page(s) 11, 12, 36, 37, 59, 60, 157.
- [RZFM91] C.A. Rothwell, A. Zisserman, D.A. Forsyth, and J.L. Mundy. Using projective invariants for constant time library indexing in model based vision. In *Proceedings of British Machine Vision Conference*, pages 62–70, 1991. Referred to in page(s) 17, 19.
- [RZMF92] C.A. Rothwell, A. Zisserman, J.L. Mundy, and D.A. Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–114, 1992. Referred to in page(s) 19, 20, 20.
- [SB91] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991. Referred to in page(s) 3, 3, 11, 16, 16, 16.
- [SC96a] B. Schiele and J.L. Crowley. Object recognition using multidimensional receptive field histogram. In *Proceedings of the European Conference on Computer Vision*, pages 610–619, Cambridge UK, April 1996. Referred to in page(s) 3, 16, 16, 34.
- [SC96b] B. Schiele and J.L. Crowley. Probabilistic object recognition using multidimensional receptive field histogram. In *Proceedings of the International Conference on Pattern Recognition*, volume B, pages 50–54, Vienna, August 1996. Referred to in page(s) 3, 16, 16, 34.
- [Sch77] E.L. Schwartz. Spatial mapping in primate sensory projection: analytic structure and relevance to perception. *Biological Cybernetics*, 25:181–194, 1977. Referred to in page(s) 3, 10, 54, 59.

- [SE95] J.P. Siebert and I. Eising. Scale-space recognition based on the retino-cortical transform. In *Proc. IEE Conference on Image Processing and its Applications*, 1995. Edinburgh. Referred to in page(s) 9, 11, 159.
- [SKP93] R. Shapley, E. Kaplan, and K. Purpura. Contrast sensitivity and light adaptation in photoreceptors or in the retinal network. In R. Shapley and D. Man-Kit Lam, editors, *Contrast Sensitivity*, volume 5, chapter 7, pages 103–116. MIT Press, 1993. Referred to in page(s) 55.
- [SL97] G. Sela and M.D. Levine. Real-time attention for robotic vision. *Real-Time Imaging*, 3:173–194, 1997. Referred to in page(s) 59.
- [ST92] G. Sandini and M. Tristarelli. Vision and space-variant sensing. In H. Wechsler, editor, *Neural Networks for Perception*, volume 1(Human and Machine Perception), chapter II.11, pages 398–425. Academic Press, 1992. Referred to in page(s) 5, 10, 37, 59, 60, 62, 62, 157, 157.
- [STZ89] G.L. Scott, S.C. Turner, and A. Zisserman. Using a mixed wave/diffusion process to elicit the symmetry set. *Image and Vision Computing*, 7(1):63–70, 1989. Referred to in page(s) 68.
- [Sun01] Y. Sun. Modelling visual attention based on integrated competition and bidirectional dynamic interaction. PhD Thesis Proposal, 2001. Referred to in page(s) 99.
- [TB95] L.N. Thibos and A. Bradley. Modeling off-axis vision ii: the effect of spatial filtering and sampling by retinal neurons. In E. Peli, editor, *Vision Models for Target Detection and Resolution*, pages 338–379. World Scientific Press, Singapore, 1995. Referred to in page(s) 62.
- [Tho96] T. Thorhallsson. Detecting bilateral symmetry of 3d point sets from affine views. In *Proceedings of British Machine Vision Conference*, pages 73–82, Edinburgh, Scotland, 1996. Referred to in page(s) 68.
- [Tro93] J.B. Troy. Modeling the receptive fields of mammalian retinal ganglion cells. In R. Shapley and D. Man-Kit Lam, editors, *Contrast Sensitivity*, volume 5, chapter 6, pages 85–102. MIT Press, 1993. Referred to in page(s) 55.
- [Ull96] S. Ullman. *High-level vision : object recognition and visual cognition*. MIT Press, Cambridge, Mass. London, 1996. Referred to in page(s) 17.
- [VC99] V.C. Vèrdiere and J.L. Crowley. A prediction-verification strategy for object recognition using local appearance. Technical report, Prima project, GRAVIR Laboratory, 1999. Available at <ftp://ftp.inrialpes.fr/pub/prima/rapports/ColindeVerdiereCrowley-TechReport99.ps.gz>. Referred to in page(s) 9, 17.
- [Vos92] G. Vosselman. *Relational Matching*. Number 628 in Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, 1992. Referred to in page(s) 2.
- [Wan95] B.A. Wandell. *Foundations of Vision*. Sinauer Associates Inc., 1995. Referred to in page(s) 43.

- [Wat88] R.J. Watt. *Visual processing: computational, psychophysical and cognitive research*. Lawrence Erlbaum, 1988. Referred to in page(s) 26.
- [Wat91] R.J. Watt. *Understanding Vision*. Academic Press, 1991. Referred to in page(s) 1.
- [Wil83] S.W. Wilson. On the retino-cortical mapping. *International Journal on Man-Machine Studies*, 18:361–389, 1983. Referred to in page(s) 3, 10, 54, 59.
- [Win77] P.H. Winston. *Artificial Intelligence*. Addison-Wesley, 1977. Referred to in page(s) 18, 23, 24, 25.
- [Woo97] D. R. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21:211–217, 1997. Referred to in page(s) 116.
- [Yar67] A.L. Yarbus. *Eye Movements and Vision*. Plenum Press, 1967. Referred to in page(s) 65.
- [YRW92] Y. Yeshurun, D. Reifeld, and H. Wolfson. Symmetry: a context free cue for foveated vision. In H. Wechsler, editor, *Neural Networks for Perception*, volume 1(Human and Machine Perception), chapter II.15, pages 477–491. Academic Press, 1992. Referred to in page(s) 32.
- [Zek93] S. Zeki. *A Vision of the Brain*. Blackwell Scientific Publications, 1993. Referred to in page(s) 3.
- [ZFM<sup>+</sup>95] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, J. Liu, and N. Pillow. 3d object recognition using invariance. *Artificial Intelligence Journal*, 78:239–288, 1995. Referred to in page(s) 21.

## *Appendix A*

# *Primal Sketch Feature Extraction Examples*

This appendix presents some results of the system when applied to synthetic and real images. In order to facilitate the visualisation of the extracted features, log-polar images were converted back onto the Cartesian space. Darker pixels represent a higher estimated contrast, and lighter pixels represents the contrary.

Figures [A.1](#) to [A.5](#) exemplifies the use of the system (learning approach) on some simple synthetic features. The figures also show, for comparison purposes, the features extracted using the previous approach (logical operators) proposed in [[GF96](#)].

Figure [A.6](#) uses a real image to compare features extracted using our approach and the previous one proposed in [[GF96](#)].

Figure [A.7](#) shows the level of improvement obtained after the addition of real features into the training sets. Image 1 was used as a source of additional training exemplars (see table [4.6](#)) which were manually selected from the results of an initial application of the trained networks over this image.

Figures [A.8](#) and [A.9](#) show the results on entirely new images (images 2 to 4, from which no real features have been selected).




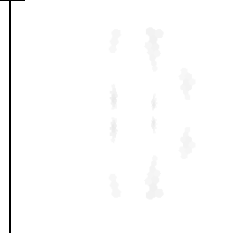
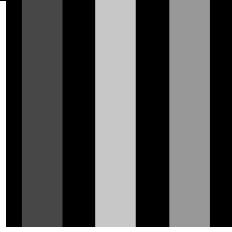

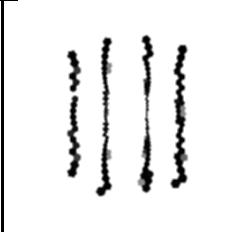
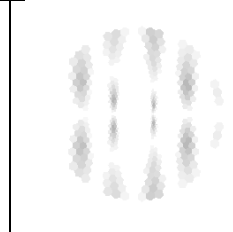





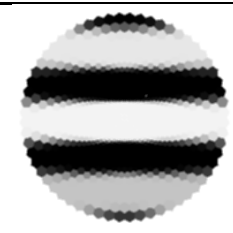
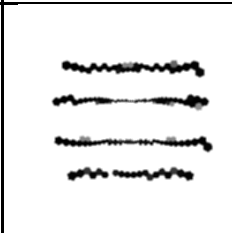
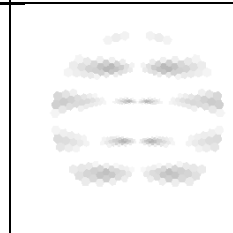

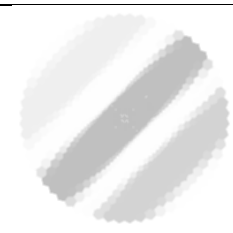

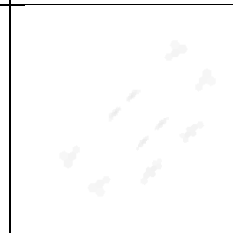

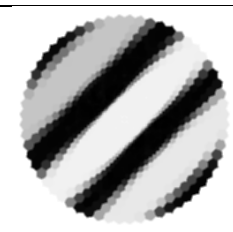
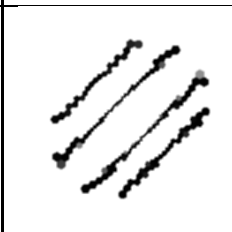
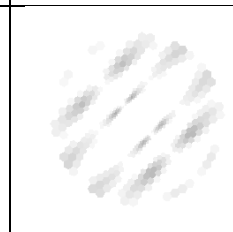
Input image	Retinal image	Learning approach	Logical operators
			
			
			
			
			
			

Figure A.1: *Edge classifiers on synthetic images.*

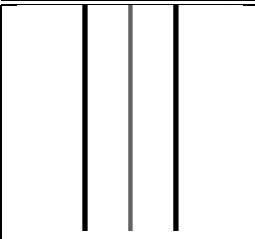

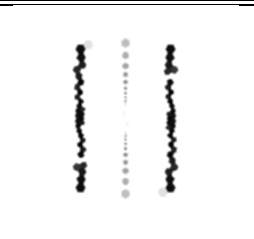
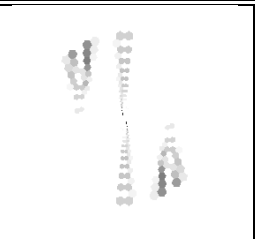
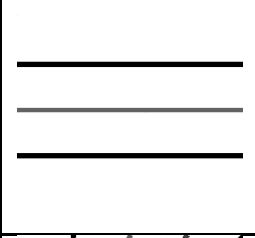


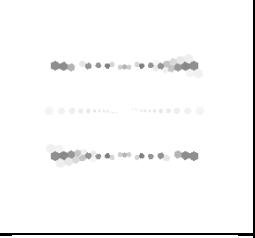
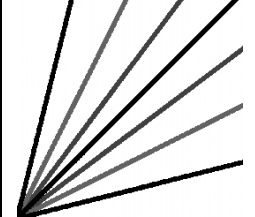
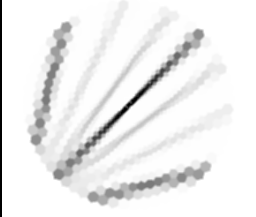

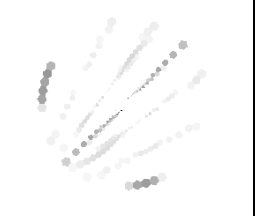
Input image	Retinal image	Learning approach	Logical operators
			
			
			

Figure A.2: *-Bar* classifiers on synthetic images.

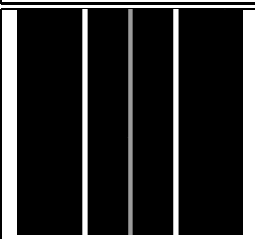
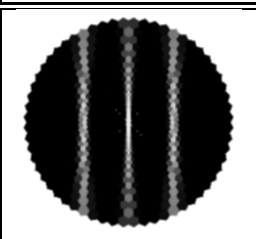

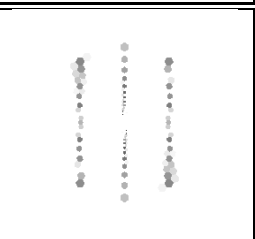
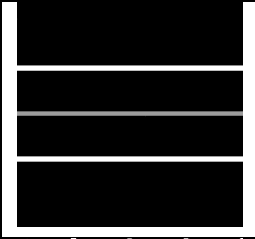


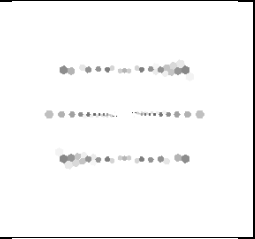
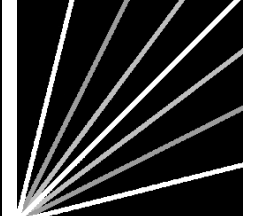


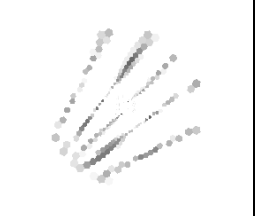
Input image	Retinal image	Learning approach	Logical operators
			
			
			

Figure A.3: *+Bar* classifiers on synthetic images.

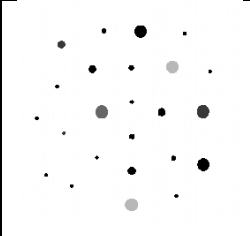
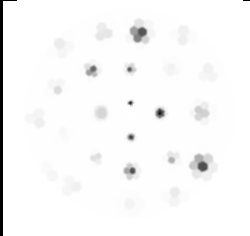


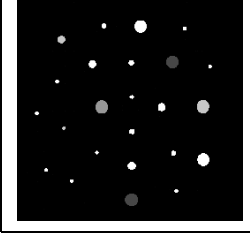
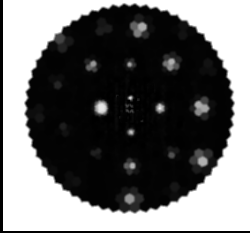

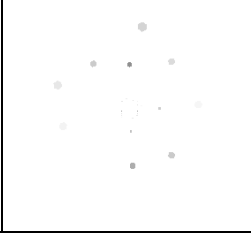
Input image	Retinal image	Learning approach	Logical operators
			
			

Figure A.4:  $\pm$ Blob classifiers, respectively, on synthetic images.

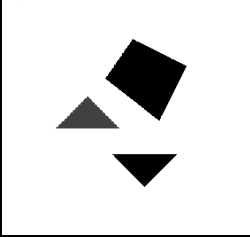
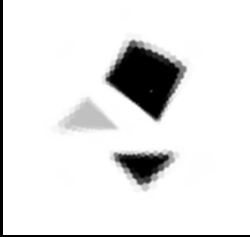


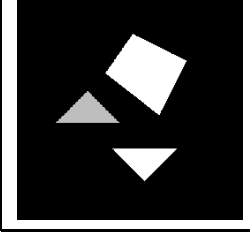


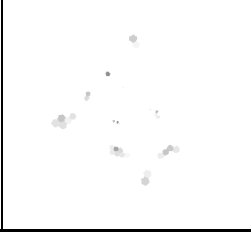
Input image	Retinal image	Learning approach	Logical operators
			
			

Figure A.5:  $\pm$ End classifiers, respectively, on synthetic images.

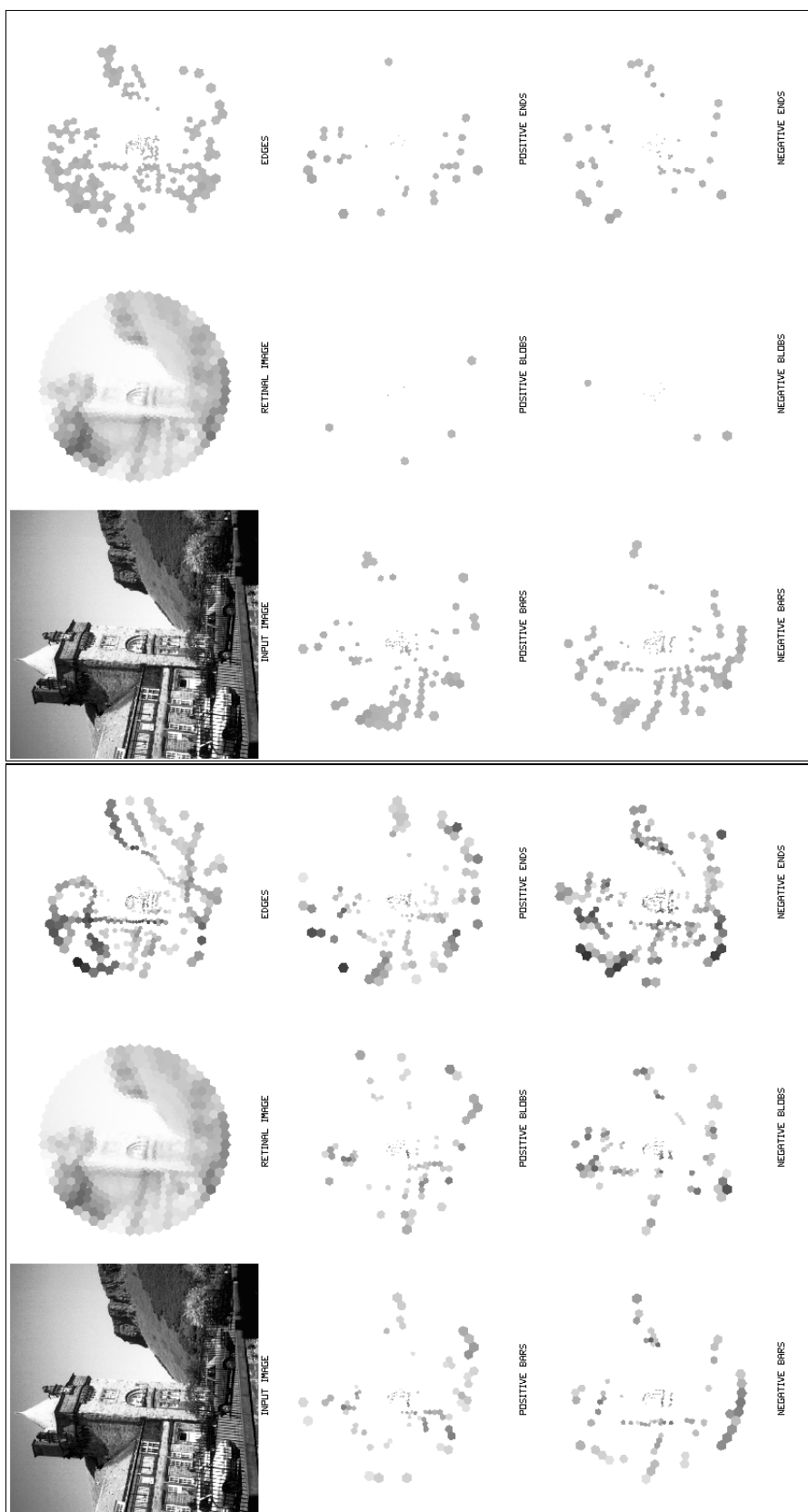


Figure A.6: Test image 1: from left to right, final results from our approach and from the previous approach.



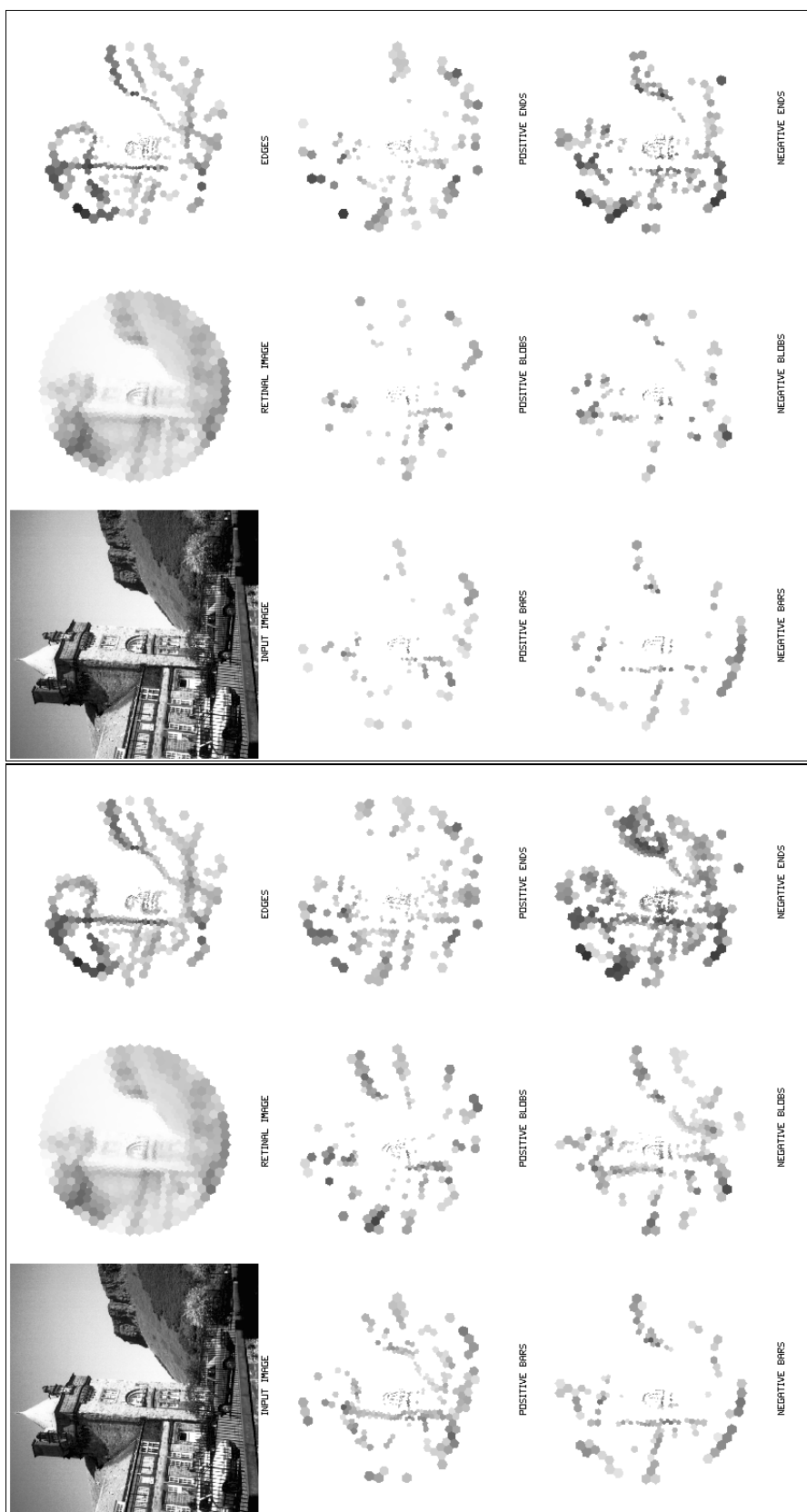


Figure A.7: Test image 1: from left to right, initial results from the architecture trained using synthetic features and after the addition of real features.

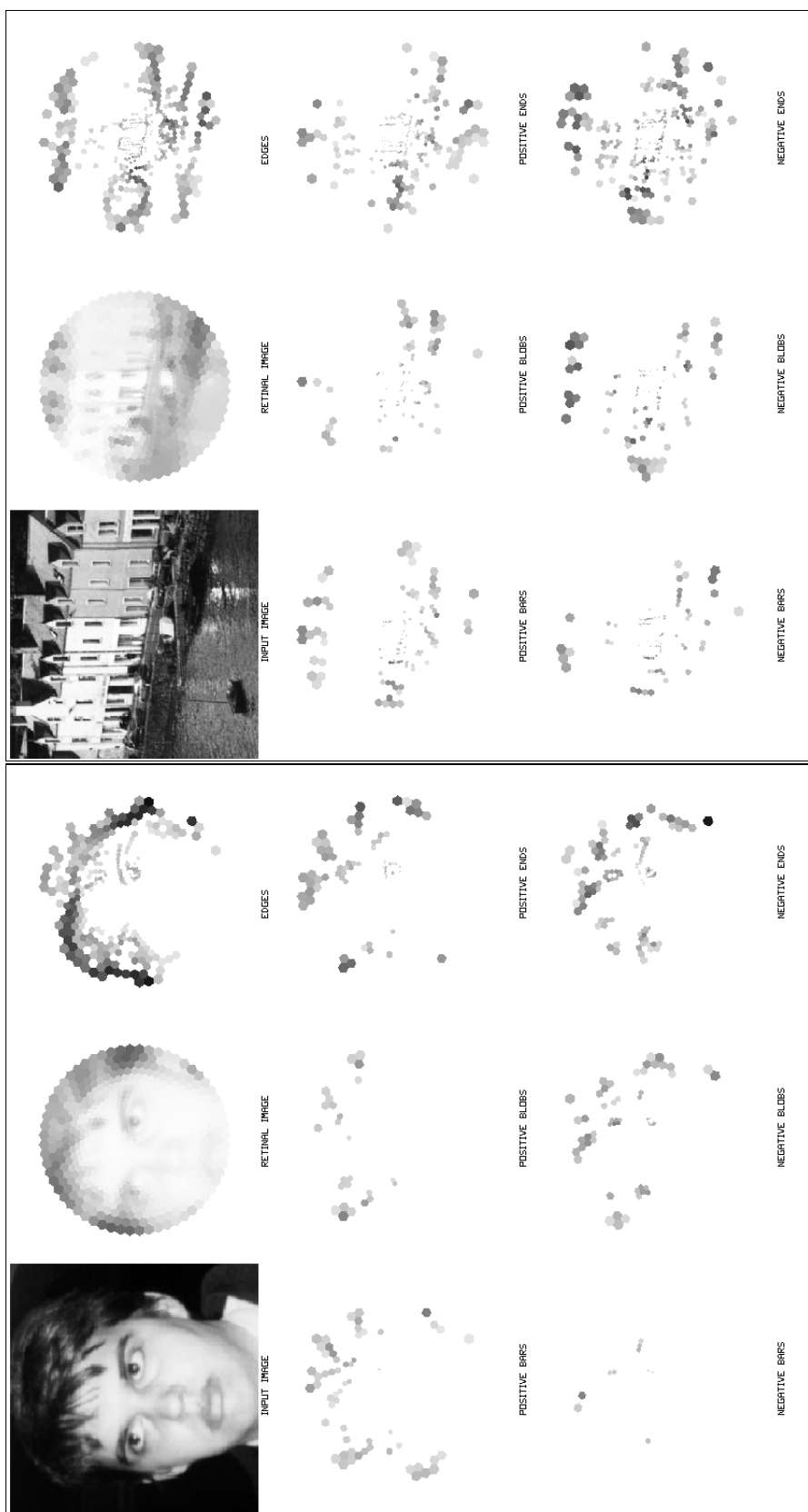


Figure A.8: Final results for test images 2 and 3.

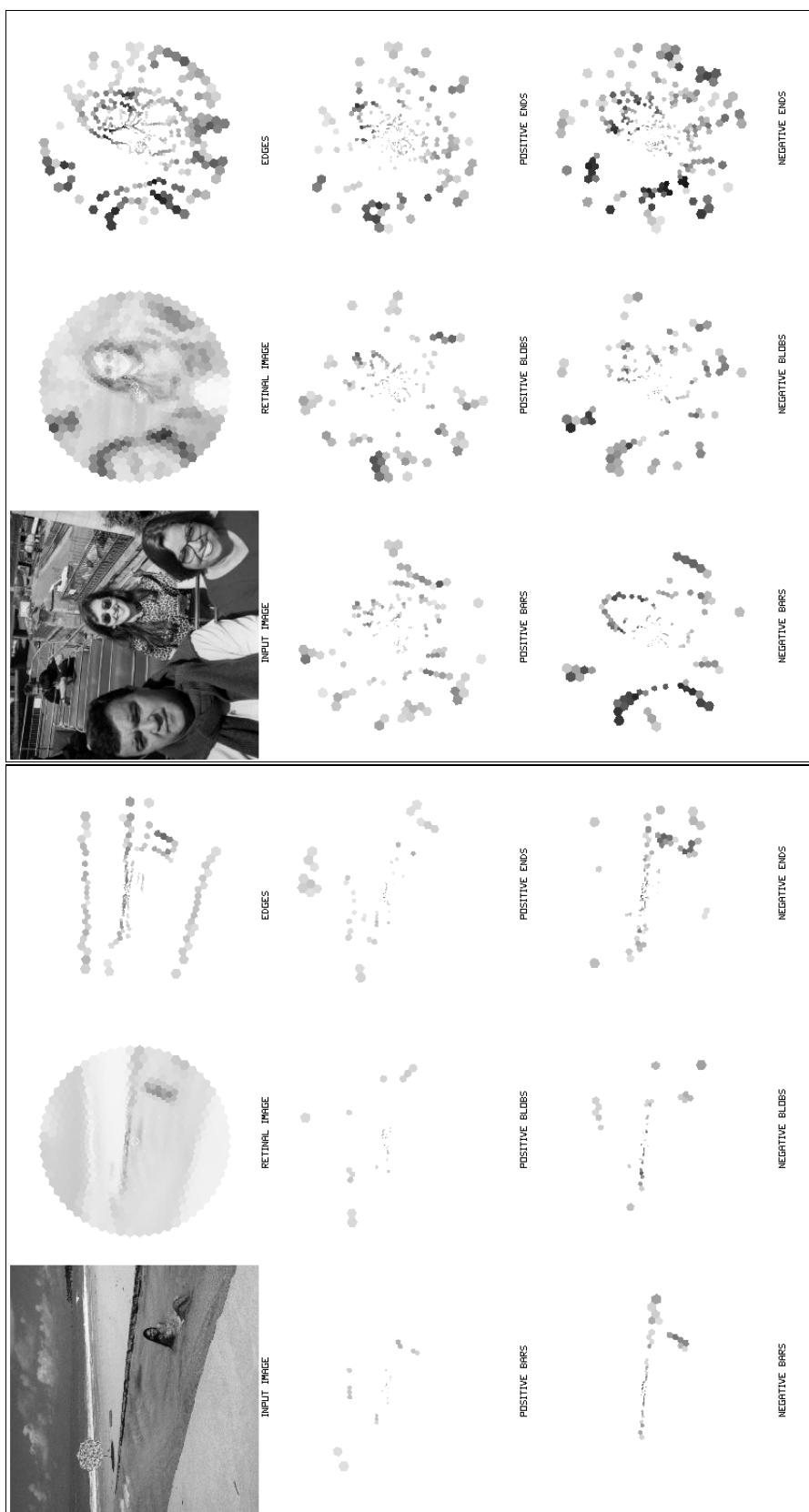


Figure A.9: Final results for test images 4 and 5.

## *Appendix B*

### *Evaluating the Similarity Function*

This appendix presents the results of the evaluation of the similarity function under the misplacement of the fixation points. Each figure represents a combination of type of object and background. Two types of objects were used: simple, having few features to discriminate from other objects; and complex, the other way around. Also, two types of background were considered: plain, formed mainly by an uniform colour or pattern; and cluttered.

The experiment consisted of initially choosing a fixation point on the object and extracting a model instance there. Then, all possible fixation points inside a circle centred on the first point were used to generate neighbouring model instances, which were in turn compared to the first model instance according to the similarity metric.

Figure B.1 shows the experiment on a simple object, plain background. Figure B.2 shows the experiment on a simple object, cluttered background. Figure B.3 shows the experiment on a complex object, plain background. Finally, Figures B.4 and B.5 show the experiment on complex objects and cluttered backgrounds.

The outer and inner circles drawn in Figures B.1(a) through B.5(a) represent the area covered by the retinal mask and the area from which misplaced fixation points were selected, respectively. Figures B.1(b) through B.5(b) shows the resulting similarities as the vertical y-axis of a 3-D graph, being the x-z axis represented by local image coordinates inside the circle. Figures B.1(c) through B.5(c) contain plots of the average similarities versus the radii.

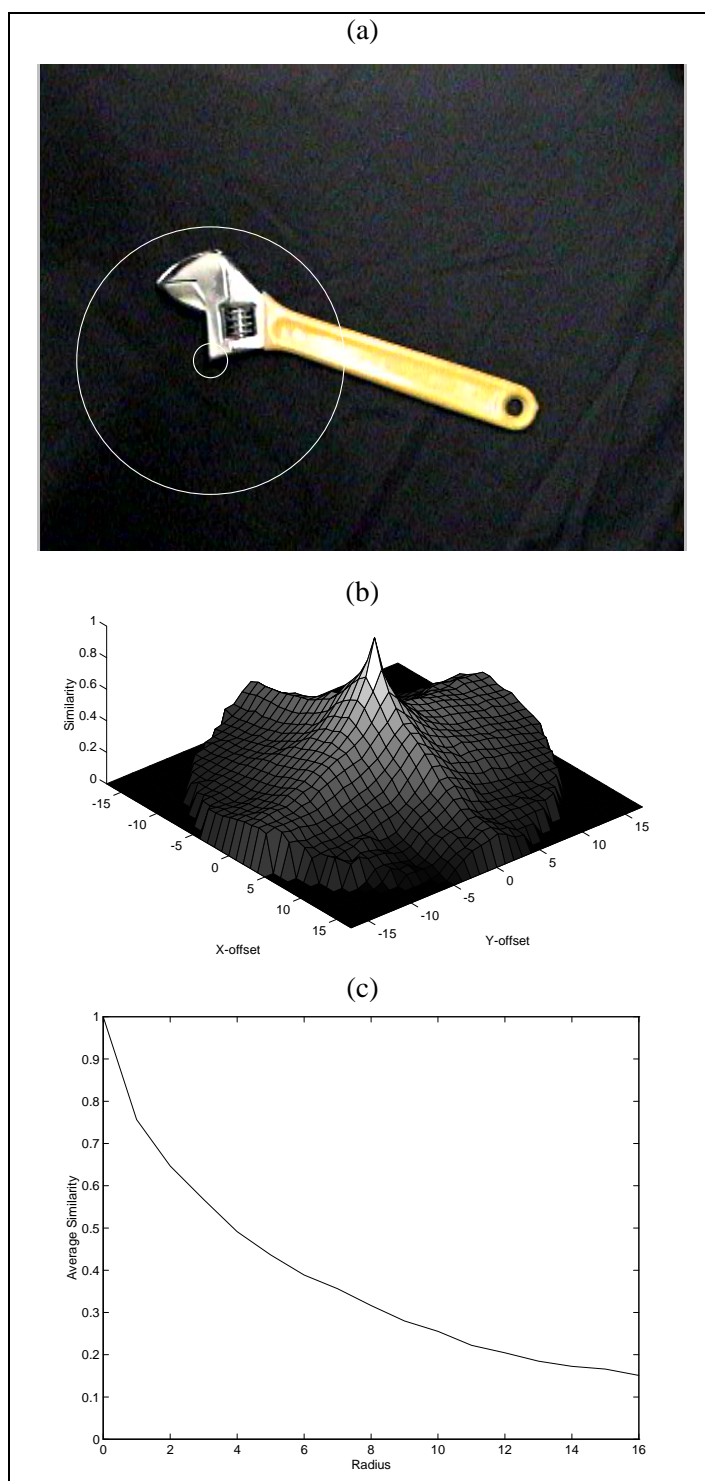


Figure B.1: Similarity experiment with *Tool1*: simple object in a plain background. (a) Input Cartesian image. (b) Plot of similarities between a model instance on the central point and all instances centred on points of the inner circle. (c) Similarities averaged over the distance (radius) from the central point.

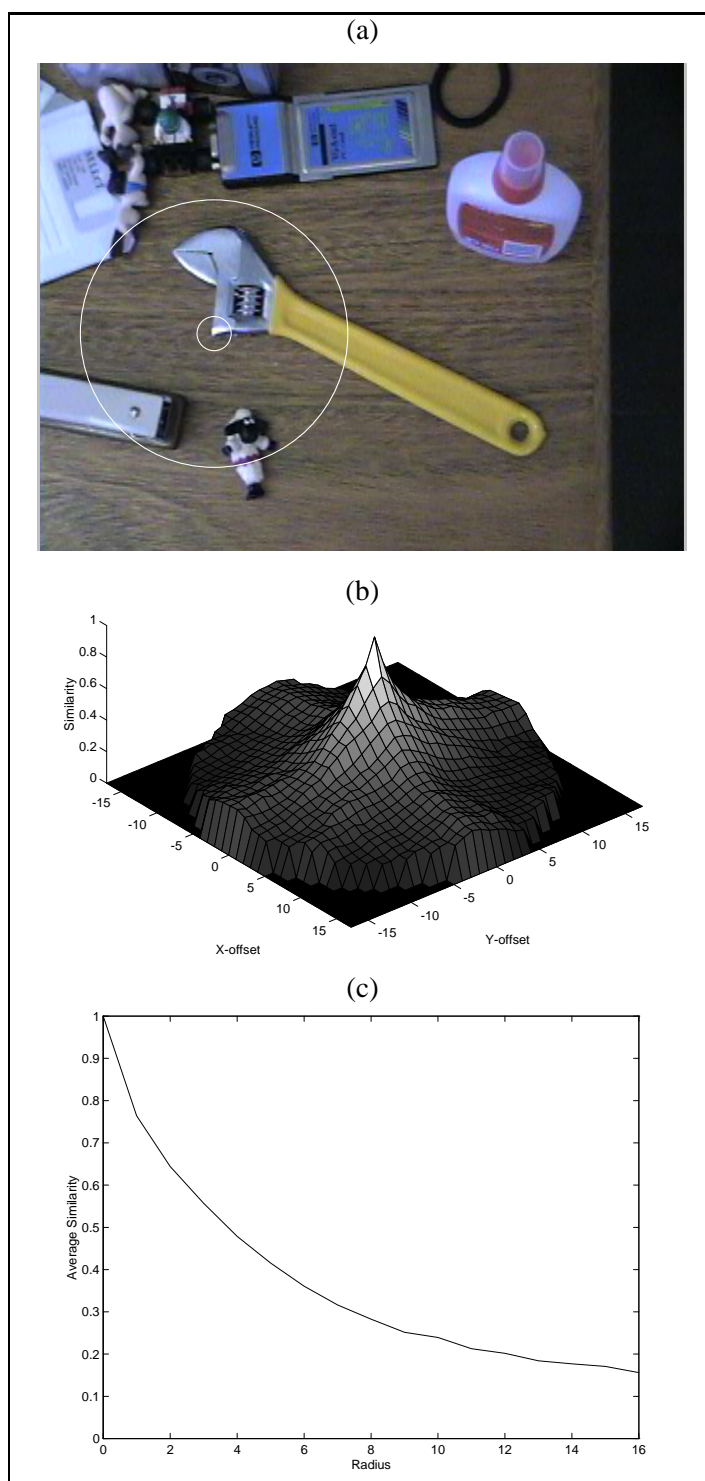


Figure B.2: Similarity experiment with *Tool2*: simple object in a cluttered background. (a) Input Cartesian image. (b) Plot of similarities between a model instance on the central point and all instances centred on points of the inner circle. (c) Similarities averaged over the distance (radius) from the central point.

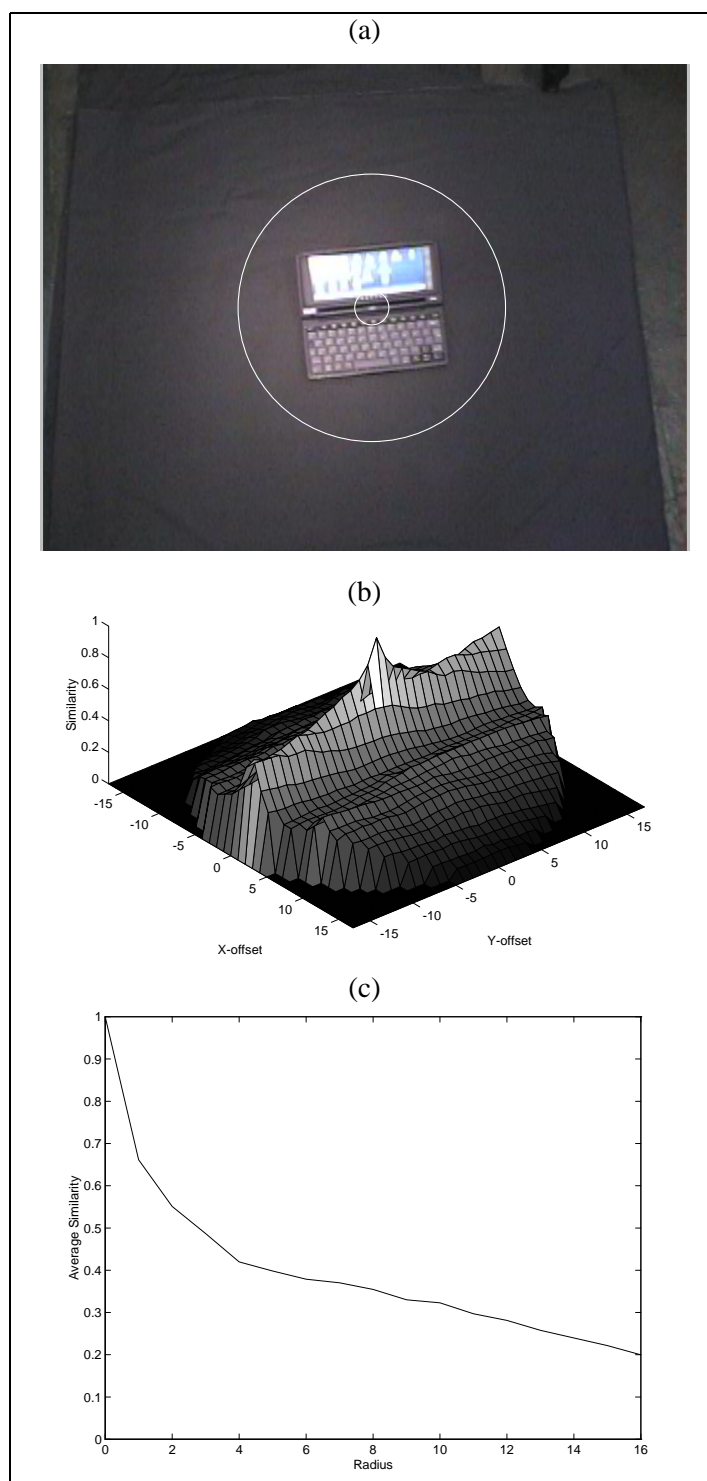


Figure B.3: Similarity experiment with *Pda1*: complex object in a plain background. (a) Input Cartesian image. (b) Plot of similarities between a model instance on the central point and all instances centred on points of the inner circle. (c) Similarities averaged over the distance (radius) from the central point.

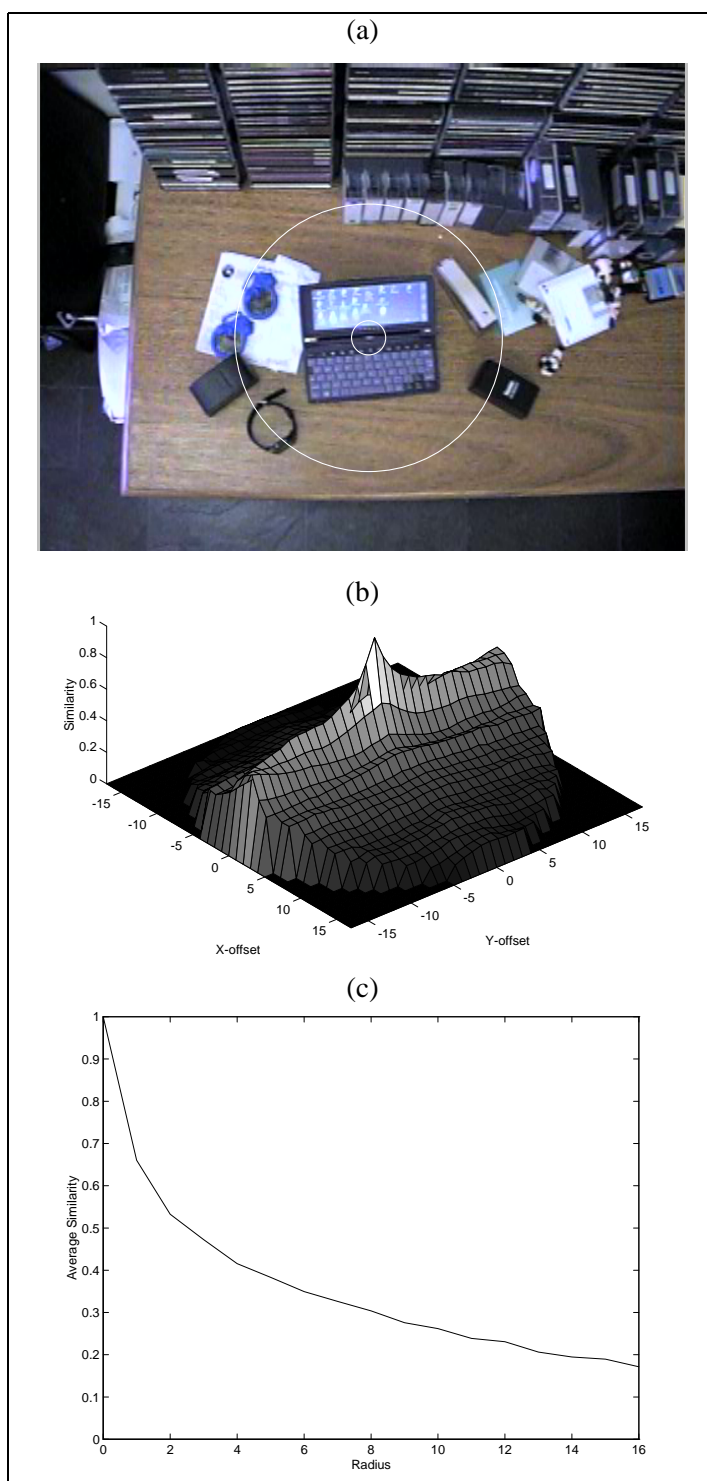


Figure B.4: Similarity experiment with *Pda2*: complex object in a cluttered background. (a) Input Cartesian image. (b) Plot of similarities between a model instance on the central point and all instances centred on points of the inner circle. (c) Similarities averaged over the distance (radius) from the central point.



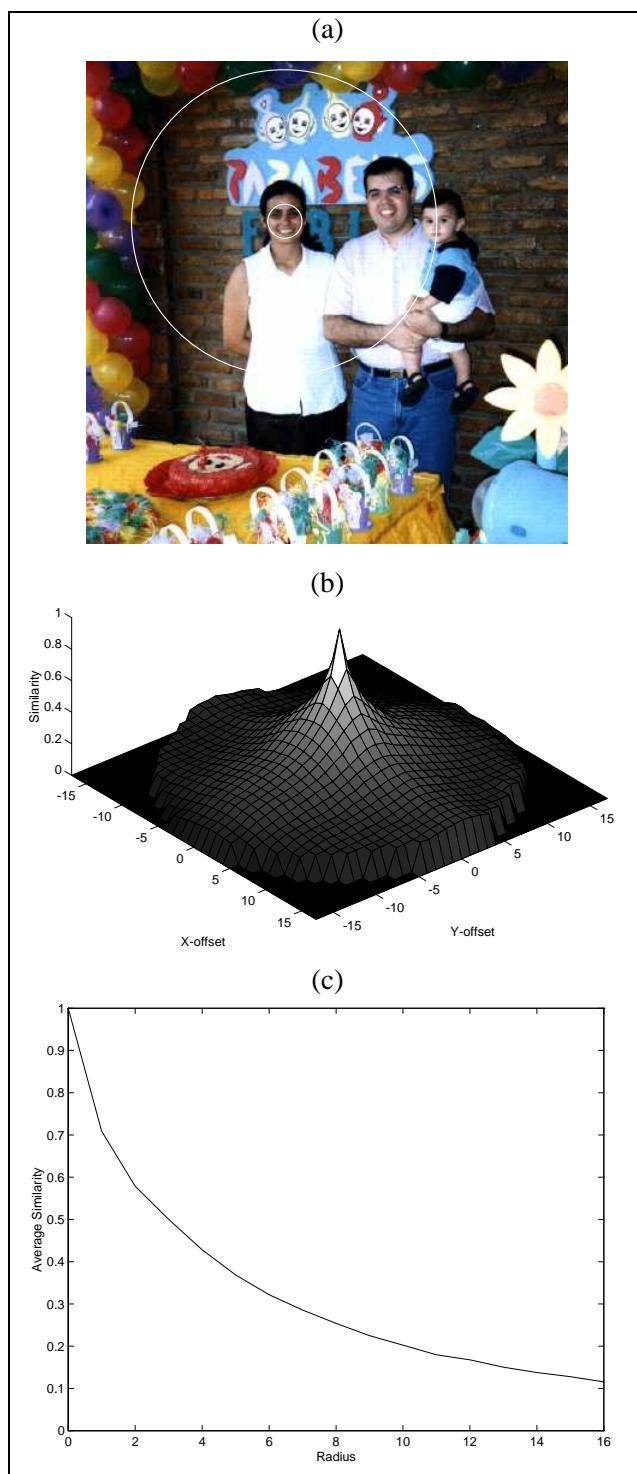


Figure B.5: Similarity experiment with *Party*: complex object in a cluttered background. (a) Input Cartesian image. (b) Plot of similarities between a model instance on the central point and all instances centred on points of the inner circle. (c) Similarities averaged over the distance (radius) from the central point.

## *Appendix C*

# *Graphs from the Combinatorics Experiments*

This appendix presents graphs from the experiments designed for analysing the combinatorics related to our structure learning approach (see Section 5.3.2, pages 135–140, for a detailed explanation).

Each of the figures contains 3 graphs, (a), (b) and (c), which are associated with a particular value of  $K$ , the number of allowed wild-cards ( $K = 0, 1, 2$ ). Each graph contains three curves, corresponding to the different values of  $L$ , the number of feature instances randomly removed from the model base per feature type ( $L = 0, 1, 2$ ).

There are 9 different figures plotting a variable against the number of scenes  $N$  in the model base. Figure C.1 shows the average total CPU time <sup>1</sup>, C.2 the average natural logarithm of the total CPU time, C.3 the number of iterations of the clique finding algorithm, C.4 the number of raw vertices, C.5 the number of raw edges, C.6 the number of pruned edges, C.7 the number of pruned vertices, C.8 the number of raw cliques, and C.9 the number of maximal cliques) Finally, Figure C.10 shows the result of fitting an exponential function to the original data of curve  $L = 2$  in Figure C.1(c) and then extrapolating this function to 30 images.

---

<sup>1</sup> We used a 700MHz Athlon processor with 256MB of RAM under the Linux operating system

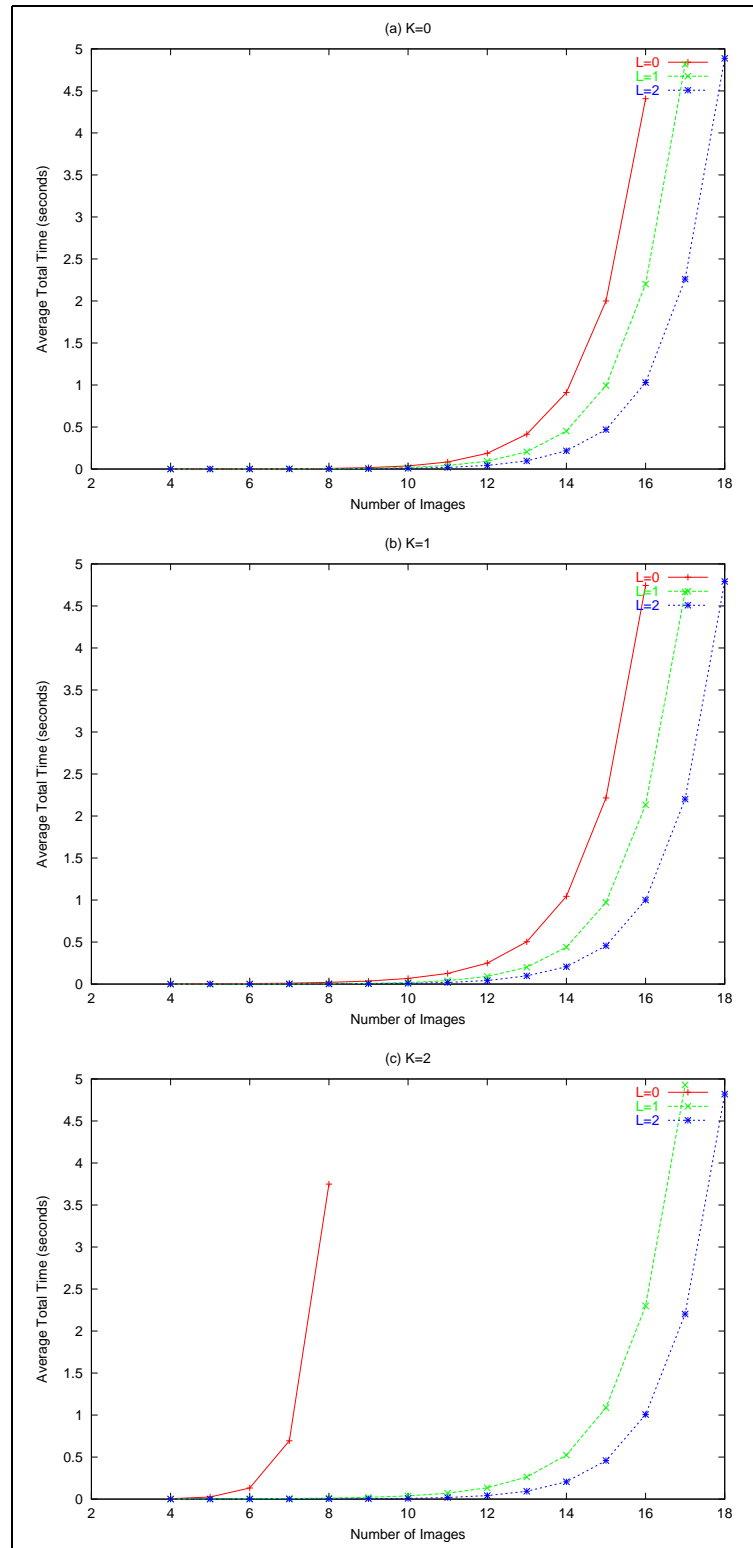


Figure C.1: Average total CPU time spent by the entire structure learning approach versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

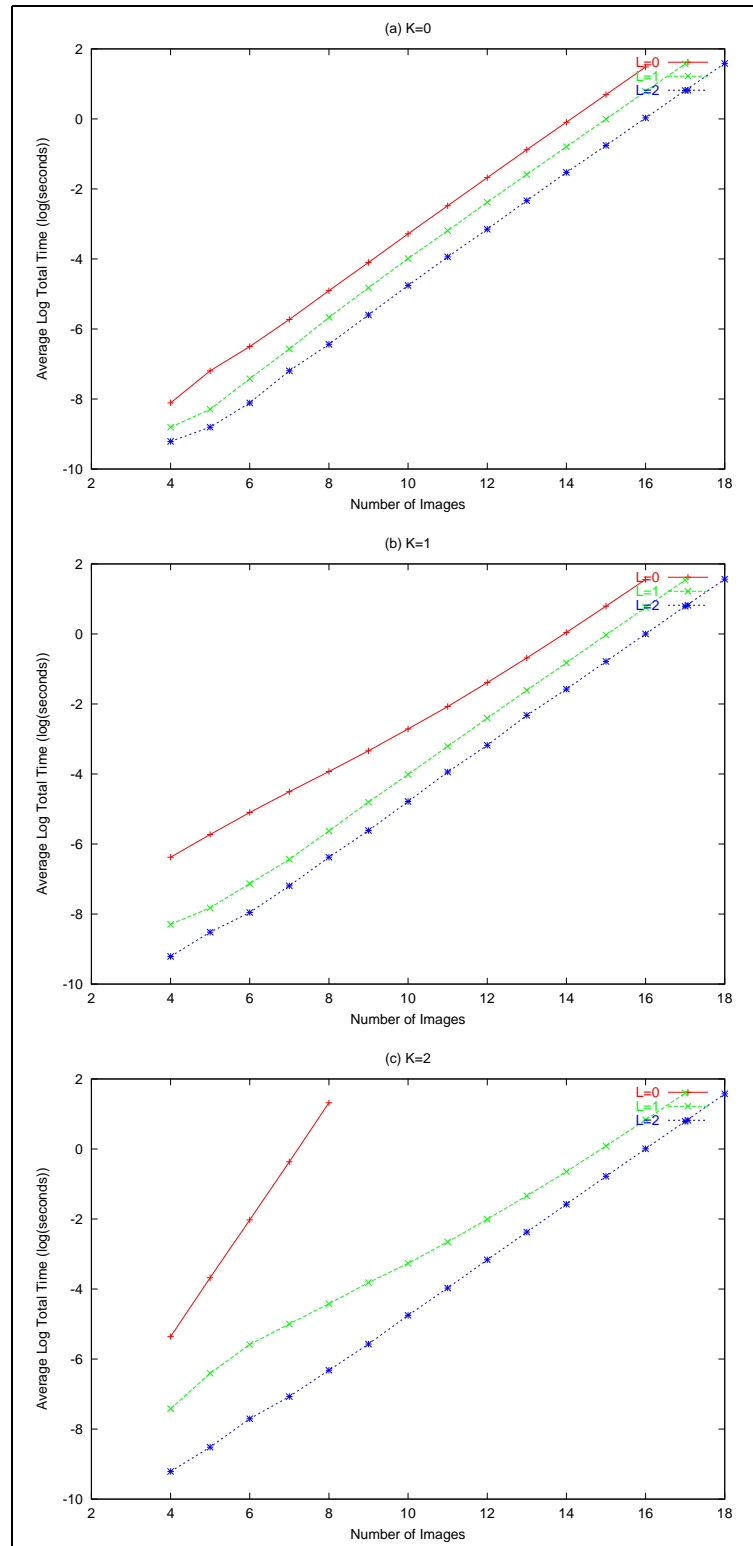


Figure C.2: Average natural logarithm of the total CPU time spent by the entire structure learning approach versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

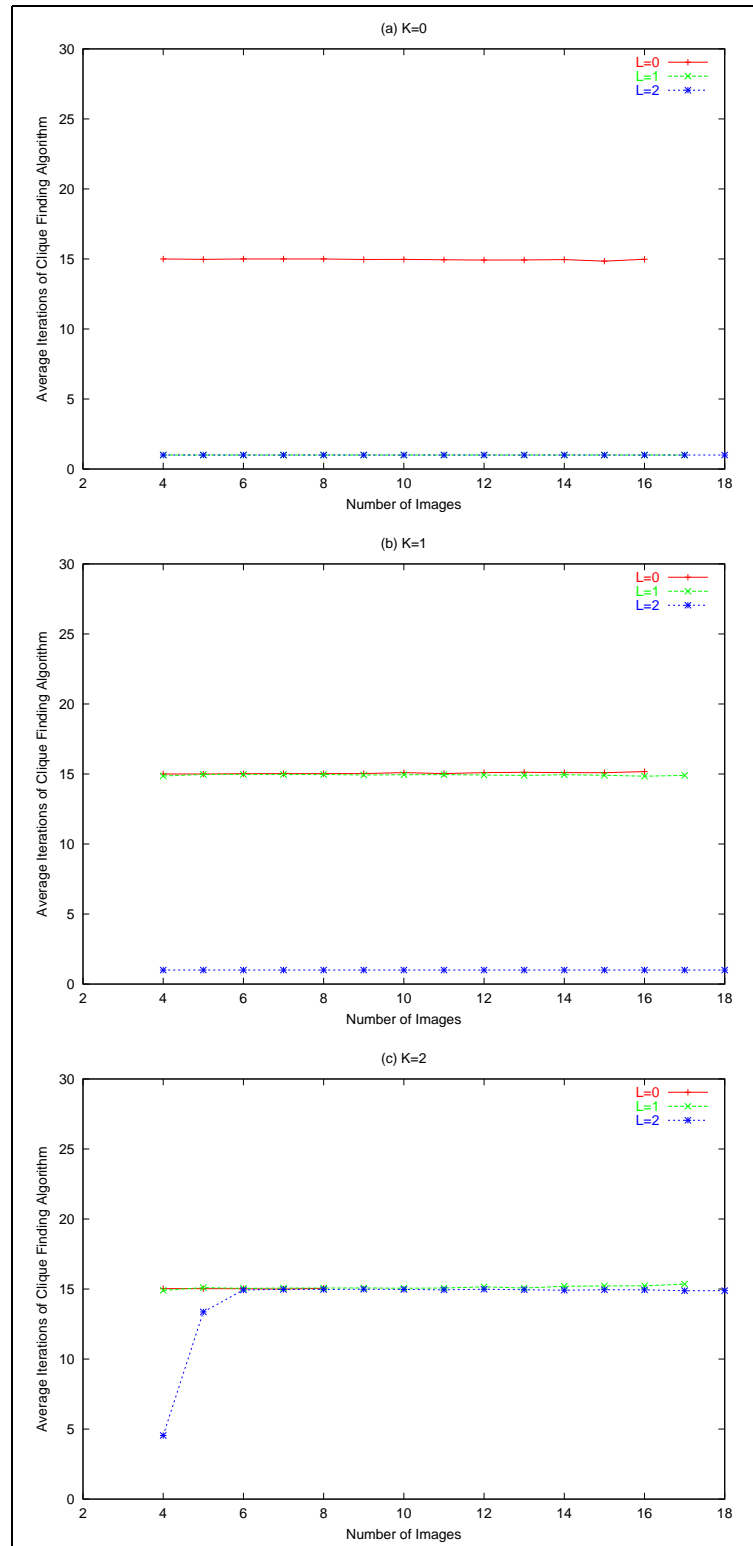


Figure C.3: Average number of iterations by the clique finding algorithm versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

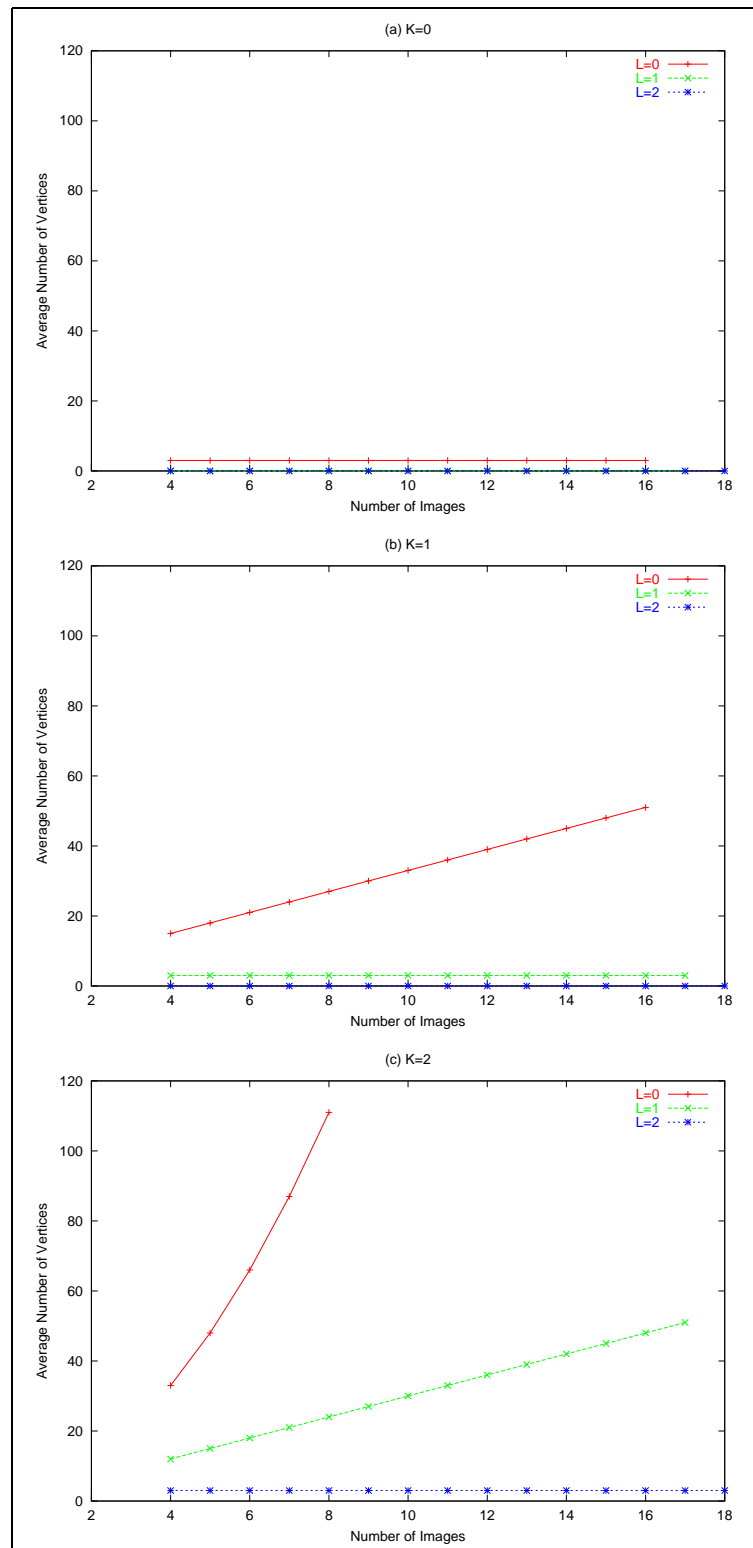


Figure C.4: Average number of vertices versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

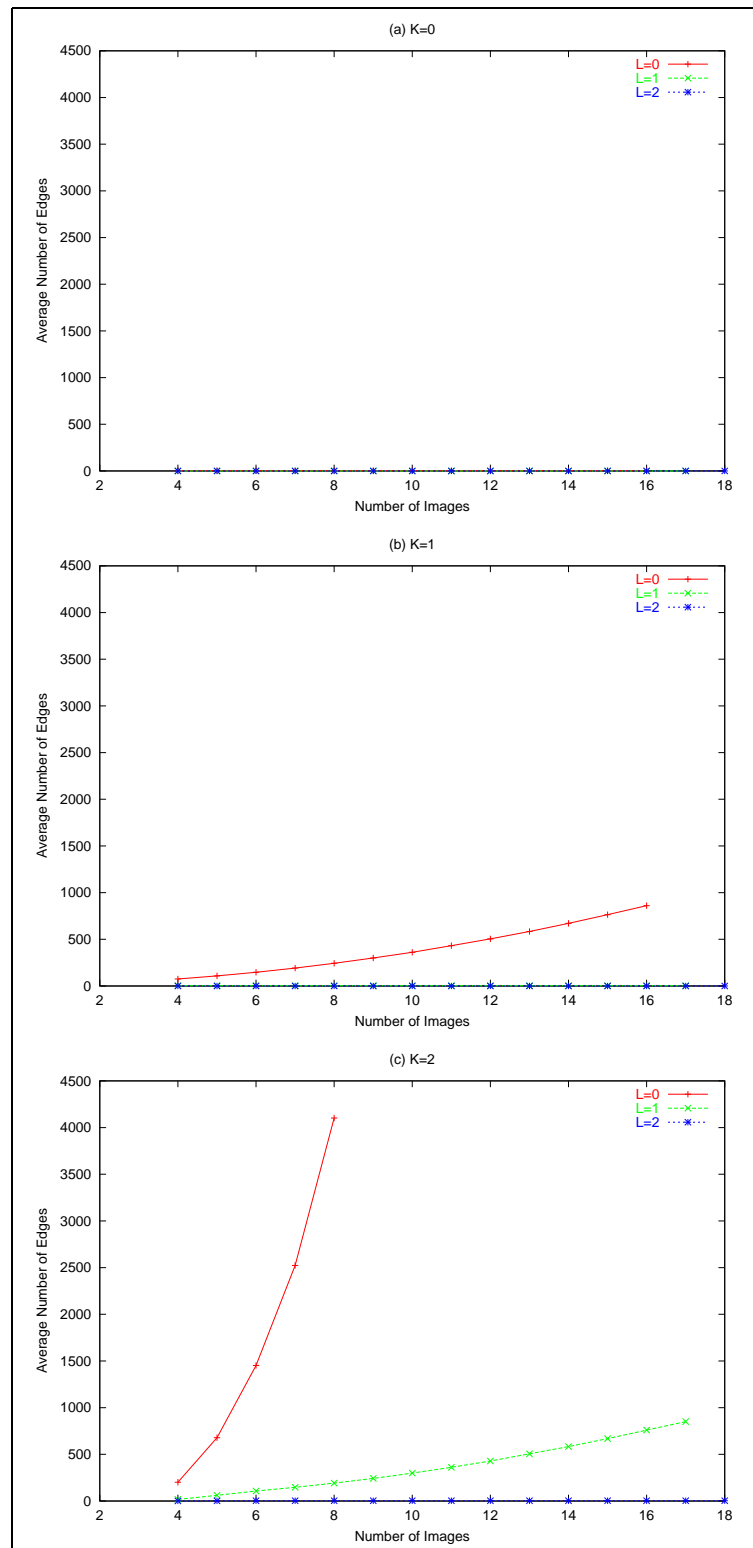


Figure C.5: Average number of edges versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

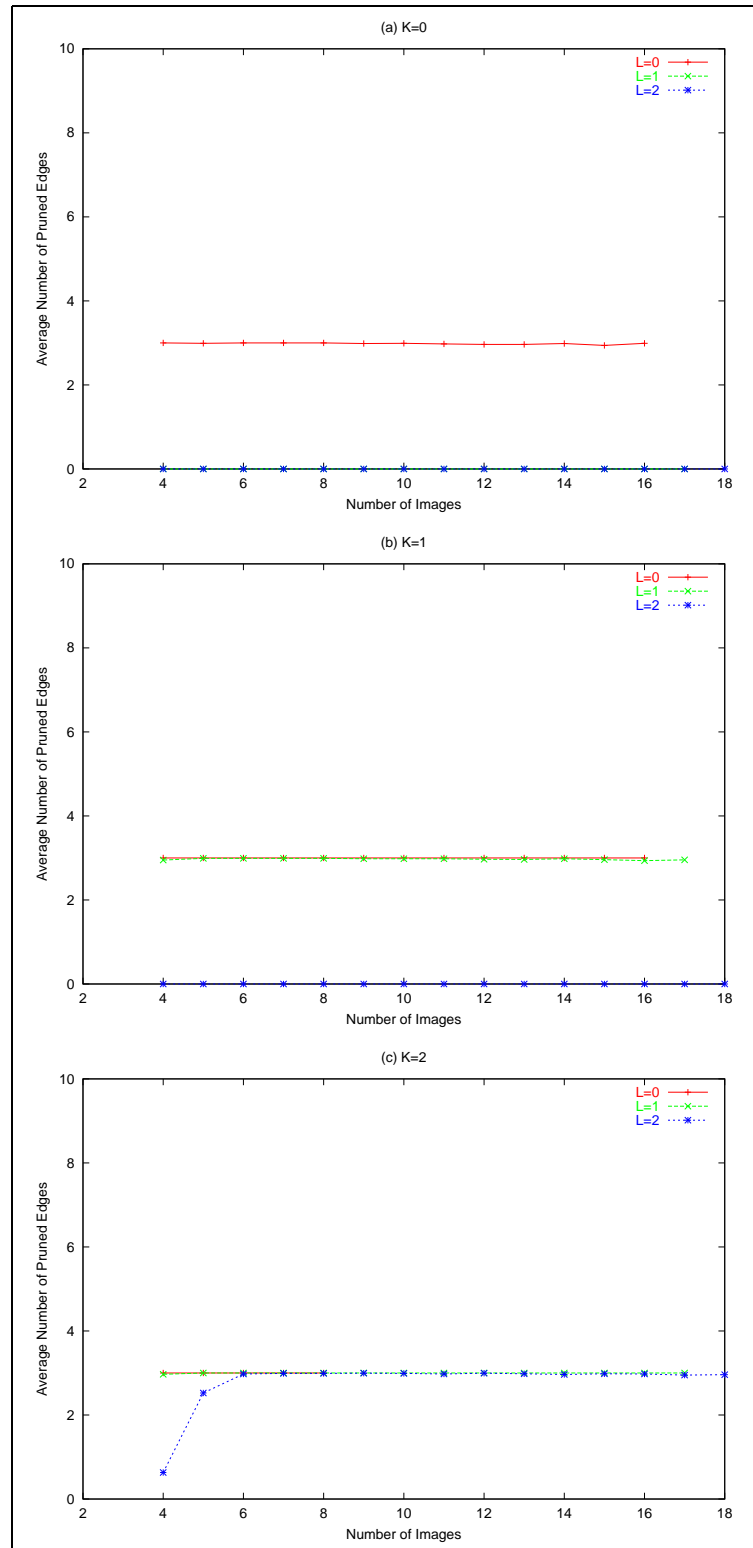


Figure C.6: Average number of pruned edges versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.



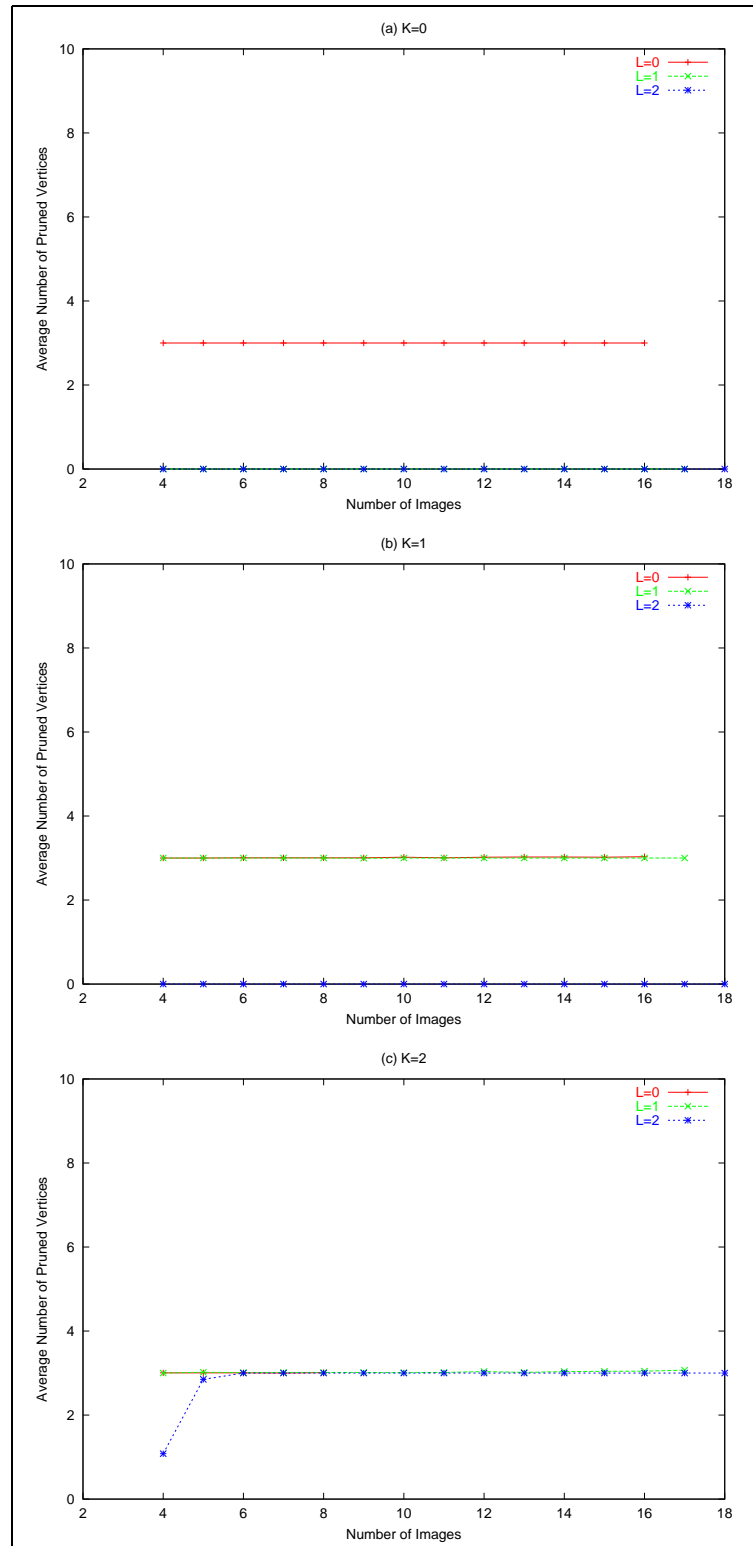


Figure C.7: Average number of pruned vertices (after edge pruning) versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

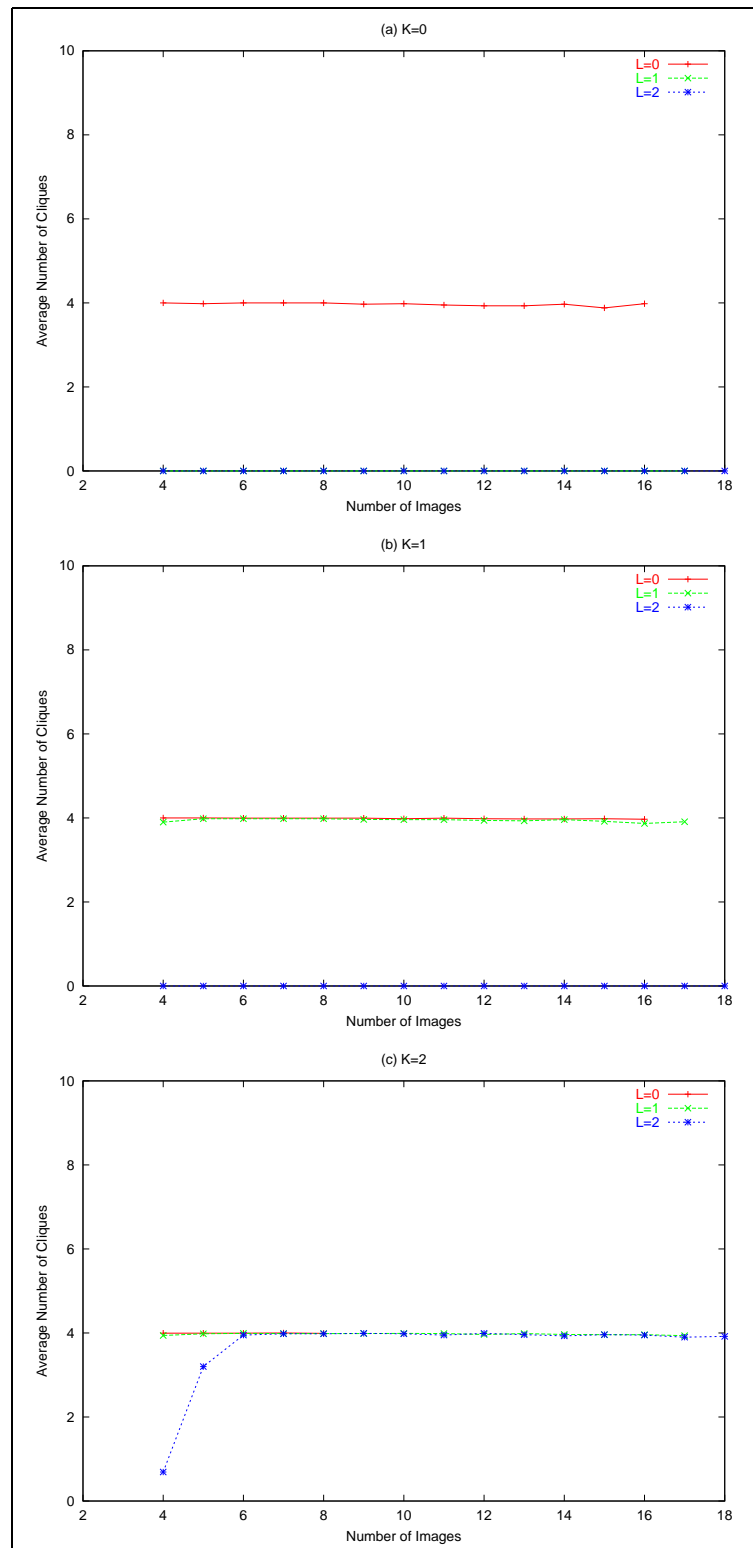


Figure C.8: Average number of (raw) cliques found versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

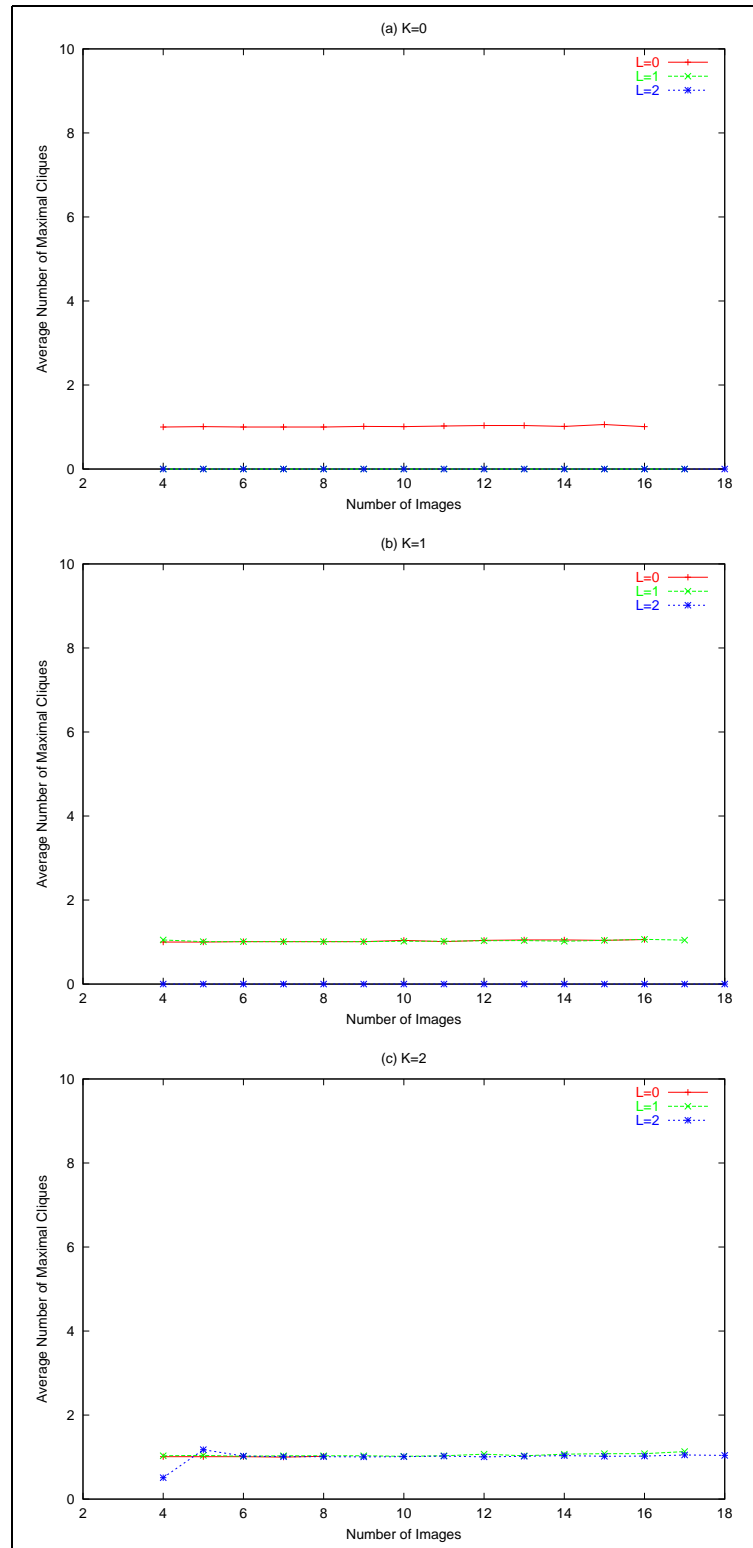


Figure C.9: Average number of maximal cliques found versus the number of scenes  $N$ , allowing  $K = 0, 1, 2$  wild-cards per vertex and removing  $L = 0, 1, 2$  occurrences of each feature type.

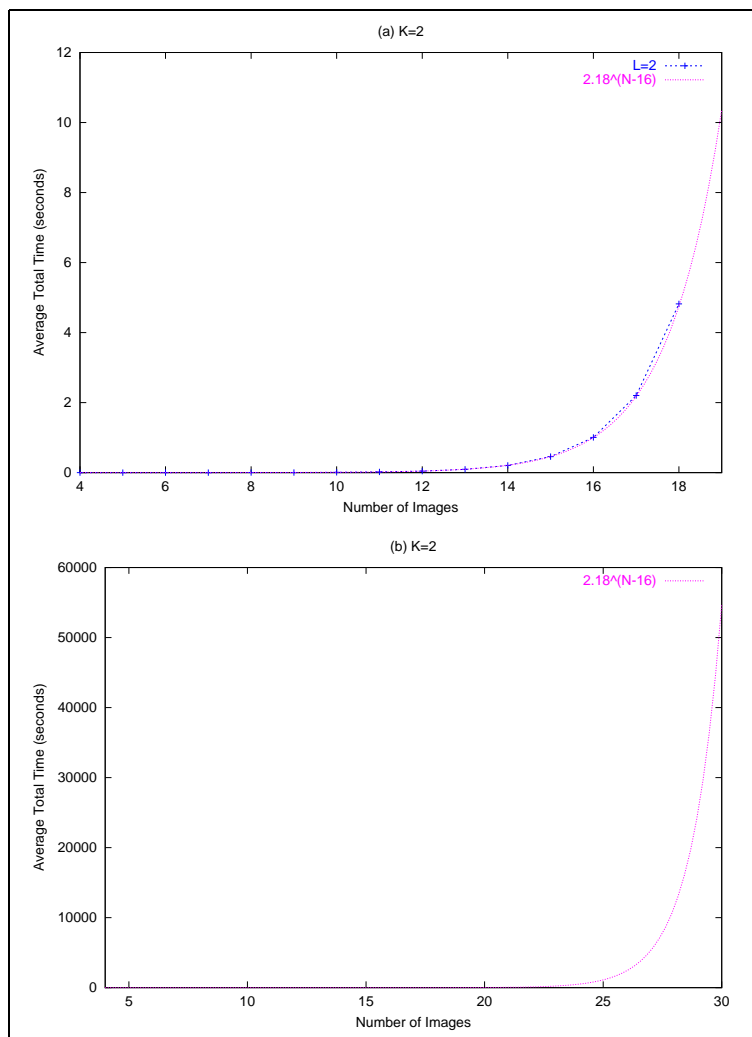


Figure C.10: (a) Fitting an exponential function to the original data of curve  $L = 2$  in Figure C.1(c); (b) extrapolating this function to 30 images.